Meowlang Programming Language (Final Report)

Language Guru: Carolyn Chen (cec2192)

Manager: Megan Frenkel (mmf2171)

System Architect: Lauren Pham (lyp2106)

Tester: Michelle Lin (ml4080)

Programming Languages and Translators
Spring 2021

Contents

1 Introduction

`Meowlang` is an object-oriented esoteric programming language inspired by LOLCODE, a language created by Adam Lindsay based on internet lolspeak. `Meowlang` pushes the boundaries of language design in a creative, humorous way while still providing functionality and usability. The greatest difference between `Meowlang` and the typical programming language is that `Meowlang` fundamentally replaces symbols with keywords. But underneath its playful exterior, `Meowlang`'s syntax and structure strongly resemble that of the C language. As an imperative object-oriented programming language, `Meowlang` provides the powerful object-oriented feature of encapsulation in the form of classes. Combined with the familiar syntax and structure of C language, programming in `Meowlang` is intuitive even for beginners.

1.1 Language Paradigm and Features

`Meowlang` is heavily inspired by LOLCODE, sharing similar syntax and, in some cases, keywords. The LOLCODE language reference manual is located at the following link: `https://github.com/justinmeza/lolcode-spec/blob/master/v1.2/lolcode-spec-v1.2.md`.

However, `Meowlang` has some key differences in its language paradigm. `Meowlang` is a statically typed object-oriented programming (OOP) language while LOLCODE is a dynamically typed general-purpose programming language that does not support OOP. `Meowlang` improves upon LOLCODE in terms of safety and functionality by adopting a different type system and by introducing a number of new features, including arrays, imports, several built-in functions and perhaps most notably, classes. Though inheritance and polymorphism are not currently supported by `Meowlang`, these are key areas of future expansion.

1.2 Goals

1.2.1 Functionality

`Meowlang` aims to provide as much functionality as a non-esoteric programming languages, but with more playful syntax. With this goal in mind, `Meowlang` supports classes, arrays, import modules and built-in functions. Arrays are an important and versatile construct in programming, while the introduction of object-oriented programming allows the creativity of the esoteric language to shine and provides helpful constructs for organizing data to `Meowlang` programmers. Supporting imports allows programmers to better organize their code and reuse helpful functions and classes in many different programs. Meanwhile, built-in functions to support casting, concatenation, and I/O provide convenience for the programmer. These features ensure that `Meowlang` is powerful enough to support a significant range of applications, setting it apart from other esolangs.

1.2.2 Usability

Unlike most esoteric programming languages, `Meowlang` was developed with usability taken into great consideration. While lolspeak mangles the English language, `Meowlang`'s lexicon is intended to be comprehensible. Careful choice of keywords and an overall structure inspired by the C language make coding in `Meowlang`

intuitive. The language is designed so that even programmers with a shallow understanding of the language syntax can grasp what the code is doing. Further support for usability is expanded on in **3.1 Conventions**.

1.2.3 Esolang Aesthetics

At its core, `Meowlang` is an esoteric programming language and therefore must maintain its unique esotericism. Many of our syntactical choices prioritized establishing an esoteric aesthetic, and in particular, visually emulating natural language. For example, `Meowlang` makes use of commonly used punctuation, such as question marks, commas and periods, as well as `Meowlang` -specific keywords in place of symbols. With these syntactical adjustments in place, `Meowlang` blurs the line between code and natural language, earning a place in the esolang family.

2 Language Tutorial

2.1 Installing the Meowlang Compiler

The following list represents the prerequisite software, with their respective versions, required in other to make use of the Meowlang compiler. It is likely that other software versions are also sufficient, though these settings were used in development:

- Ocaml (version 4.11.1)
- LLVM (version 8.0.1)
- Opam (version 2.0.7)
- gcc (clang version 12.0.0)

The Meowlang compiler itself can be cloned from https://github.com/mmfrenkel/meowlang, a public repository. In order to setup the compiler, simply run the following from the root directory of the repository:

```
$ make
```

Once the compiler is compiled, you're free to run any .meow program using the meowlang.sh script found in the bin directory. You must provide the relative path to the .meow file as a command line argument. For example, from the root directory:

```
$ ./bin/meowlang.sh <relative path to .meow file>
```

Other command line arguments for debugging your program and extra testing scripts are outlined in the compiler README found at https://github.com/mmfrenkel/meowlang.

2.2 Your First Program

So you have the Meowlang compiler downloaded and compiled on your computer. Now it's time to write your very first program in Meowlang! In this demo, we'll take you step-by-step through building the classic hello-world program.

Note that every program in Meowlang must have one and only one Main function, which exists as an entry point to the program. We'll begin by declaring the function Main with the following line:

```
HAI ITZ ME NUMBR FUNC Main,
```

You will notice that every declaration in Meowlang includes the keywords ITZ ME followed by a type, here FUNC (for function), and then an identifier, here Main. After ITZ ME, we specify the return type. This function happens to return an integer, so we specify the Meowlang-equivalent, NUMBR. If the function is void, you can simply not provide anything for the return type.

Next, let's declare and initialize a string variable, using the same ITZ ME syntax. Note that strings in Meowlang are actually called YARNs. Below, we declare a string variable message and set its value to "Hello, world!".

```
ITZ ME YARN message IZ "Hello, world!".
```

Note that the `IZ` keyword is the assignment operator, equivalent to an equals sign. Don't forget to end the statement with a '.' period at the end; this is a common mistake that results in a parsing error! Next, we will use `Meowlang`'s built-in print function `Meow` to print our `message` to standard output:

```
PURR Meow WIT message.
```

The `PURR` keyword indicates a function call and immediately precedes a function identifier, in this case `Meow`. `WIT` precedes the argument(s) to be passed into a function. If there were additional arguments to add, you'd separate the arguments using the keyword `AN`.

In order to remember that `Meow` is a built-in function, maybe you'd like to write a comment. In this case, simply preface your message with the keyword `PSST` as shown below:

```
PSST Meow is a built-in print function!
```

To finish out our function, we want to return a 0 to indicate success. To do this, we make use of the `GIVE` keyword to return the integer literal 0. We end the `Main` function with the last required keyword, `KBYE`.

```
    ...
    GIVE 0.
KBYE
```

Note that outside of class and function declarations, `HAI` and `KBYE` operate similarly to curly braces in other languages. More info can be found in **3.1.4 Blocks and Scope**.

Important newbie note: `Meowlang` is whitespace insensitive. This means that you can add spaces and tabs liberally and it will not impact your program. However, it is best practice to conform to `Meowlang` conventions of using 1 level of indentation (size 4 spaces) in a function body. After programming in `Meowlang` for a little more time, you'll see how this will preserve the readability of your programs.

Last but not least, save your program as `hello_world.meow` and run with with the `meowlang.sh` script. There you have it, your very first program in `Meowlang`! See full and completed program below.

```
$ ./bin/meowlang.sh hello_world.meow
```

Listing 1: `hello_world.meow`

```
1 HAI ITZ ME NUMBR FUNC Main,
2
3     ITZ ME YARN message IZ "Hello, world!".
4
5     PSST This is a built-in function!
6     PURR Meow WIT message.
```

```
7       GIVE 0.
8
9   KBYE
```

For the sake of science, try making another version of this program typed in lowercase. What happens when you try to run the program? (PSST: `Meowlang` is case-sensitive!)

2.3 Your Second Program

Now that you've mastered the basics of `Meowlang`, that is, working with functions and variables, you're ready to move on to some more advanced features like classes, arrays and imports. For this demo, we're going to create a program that allows a user to pick from a list of pets at a pet store. Every pet has a name, age, and species. The user will submit a number representing the animal that they wish to take home.

The first thing to notice is that our pets in this program have many attributes. Additionally, there are probably things we want to do for the pet or get from them, like feed them or rename them once you take them home. How should we handle this? This is the perfect use-case for classes!

Defining a new class in `Meowlang` is a lot like defining a new function, except that we must utilize the keyword `CLASS`. We'll start off creating a shell for our new `PET` class as follows:

```
HAI ITZ ME CLASS PET,
    ...
KBYE
```

We want each instance of our `PET` class to have a few defining attributes – let's add them! We can do this simply by writing variable declarations underneath the class declaration like so:

```
HAI ITZ ME CLASS PET,
    ITZ ME YARN name.
    ITZ ME YARN species.
    ITZ ME NUMBR age IZ 0.
    ...
KBYE
```

Notice that we used an assignment statement to define a default value for the age of each `PET`. This is strictly optional, but can be useful if you expect many objects to start with the same value.

Didn't we also say that pets can be fed and renamed? Let's add some class methods to make this happen. Class methods share the same syntax as functions and we can access class variables directly within them. We'll add four methods, one for changing the pet's name, another for feeding them, and two others for getting the current pet name and its species. These are shown below:

```
HAI ITZ ME YARN FUNC Get_Name,
    GIVE name.
KBYE

HAI ITZ ME YARN FUNC Get_Species,
    GIVE species.
KBYE

HAI ITZ ME FUNC Rename WIT YARN new_name,
    name IZ new_name.
KBYE

HAI ITZ ME FUNC Feed,
    PURR Meow WIT "Nom nom".
KBYE
```

This is sufficient for our `PET` class. Let's save it in a file called `pets.meow`.

What else are we missing? Remember that every program needs a "Main" function. This main function will keep track of all the pets in the store. Let's write that next in a new file called `pet_store.meow`. In order to access our `PET` class that lives in `pets.meow`, we'll need to import it. Do so by writing the following at the top of your new file:

```
GIMME PETS?
```

Next, we'll create a few different possible pets: a cat, a dog and a rabbit. Below, we initialize the instance variables for our cat and dog utilizing `Meowlang`'s object constructor support. To demonstrate how `Meowlang` allows you to set these values after the object is created as well, we create the rabbit in multiple statements. Note that when we're done with our pets, we must remember to free their memory using the keyword `BLEEP`.

```
 HAI ITZ ME NUMBR FUNC Main,

    ...
    PSST Make some pets
    MAEK Silvester NEW PET,
        WIT name IZ "Silvester"
        AN species IZ "cat"
        AN age IZ 4.

    MAEK Tank NEW PET,
        WIT name IZ "Tank"
        AN species IZ "dog"
        AN age IZ 2.
```

```
    MAEK Hopper NEW PET.
    name IN Hopper IZ "Hopper".
    species IN Hopper IZ "Rabbit".
    age IN Hopper IZ 10.

    ...

    BLEEP Hopper.
    BLEEP Silvester.
    BLEEP Tank.
    GIVE 0.

KBYE
```

In order to keep track of all the pets, let's make use of `Meowlang` arrays, which are called `BUCKET`s. You can make a new `BUCKET` that holds items of type `PET` with the following statement, which also sets the contents of the array with the pets we just created. Note that when we're done with the array, we need to free its memory:

```
MAEK store_pets NEW BUCKET OF PET HOLDS 3,
    WIT Silvester
    AN Tank
    AN Hopper.
...

BLEEP store_pets.
```

Great! The next thing we want to do is ask the user which pet they want to pick to take home with them. We have three pets, so we ask the user to provide the value 0, 1, or 2. They don't know which pet they'll be choosing when they provide this number, so it will be a surprise! To accomplish this, we can make use of `Meowlang`'s built-in I/O functionality using the `Scan` function. Note that although the user provides an integer value, all values read from user input are initially stings; hence we must case the string to an integer in order to use it as an index in our array.

```
...
ITZ ME YARN user_value.
ITZ ME NUMBR user_selection.
...
PURR Meow WIT "Please pick a pet by specifying a value between 0 and 2!".
PURR Scan WIT user_value.
user_selection IZ NUMBR user_value.
...
```

Now that we have the user's selection, we can tell them the name and species of animal they're bringing home! Let's index into the array to get the selected pet, build up a concatenated string with `Meowlang's` concatenation functionality, and finally print it out!

```
...
ITZ ME YARN pet_name.
ITZ ME YARN pet_species.
...
pet_name IZ PURR Get_Name IN store_pets[user_selection].
pet_species IZ PURR Get_Species IN store_pets[user_selection].
PURR Meow WIT CAT "You're bringing home a " AN CAT pet_species AN CAT " named " AN pet_name.
```

Save your program as `pet_store.meow`. Below is our completed program, consisting of two files. All that's left to do is run it!

```
$ ./bin/meowlang.sh pet_store.meow
```

Note that in a real program, you'd probably want to do some user-input validation to be sure that they provided a valid integer value. How would you do this? (PSST use if-statements in `Meowlang` to check if the user-input is within the valid index range of the pet array!)

Listing 2: `pets.meow`

```
1  HAI ITZ ME CLASS PET,
2      ITZ ME YARN name.
3      ITZ ME YARN species.
4      ITZ ME NUMBR age IZ 0.
5
6      HAI ITZ ME YARN FUNC Get_Name,
7          GIVE name.
8      KBYE
9
10     HAI ITZ ME YARN FUNC Get_Species,
11         GIVE species.
12     KBYE
13
14     HAI ITZ ME FUNC Rename WIT YARN new_name,
15         name IZ new_name.
16     KBYE
17
18     HAI ITZ ME FUNC Feed,
19         PURR Meow WIT "Nom nom".
20     KBYE
21 KBYE
```

Listing 3: pet_store.meow

```
GIMME PETS?

HAI ITZ ME NUMBR FUNC Main,

    ITZ ME YARN user_value.
    ITZ ME NUMBR user_selection.
    ITZ ME YARN pet_name.
    ITZ ME YARN pet_species.

    PSST Make some pets
    MAEK Silvester NEW PET,
        WIT name IZ "Silvester"
        AN species IZ "cat"
        AN age IZ 4
        AN store_id IZ 0.

    MAEK Tank NEW PET,
        WIT name IZ "Tank"
        AN species IZ "dog"
        AN age IZ 2
        AN store_id IZ 1.

    MAEK Hopper NEW PET.
    name IN Hopper IZ "Hopper".
    species IN Hopper IZ "Rabbit".
    age IN Hopper IZ 10.
    store_id IN Hopper IZ 2.

    MAEK store_pets NEW BUCKET OF PET HOLDS 3,
        WIT Silvester
        AN Tank
        AN Hopper.

    PSST Ask the user which pet they want
    PURR Meow WIT "Please pick a pet by specifying a value between 0 and 2!".
    PURR Scan WIT user_value.
    user_selection IZ NUMBR user_value.

    PSST Report back!
```

```
40      pet_name IZ PURR Get_Name IN store_pets[user_selection].
41      pet_species IZ PURR Get_Species IN store_pets[user_selection].
42      PURR Meow WIT CAT "You're bringing home a " AN CAT pet_species AN CAT "
        named " AN pet_name.
43
44      BLEEP Hopper.
45      BLEEP Silvester.
46      BLEEP Tank.
47      BLEEP store_pets.
48      GIVE 0.
49
50  KBYE
```

3 Language Reference Manual

3.1 Conventions

3.1.1 Whitespace

Spaces are used to demarcate tokens in `Meowlang`, although some keyword constructs may include spaces (see **3.1.3**). However, `Meowlang` is not whitespace sensitive; multiple spaces and tabs are treated as single spaces and are otherwise irrelevant. Indentation is also irrelevant. This means that statements may span multiple lines, as long as the end of the statement is properly marked using the "`.`" character, as explained in section **3.1.4**. Nevertheless, it is recommended to make thoughtful use of tabs, new lines and spaces and use the `Meowlang` conventions illustrated in the following code samples to maximize readability.

3.1.2 Identifiers

Identifiers in `Meowlang` cannot be strictly numeric (i.e., '123'). Convention dictates that identifiers do not begin with numeric characters or underscores, but they can be a combination of alphanumeric characters and underscores. Though not strictly mandatory for compilation purposes, it is strongly recommended that programmers use the following conventions for different types of identifiers to improve the readability of their program:

- Imports: Import names should begin with a capitalized letter from A-Z, followed by other capitalized letters, numbers and underscores.
- Classes: New classes are given in all capitalized characters, followed by other capitalized letters, numbers and underscores.
- Functions: The first character of a function name is capitalized, followed by any sequence of lower case letters, numbers and underscores. It is generally best if the letter following any underscores is also capitalized.
- Variables: Variables begin with a lowercase letter, followed by other lowercase letters, numbers and underscores. An exception is made for object identifiers: unlike other variables, they should follow the function identifier convention above.

Code examples provided in this reference manual illustrate these identifier conventions.

3.1.3 Keywords

`Meowlang` utilizes a set of "reserved" words or keywords that cannot be used as identifiers. Keywords have all capital letters and are case-sensitive. The following is a list of keywords `Meowlang` relies on, by category:

Listing 4: `keywords.meow`

```
1 PSST (1) Structure and Control of Flow
2 IZ GIMME HAI ITZ ME KBYE GIVE WIT R AN NEW MAEK BLEEP PURR O RLY? YA RLY NO
      WAI IM IN YR LOOP UPPIN NERFIN, HOLDS
3
```

```
PSST (2) Types
CLASS FUNC YARN BOO AYE NAY NUMBR NUMBAR BUCKET

PSST (3) Operators
CAT SUM DIFF PRODUKT QUOSHUNT MOD BIGGR SMALLR BOTH EITHER NOT SAEM DIFFRINT
    THAN OF
```

3.1.4 Blocks and Scope

As a whitespace-insensitive language, `Meowlang` requires programmers to make use of two different constructs to denote (a) the end of statements and (b) the beginning and end of function and class definitions.

All statements in `Meowlang` must end with the period (`"."`) character to indicate completion. The use of "." is analogous to the use of the semi-colon in the C programming language and should be used in the same way.

The keywords `HAI` and `KBYE` denote the beginning and end of both class and function declarations, respectively. These keywords enclose the statements relevant to the function/class, similar to the way curly brackets are used in C, except that the opening bracket in `Meowlang` *precedes* the function/class definition. Variables declared within a `HAI` and `KBYE` block are local variables available only within the scope of these "brackets."

3.1.5 Comments

The `PSST` keyword precedes every comment. Comments continue until the end of the line for both single line comments and comments inline with code. There are no multi-line comments.

Listing 5: `comment.meow`

```
PSST This is a valid single-line comment
ITZ ME YARN kitty IZ "Furry".  PSST Comments inline w/ code
```

3.2 Built-In Data Types

Four data types are supported by `Meowlang` out of the box: strings (`YARN`), integers (`NUMBR`), floats (`NUMBAR`) and booleans (`BOO`).

3.2.1 Strings

What other languages refer to as a "string" `Meowlang` refers to as an instance of the `YARN` data type. `YARN` literals must be demarcated by double quotation marks, and the content of a string can be any sequence of characters. Under the hood, strings are arrays of characters, but there is no separate type for characters. In other words, a string of length 1 is still just a string. The `YARN` type is immutable, so manipulations of a `YARN` variable always produce a new `YARN`.

Listing 6: `valid_and_invalid_strings.meow`

```
PSST These are valid strings
""
"hi, I'm a string"

PSST These are not valid strings
"Something seems incomplete here...
'I am not a string'
```

3.2.2 Integers and Floats

Numeric data types referred to as integers and floats in other languages correspond to the `NUMBR` and `NUMBAR` data types in `Meowlang`, respectively.

Integer literals are a sequence of one or more digits. Note that if your program provides an integer literal with leading zeros, such as `002`, this is read as the integer literal `2`.

Because `Meowlang` relies on the period character `"."` to terminate statements, float literals have one key difference from other language implementations: whereas other languages would consider `2.` a float declaration, `Meowlang` considers it an integer. Writing `2.` would otherwise be ambiguous and could mean either a float declaration without a termination or an integer with a termination character. If you want to declare a float, you must include a value after the decimal point, even if just a 0 (e.g., `2.0`). Note that if you choose to utilize more mathematical expressions using `e` or `E`, such as `1e-10`, the data type will be interpreted as a float.

3.2.3 Boolean

`Meowlang` supports boolean values with the `BOO` data type. Items of type `BOO` can be have either the value `AYE` (true) or `NAY` (false), which underneath the hood correspond to values of 1 and 0, respectively. Note that boolean values can neither be used in substitution of 1 and 0 and cannot be cast into integer values 1 and 0.

3.3 Variables

Variables are declared with the keyword phrase `ITZ ME` followed by the variable type and selected identifier. To create new identifiers, please refer to the identifier rules in section **3.1.2**. Variable identifiers must be unique within a scope. Below is a template variable declaration:

```
ITZ ME <type> <identifier>.
```

Note that is is possible to define and declare a variable at the same time. Defining a variable requires use of the keyword `IZ`, which acts like the assignment operator "`=`" in other languages.

```
ITZ ME <type> <identifier> IZ <value>.
```

When a variable is declared but not yet assigned a value, the value of a variable is a garbage value. Below, find examples of variable declarations and definitions for various built-in data types.

Listing 7: `definitions.meow`

```
1 ITZ ME NUMBR num IZ 2.
2 ITZ ME YARN random_string IZ "This is a string".
3 ITZ ME NUMBAR value IZ 2.0.
4 ITZ ME BOO fact IZ AYE.
5 ITZ ME BOO fiction IZ NAY.
```

Note that variables must be declared within a function or a class; **Meowlang** does not support global variables.

3.3.1 Type Casting

Type casting is supported among integers (`NUMBR`), floats (`NUMBAR`), and strings (`YARN`). This means you can convert freely between any of these three types. There is currently no support for casting more complex types (e.g., objects to strings). Note that casting a float to an integer truncates the value, such that it is rounded down to the nearest whole number.

Casting can be accomplished using assignment-like syntax using the keyword `IZ` as in assignment expressions as shown below, where the `<expression>` represents the value to cast (this could be an identiifer for another variable, for example), and `<identifier-type>` represents the type to cast to:

```
<identifier >IZ <identifier-type> <expression>
```

For example, the following code snippet converts a string into a float:

Listing 8: `string_to_float.meow`

```
1 ITZ ME NUMBAR value.
2 ITZ ME YARN str IZ "2.234".
3 value IZ NUMBAR str. PSST value is now 2.234 as a float
```

Important Note: When casting from integers and floats to strings, heap memory is allocated for the new string underneath the hood. This means that any time a string is defined using a cast from an integer or float it should (eventually) be followed by a call to the **Meowlang**-specific `free` function, `BLEEP`, in order to not leak memory. See more about the use of `BLEEP` in the sections below on classes.

```
BLEEP <string-identifier>
```

3.4 Arrays

3.4.1 Creating Arrays

An array is called `BUCKET` in **Meowlang**. **Meowlang** supports the creation of fixed-length arrays with size known at compile time and variable length arrays with a length is known at runtime. While **Meowlang**

does not provide "out-of-the-box" support for dynamically allocated arrays, it does provide the toolbox for programmers to implement such a feature on their own.

Each `BUCKET` can only hold elements of a single type; valid types include those listed in section **3.2 Built-In Data Types** as well as user-defined classes. To declare and/or create a new `BUCKET`, you have three options:

1. Declare and allocate heap memory for a new `BUCKET`, specifying the size of the `BUCKET` and the type of each element in the `BUCKET`, as well as elements within the array. Elements are inserted into the array in the order that they are provided (i.e., the first element after keyword `WIT` will be placed at index 0, the second at index 1, and so on...). While you cannot provide *more* elements than the number specified by the array size (and, in fact, the compiler will tell you if you exceeded the bounds of an array initialized with an integer literal as the size), you are allowed to initialize the array with only *some* elements specified.

   ```
   MAEK <identifier> NEW BUCKET OF <bucket_type> HOLDS <bucket_size_X>,
       WIT element1_value AN element2_value AN ... AN elementX_value.
   ```

2. Declare and allocate heap memory for a new `BUCKET` with a specified size, but no elements initialized. Be careful in using these arrays as the memory will be allocated, but the contents should be considered "garbage" until the user program sets the values explicitly.

   ```
   MAEK <identifier> NEW BUCKET OF <bucket_type> HOLDS <bucket_size>.
   ```

3. Declare a new `BUCKET` with both size and contents unspecified. The use of this option should be limited to scenarios where arrays are being returned from functions. Note that heap memory is **not** allocated in this case (hence, no use of `MAEK` keyword). This effectively creates a just a pointer to an array, without the actual memory for the array created.

   ```
   ITZ ME BUCKET OF <bucket_type> <identifier>.
   ```

`Meowlang` requires that users provide either a integer variable (with a value `>0`) or a integer literal (`>0`) for the array size (i.e., what is referred to as `<bucket_size>` in the templates above). In other words, the compiler will not accept any arbitrary expression. This decision was made for practical purposes; `Meowlang` is a verbose language and this rule is in place to ensure readability. Instead, users should create an integer variable, define it using whatever complex expression they require, and use that variable in the array declaration.

Listing 9: `array_declaration.meow`

```
1 PSST specify the elements on array creation
2 MAEK string_array NEW BUCKET OF YARN HOLDS 2,
3     WIT "item1"
4     AN "item2".
5
```

```
PSST do not specify elements
MAEK another_string_array NEW BUCKET OF YARN HOLDS 10.

PSST just create a pointer to an array
ITZ ME BUCKET OF YARN yet_another_string_array.
```

3.4.2 Accessing & Assigning Array Elements

Individual `BUCKET` elements may be accessed using standard bracket notation to access element in index `i`:

`<bucket_identifier>[i]`

Array indexing starts at index `0` and ends at `array_length-1`. You should not attempt to access elements beyond the size of the array or you will achieve a segmentation fault. Below, we create an array of exclamations and access the first element, which is the string "Lucky."

Reassigning elements within an array can be achieved with simple assignment statements, with array accesses on the left hand side. Of course, the element being assigned must be of the correct type for the array.

`<bucket_identifier>[i] IZ <expression>.`

Listing 10: `array_access.meow`

```
ITZ ME YARN str1 IZ "whoooopie!".
ITZ ME YARN str2 IZ "yiiippeee!".
ITZ ME YARN recovered_str.
ITZ ME NUMBR index IZ 0.

PSST Create simple array with elements that are variables
MAEK exclamations NEW BUCKET OF YARN HOLDS 3,
    WIT str1
    AN str2.

PSST the recovered string is "whoooopie!"
recovered_str IZ exclamations[index].
```

3.5 Basic Operations

`Meowlang` has support for a number of built-in operators that allow programmers to perform basic math, boolean expressions and comparisons out of the box. These operators rely on prefix notation, taking the following forms:

Unary operators:

```
<operator> <expression>
```

Binary operators:

```
<operator> <expression1> AN <expression2>
```

More specifically, `Meowlang` supports the basic operations outlined below.

3.5.1 Math

Basic math operators are provided as binary prefix operators. These operations are available for use on integers (`NUMBR`) and floats (`NUMBAR`) only. A combination of integers and floats may be used, in which case the operands will be treated two floats and yield a float as a result.

```
SUM OF X AN Y       PSST +
DIFF OF X AN Y      PSST -
PRODUKT OF X AN Y   PSST *
QUOSHUNT OF X AN Y  PSST /
BIGGR OF X AN Y     PSST max
SMALLR OF X AN Y    PSST min
```

Note that each `X` and `Y` below could be another math expression, such that the operators can be nested. As expected, multiplication and division have higher precedence than subtraction and addition. The use of a prefix operator also clearly denotes how an expression should be interpreted, without further parentheses, and operators are right-associative. This means that if you wanted to write an expression for `2 * 4 + 5`, you could write either of the following expressions:

```
SUM OF 5 AN PRODUKT OF 2 AN 4.
SUM OF PRODUKT OF 2 AN 4 AN 5.
```

3.5.2 Boolean

Boolean operators are limited to the following set of operators. Note here that `X` and `Y` must themselves be expressions that produce a boolean (`AYE`/`NAY`):

```
BOTH OF X AN Y      PSST X and Y
EITHER OF X AN Y    PSST X or Y
NOT X               PSST not X
```

3.5.3 Comparison

Comparisons can be performed using the following operators. Note that comparisons of two `NUMBR`s use integer comparison, and floating point comparison if one or both is a `NUMBAR`. This means that `Meowlang` considers `10` and `10.0` as equivalent and the expression `SAEM 10 AN 10.0` results in `AYE` (true).

```
SAEM X AN Y         PSST AYE (true) if x == y, else NAY (false)
DIFFRINT X AN Y     PSST AYE (true) if x != y, else NAY (false)
SMALLR X THAN Y     PSST AYE (true) if x < y, else NAY (false)
BIGGR X THAN Y      PSST AYE (true) if x > y, else NAY (false)
```

You cannot compare a string with an integer, float or boolean nor can you perform any of these operations with any custom object. However, `Meowlang` does provide support for string comparisons. More specifically, you may use `SAEM` and `DIFFRINT` with two strings to determine if the individual characters within the two strings match.

```
SAEM <string_identifier1> AN <string_identifier2>
```

3.5.4 Concatenation

`Meowlang` supports concatenation of two `YARN` elements or a `YARN` and a `NUMBR` or `NUMBAR`. At least one of the operands must be a `YARN`. To do so, use the `CAT` operator as follows:

```
cat_str IZ CAT str1 AN str2             PSST cat_str is str2 concatenated to str2
```

Note that the `CAT` operator creates a new `YARN`. `CAT` autocasts a `NUMBR` or `NUMBAR` to a `YARN` before concatenating the `NUMBR` or `NUMBAR` to a `YARN`. As stated above, strings created via `CAT` are allocated on the heap and must be freed using `BLEEP` when the program is finished with them.

Listing 11: `basic_operators_demo.meow`

```
1  PSST ~~ Math Examples ~~
2  SUM OF PRODUKT 3 AN 4 AN 5          PSST (3 * 4) + 5
3  BIGGR OF 15 AN QUOSHUNT 100 AN 10   PSST 15 > (100 / 10)
4
5  PSST ~~ Boolean Examples ~~
6  PSST (2 < 4) && (10 > 12) = false
7  BOTH OF SMALLR 2 THAN 4 AN BIGGR 10 THAN 12
8
9  PSST ~~ Concatenation Example ~~
10 PSST create string "1 fish 2.000000 fish red fish blue fish"
11 ITZ ME YARN catstr.
12 catstr IZ CAT CAT CAT CAT CAT 1 AN " fish " AN 2.0 AN " fish " AN "red fish "
       AN "blue fish".
```

3.6 Control Flow

3.6.1 If-Then-Else

Branching in `Meowlang` is accomplished by if-then-else statements, as in other languages. The template for a set of if-else conditions is as follows:

```
<expression>            PSST Conditional expression
O RLY?                  PSST If keyword
YA RLY HAI              PSST Then keyword and opening bracket to begin code block
     <code block>       PSST Jump here if expression is true
```

```
KBYE                    PSST Closing bracket on code block
NO WAI HAI              PSST Else keyword and opening bracket to begin code block
     <code block>       PSST Jump here if expression is not true
KBYE                    PSST Closing bracket on code block
```

Note that it is possible to omit the `ELSE` if you only need an `IF` condition:

```
<expression>
O RLY?
YA RLY HAI
    <code block>
KBYE
```

The following code snippet illustrates usage of the if-then-else construct in `Meowlang`. Note carefully that there is no period (".") after the comparison expression and that although the construct spans multiple lines, the whitespace is not required.

Listing 12: `if_else.meow`

```
1  HAI ITZ ME FUNC Conditions_Example,
2          PSST Test if-then-else
3          ITZ ME YARN condition.
4
5          PSST IF THEN
6          SMALLR 4 THAN 10
7          O RLY?
8          YA RLY HAI
9              condition IZ "math is the only Truth".
10         KBYE
11
12         PSST IF THEN ELSE
13         SAEM 4 AN 4
14         O RLY?
15         YA RLY HAI
16             condition IZ "same same".
17         KBYE
18         NO WAI HAI
19             condition IZ "not same same".
20         KBYE
21
22         PSST IF THEN ELSE with code blocks
23         SAEM "kittens" AN "puppies"
24         O RLY?
```

```
25          YA RLY HAI
26              condition IZ "this is not the correct answer. ".
27              condition IZ CAT condition AN "pick a team!".
28          KBYE
29          NO WAI HAI
30              condition IZ "clearly kittens are cuter".
31              condition IZ CAT condition AN "or are they?".
32          KBYE
33
34          GIVE condition.
35  KBYE
```

3.6.2 For Loops

The only count-controlled loop in `Meowlang` is the for-loop. The loop may either increment an index variable value (using the `UPPIN` keyword) or decrement an index variable value (using the `NERFIN` keyword) with each iteration. A general template for creating a loop is provided below for the increment case (replace `UPPIN` with `NERFIN` for the decrement case):

```
IM IN YR LOOP <index var identifier> UPPIN <index var assignment (optional)>
    AN <termination condition>
HAI
    <code block>
KBYE
```

Note that you may initialize the value of the index variable in the loop, as is customary in many languages. For example, a common programming pattern is to set an index variable to `0` at the beginning of a loop. However, the index variable assignment can be omitted when you have previously declared the index variable you want to use, as in the first two loops in the example below.

Listing 13: `for_loops.meow`

```
1   HAI ITZ ME FUNC Loops_Test,
2
3           ITZ ME NUMBR count.
4           ITZ ME NUMBR index.
5           ITZ ME YARN condition.
6           count IZ 0.
7
8           PSST Incrementing with initialized index
9           IM IN YR LOOP count UPPIN AN SMALLR count THAN 10 HAI
10              condition IZ "count is still not 10".
11          KBYE
```

```
12
13          PSST Decrementing with initialized index
14          count IZ 20.
15          IM IN YR LOOP count NERFIN AN BIGGR count THAN 10 HAI
16              condition IZ "count is still more than 10".
17              condition IZ CAT condition AN "count is".
18              condition IZ CAT condition AN count.
19          KBYE
20
21          PSST Initializing index and then incrementing
22          IM IN YR LOOP index NERFIN index IZ 15 AN BIGGR index THAN 10 HAI
23              condition IZ "index is still more than 10".
24          KBYE
25  KBYE
```

3.7 Functions

Functions in `Meowlang` have a fixed number of zero or more arguments and can return at most one value. The types of arguments and return types must be specified in the function definition. A new function can be defined using the following syntax, which makes use of the `HAI` and `KBYE` scoping keywords. The list of parameters uses the `WIT`.. `AN` construct, as previously seen in array declarations, with the comma to indicate completion of the parameter list. Note that the number of arguments is may extend indefinitely, though it is recommended to keep the number of arguments to no more than four for readability:

```
HAI ITZ ME <return_type> FUNC <function_identifier>
    WIT <arg1_type> <arg1_identifier> AN <arg2_type> <arg2_identifier> ...
        AN <argX_type> <argX_identifier>,
    <local-variable-declarations>
    <other-statements>
KBYE
```

Note that variable declarations must come first in a function; you may redefine the variable anywhere elsewhere.

To return a value from a function, use the keyword `GIVE` and any expression of the correct return type:

```
GIVE <expression>.
```

Below, we define a new function called ‘Chase‘ that takes two `YARN` arguments and returns another `YARN` representing the concatenation of the two provided strings with “chases”.

Listing 14: `chase.meow`

```
1  # PSST return values and arguments are optional
```

```
HAI ITZ ME FUNC Do_Nothing,
    PSST This function does nothing
KBYE

HAI ITZ ME YARN FUNC Chase WIT YARN bad_cat AN YARN poor_mouse,
    GIVE CAT CAT bad_cat AN " chases " AN poor_mouse.
KBYE
```

Once a function is defined, it can be called using the keyword `PURR` in the following way, which again makes use of the `WIT`.. `AN` .. pattern:

```
PURR <function_identifier> WIT <arg1> AN <arg2> AN ... AN <argX>.
```

The code sample below calls the `Chase` function defined in **Listing 9**. Note that we capture the returned string in a new variable of type `YARN`:

Listing 15: `chase_call.meow`

```
PSST This function call will return "Silvester chases Gary"
ITZ ME YARN message IZ
    PURR Chase WIT "Silvester" AN "Gary".
```

3.7.1 Built-In Functions

Out of the box, `Meowlang` comes with several "built-in" functions that allow for basic I/O operations such as printing (`Meow`) and retrieving input from users (`Scan`).

- `Meow`: A void function that expects a single argument of any basic type (string, integer, boolean, or float type) and prints it to stdout. It does not yet support printing arrays or classes.
- `Scan`: Reads user input from stdin as a `YARN` (string). It takes a single argument, an identifier for a `YARN` to hold the user input, and returns the number of characters read. Because the `Scan` function allocates heap memory behind the scenes to retrieve and store user input, a memory-conscious programmer should always use `Meowlang`'s memory deallocation keyword `BLEEP` when the `YARN` variable is not longer needed (you can read more about other scenarios where `BLEEP` is used under the **Object Oriented Programming** section below).

Examples of how to use these built-in I/O functions are provided below. While `Scan` always reads the user input initially as a `YARN`, it is very straightforward to convert the input into a `NUMBR` or `NUMBAR` for further use in the program using casting.

Listing 16: `favorite_color.meow`

```
HAI ITZ ME FUNC Main,

        ITZ ME YARN user_message.    PSST String to hold read content
```

```
4
5           PURR Meow WIT "Please tell me your favorite color: ".
6           PURR Scan WIT user_message.  PSST Reading in content
7
8           PURR Meow WIT "Your favorite color is: ".
9           PURR Meow WIT user_message.
10
11          BLEEP user_message.
12  KBYE
```

3.8 Import Modules

`Meowlang` allows programmers to import functions and classes from `.meow` files and use them in another `Meowlang` source file. The syntax for the import statement uses the keyword `GIMME` and character "`?`" as in the following pattern:

```
GIMME <MODULE_NAME>?
```

`Meowlang` has a funny convention regarding module names. Note that the `MODULE_NAME` that should be used for the import is actually the name of the `.meow` source file, but transformed into uppercase and without the `.meow` suffix. This means that if you defined a source file `hello_world.meow` it becomes importable into another '`.meow`' file as `HELLO_WORLD`. It is possible to import more than one module; each module to be imported must have its own `GIMME<module_name>?` line. Imported modules may also contain imports and it is OK for multiple imported files to also import the same file. There is no import hierarchy in `Meowlang`, meaning that if `example.meow` imports `colors.meow` imports `blue.meow`, `example.meow` will have access to the contents of `blue.meow` even without directly importing it.

For now, the `Meowlang` compiler only searches for imported files in the same directory as the file attempting the import. Note that importing functions and classes containing an identifier already in use will result in a compiler error; there cannot be duplicate identifiers. It is also important to remember that there must be one and only one `Main` function within a program. Convention also dictates that a module that is imported contains "helper" or "accessory" functions and classes and should not contain the `Main` function anyway.

3.9 Object Oriented Programming

Unlike the original LOLCODE, which does not have support for custom objects, `Meowlang` allows programmers to define their own classes with instance variables and methods. Classes are composed of class variables and class functions. Inheritance and interfaces are not supported in this version of `Meowlang`.

3.9.1 Creating A New Class

You can define a new user-defined class using the `HAI ITZ ME ... KBYE` set of keywords, as seen in creating new functions. Note that here, you must specify that you are creating a new class with the `CLASS` keyword.

As previously mentioned, the `Meowlang` convention is for class name to be given in all capital letters (see section **3.1.2**); when an instance of a class is created, only the first character of its identifier is capitalized.

Classes in `Meowlang` consist of zero or more class variables and zero or more methods. Class variables can be declared like any other variable and `Meowlang` supports the option of setting a default value for each variable using declaration with assignment. Methods within classes follow the same syntax as a normal function. Methods may access a class variable directly, with on special additional syntax. For example, if the `Mouse` class has a `cookies` instance variable, any method within the `Mouse` class may refer to `cookies` directly.

Below is a generalized template for class creation; the ellipses (`...`) specifies that there can be an indefinite number of instance variables, statements, and methods for a given class.

```
HAI ITZ ME CLASS <class_name>,
    ITZ ME <type> <instance_variable1>.
    ITZ ME <type> <instance_variable2> IZ <default-value>.
        ...
    HAI ITZ ME <return_type> FUNC <method_name1>
            WIT <type1> <arg1> AN <type2> <arg2>,
        <statements>
        ...
    KBYE
    ...
KBYE
```

Below, we create a new custom `MOUSE` class, which has a single instance variable: `cookies` of type `NUMBR`. The class also has three methods: `squeek`, which simply prints out the sound that a mouse makes, `count_cookies` which returns the number of cookies that the instance has stored in the `cookies` instance variable, and `give_cookie`, which takes a `NUMBR` representing the number of cookies to increment `give_cookie` by.

Listing 17: `mouse_class.meow`

```
1  PSST Create a custom MOUSE class, class convention is in all-caps
2  HAI ITZ ME CLASS MOUSE,
3      PSST This is a class variable with default value
4      ITZ ME NUMBR cookies IZ 0.
5
6      PSST This is a class variable with no default value
7      ITZ ME NUMBR cake.
8
9      HAI ITZ ME NUMBR FUNC Count_Cookies,
10         GIVE cookies.
11     KBYE
12
```

```
    PSST This is a class method
    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This uses the SUM prefix operator
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE
```

3.9.2 Create/Remove New Class Instance

In order to create a new instance of a class, you must use the `MAEK` and `NEW` keywords, which allow memory to be allocated on the heap for the new object. `Meowlang` supports class constructor functionality by allowing programmers to optionally specify the value of one or more instance variable values on object creation. To accept the default values for the instance variables (if there are any), simply use this pattern:

```
MAEK <object_identifier> NEW <class_name>.
```

Otherwise, you may specify a value for one or more instance variables using the `WIT ... IZ ... AN` pattern outlined, which will override the default values specified by the class declaration. Note that setting only a subset of the instance variables is acceptable:

```
MAEK <object_identifier> NEW <class_name>,
    WIT <class_variable> IZ <value>
    AN <class_variable> IZ <value>.
```

To illustrate this functionality, we create a new instance of the `MOUSE` class named `Gus` that utilizes the default constructor value for instance variable `cookies` and another instance named `Jerry` that has `cookies` specified as 1.

Note that because `Meowlang` does not support automatic garbage collection, the memory associated with the object must be freed using the `BLEEP` keyword once there is no use for it anymore:

```
BLEEP <object_identifer>.
```

Listing 18: `ephemeral_mice.meow`

```
MAEK Gus NEW MOUSE.  PSST Using default values for class vars
BLEEP Gus.           PSST Release memory associated with gus

MAEK JERRY NEW MOUSE,
    WIT cookies IZ 1. PSST using constructor to specify cookies value

BLEEP Jerry.
```

3.9.3 Accessing Class Variables/Methods

Class variables may be accessed with the `IN` keyword. The value of the class variables can be re-assigned at anytime with the `IZ` keyword. See the templates for access and reassignment below, respectively.

```
<instance_variable> IN <object>.
<instance_variable> IN <object> IZ <new_value>.
```

In order to use class methods, you must not only use the same `PURR` keyword used to call a function and provide all required arguments, but you must also specify a specific object instance using the `IN` keyword. Here is the general form of a method call:

```
PURR <method_identifier> IN <object_identifier>
    WIT <arg1> AN <arg2> AN ... AN <argX>.
```

Note that to facilitate coding with objects and arrays, you may also call class methods directly from objects within arrays, if you'd like. This is featured in the code example below.

```
PURR <method_identifier> IN <array_of_objects_identifier>[<index_of_object>].
```

Listing 19: `working_with_classes.meow`

```
1  PSST How to access instance variables
2  PURR Meow WIT cookies IN Jerry.  PSST print Jerry's cookie value
3  cookies IN Jerry IZ 2.           PSST Set Jerry's cookie value
4
5  PSST How to use class methods
6  PURR Count_Cookies IN Jerry.
7
8  PSST Make an array that contains Jerry
9  MAEK my_pets NEW BUCKET OF PET HOLDS 1.
10 my_pets[0] IZ Jerry.
11
12 PSST Call Jerry's Give_Cookie() Method
13 PURR Give_Cookie IN my_pets[0] WIT 2.
```

There is one other special case to consider. If a method should call another method *within the same object*, you must utilize the `HERE` keyword in place of the object identifier in order to specify that the object being referred to is itself.

```
PURR <method_identifier> IN HERE.
```

3.10 Writing a Complete Program

The `Meowlang` compiler expects the programmer to define a single `Main` function that is the first function that gets executed when a program starts running. This `Main` function is declared like any other function

and must be named “Main” exactly.

As mentioned in the preceeding sections, a **Meowlang** source file may contain imports, functions and custom classes. It is not necessary for a single source file to contain all three. Imports must occur at the top of the file, but functions and classes may be in any order.

Below is a example **Meowlang** program made up of a single function, **Main**, and a print statement. This is sufficient for a full **Meowlang** program!

Listing 20: **example_main.meow**

```
1 HAI ITZ ME FUNC Main,
2     PURR Meow WIT "Hello World!".
3 KBYE
```

4 Project Plan

4.1 Process

4.1.1 Communication and Meetings

From the beginning of the project, our team committed to weekly team meetings and responsive communication over Slack to ensure that the project went smoothly. After our team initially formed in late January, we set up recurring team meetings on Tuesdays at 6pm over Zoom. Once we were assigned a TA for the project, we merged our TA meeting with this weekly meeting to ensure that everyone is able to participate. At several times during the semester, we also utilized an extra meeting block on Saturdays at 10am. Toward the end of the project, as work was divvied up by feature (as outlined below), our team meetings were structured similarly to a typical software team "stand up," with each member reporting on their completed action items, blocking questions, and future work for the week, in a round-robin fashion. After each team meeting, action items, important question, next meeting dates and any helpful were posted in Slack.

Outside of team meetings, a `Meowlang`-specific Slack Workspace served as our primary means of communication and well as adhoc peer-programming sessions where we worked together in groups of two or three on specific action items.

4.1.2 Planning & Development

Led by Carolyn, who organized the team and would later become our language guru, we established at the very beginning of the project that we wanted to develop an entertaining language based loosely of off LOLCODE. Over the next few weeks, we settled on making `Meowlang` a more general-purpose programming language, which guided our decision to support many generic features of modern programming languages (i.e., functions, arrays, for-loops, if-statements, basic arithmetic, comparison expressions, and classes).

Once we settled on the key features for the language, we moved on to working on the Language Reference Manual (LRM) alongside the development of the scanner, parser and abstract syntax tree simultaneously. We started by introducing scanning and parsing for simple expressions and then functions; testing manually that our code was working as expected. As we finalized the LRM in late February, we also incorporated parsing support for classes and arrays, making further adjustments to both our code and LRM as necessary.

With the parser and scanner complete, our attention turned in early March to setting up a framework for our regression test suite and introducing semantic checking, generation of the semantically-checked AST, and code generation to the compiler. Rather than write all of our semantic-checking code before starting on code generation, we split the rest of the development by language feature. Once again, we started by completing work for functions and worked our way up to classes and arrays. See more about our workflow in **4.4 Roles and Responsibilities** below.

4.1.3 Collaboration

Once the coding component of our project began, the team used Git and Github for version control and collaboration. When working in small teams on a feature, we also utilized the VS Code Live Share Extension Pack to work jointly in VS Code over Zoom in a Google-Docs-like interface. When implementing a new feature for the language, team members would checkout a new feature branch for development and merge into our mainline branch once all regression tests passed (see **Testing** section below).

4.2 Programming Style Guide (For Compiler)

Because the `Meowlang` was built in OCaml, our team tried to follow the generic OCaml Programming Guidelines as much as possible. The full list of recommended style guidelines can be found here: `https://ocaml.org/learn/tutorials/guidelines.html`.
More specifically, below are a few key style guidelines that we followed strictly, as well as a few that we created and maintained on our own.

- Be generous but judicious with use of spaces and tabs; this generally improves the readability of the code.
- Be generous with comments, not just to explain "what" but also "why"! Even if the code makes sense now, someone else will have to be able to understand your logic down the line (including you!).
- If you copy and pasted a piece of code from one place to another, it is a sign that it should probably be a new helper function. If the function is called in multiple locations, the a helper function should be specified as a `let` expression at the top of the source file.
- Lines should not exceed 100 characters. If a line is getting too long, rewrite it as multiple expressions or simply create a new line at a natural break in the code.
- Avoiding writing complex expressions as arguments to functions; instead, create a new `let...in` expression and use that expression in the function call.

Given that the OCaml is not sensitive to whitespace or tabs, we also tried to stick with the following indentation rules:

- Indents should be exactly 2 spaces and should be used primarily to clarify the code structure and improve readability. Only add excessive indentation where readability justifies it.
- When using pattern matching, the body should be left-justified, though indentation when a match statement is given in parenthesis is OK. Because pattern matching is so abundant in the compiler, here is an illustration of our indentation conventions:

```
let do_something = function
  X as x -> [make_simple_x x]
| Y as y -> [make_simple_y y]
| Z :: zz ->
```

```
let simp_z = make_simple_z z
in z :: zz
```

- In the case of (a) expressions following the definition of a `let` or (b) a series of `let` definitions, all expressions are indented to the same level.

- When using a `let...in` expression, if the `let` expression body uses a single line, keep the keyword `in` on the same line. In the case when the `let` expression body takes up multiple lines, use discretion in deciding where to place the keyword `in`; if placing `in` alone on the next line makes the code more readable, then do that.

  Because `let` expressions are omnipresent in the compiler, here is an explicit example of acceptable whitespace style for our complier:

```
let e1 = ... in
let e2 = ... in
let new_e1 =
  let e1' = do_expression e1
  and e2' = do_expression e2 in
  build_new_e e1' e2'
in
new_computation (new_e1, e2)
```

4.3 Project Timeline

Below is a recreated project timeline based on the weekly meetings notes from our primary Slack channel:

Date	Goal
1/15/21	Set up group Slack Channel for communications
1/23/21	First Team Meeting / Decision on Language Concept
1/30/21	Continue brainstorming language features
2/9/21	Setup Github Repo and Access; Begin work on LRM
2/13/21	Begin work on Scanner/Parser for language
2/17/21	Complete Basic Functional Component of Scanner/Parser/AST
2/23/21	Complete Class Component of Scanner/Parser/AST, Pretty Printing
2/24/21	Submit Language Reference Manual
3/01/16	Spring Break
3/12/16	Begin work on semantic checks for functions, setup basic regression test suite, start on imports
3/17/16	Begin semantic checks for classes and make revisions to Scanner/AST as needed
3/18/16	Start on basic code generation for "Hello, World"
3/19/16	"Hello, World" working! Generation of test programs to test semantic checks continues
3/23/16	Begin on codegen for basic operations, functions, if/for loops with corresponding tests
3/27/16	Begin on codegen for classes, complete function codegen with corresponding tests
4/03/21	Finish CI Setup With Travis to allow regression tests to run automatically
4/06/21	Codegen begins for arrays, built-in scanf functionality, imports work continues
4/13/21	Tests for binary ops, arrays and custom objects completed, casting and concatentation implemented
4/17/21	Code Freeze (Complete Programming Component); Work on documentation and presentation begins
4/25/21	Group Presentation, Documentation completion and submission

4.4 Roles and Responsibilities

Each team member took on the following "roles" during the project:

- **Language Guru**: Carolyn

  The concept for the language was Carolyn's brainchild, so she naturally took on the role of Language Guru. This role required significant amount of frontend work to develop what the language should look like and the functionality that the language should support.

- **Manager**: Meg

  Utilizing project management skills obtained at her previous job, Meg took on the role of team Manager. This involved setting deadlines, organizing meetings, providing summary/updates in Slack of deliverables, and setting objectives for each week.

- **System Architect**: Lauren

Lauren took on the role of System Architect. A major component of this role included decision making on how to handle imports, supported by Carolyn. Additionally, Lauren took on a significant component of the codegen work to support custom language features like string concatenation.

- **Testing**: Michelle

  Michelle led the effort in testing from the beginning of the project, working with Meg to generate a regression test suite and testing framework by mid-March. Michelle also identified key semantic checks and developed tens of test programs that allowed us to evaluate how our compiler handled each unique situation.

Despite these specific role designations, members of the team participated in all components of the compiler, from the scanner and parser, to codegen and testing. When coding on the compiler began, we decided to break down work into components of the `Meowlang` language, rather than components of the compiler. This meant that instead of assigning one person to work on the scanner and parser, another to work on semantics and another to work on codegen, work was assigned in units of language features that spanned all aspects of the compiler. For example, after the scanner and parser were completed, each person was assigned one or more language features with the expectation to complete the following:

1. Complete semantic checks and ensure parsing was accurate using our pretty printer.
2. Write test programs to test all semantic checks (both failing and testing cases).
3. Complete code generation for the feature.
4. Test full pipeline first manually, then by generating test programs and saving their output for the regression test suite.
5. Run full regression test suite to prevent accidentally pushing breaking changes.
6. Merge in new feature!

This scheme worked well, as it allowed work to be divvied up into reasonable chunks and for someone to "own" each feature. Where necessary, multiple members of the team worked together on a specific feature. Between the scanner/parser and semantics/codegen steps, there were some swapping of feature-assignments so that we all understood how the compiler worked for each component. Here is the break down of who took on each language features and their contribution to the feature:

Language Feature	Scanner/Parser	Semantics/Codegen	Testing
Binary Operations	Meg, Michelle	Meg, Michelle	Meg, Michelle
Unary Operations	Meg, Michelle	Meg	Meg, Michelle
Arrays	Meg	Meg	Meg
Functions	Meg, Michelle	Meg	All
If Statements	Lauren	Michelle	Michelle
For Loops	Lauren	Lauren	Lauren
Classes	Carolyn	Meg	Meg, Michelle
Imports	Meg, Carolyn, Lauren	Meg, Carolyn, Lauren	Carolyn
Casting	Meg, Michelle	Meg, Lauren	Meg, Lauren
I/O (Custom Scanf)	Meg	Meg	Meg
Custom Concatentation	Lauren	Lauren	Lauren
Report/Presentation	All	All	All

4.5 Software Development Environment

Throughout the project, all team members developed the compiler on various versions of Mac OS X. However we tried to keep the versions of software required to run the compiler consistent. By utilizing TravisCI for testing purposes (see below), we were also able to confirm that our compiler could compile and run `Meowlang` programs on a standardized Ubuntu 16.04.6 LTS operating system and that it didn't just work on one person's machine. The following list represented the software and versions installed for our local environments:

- Ocaml (version 4.11.1)
- LLVM (version 8.0.1)
- Opam (version 2.0.7)
- gcc (clang version 12.0.0)

Development of the compiler was done using VSCode with the OCaml and LiveShare extensions, with Git for version control. Our codebase lives at `https://github.com/mmfrenkel/meowlang`. All reports were written using LaTeX.

4.6 Project Log

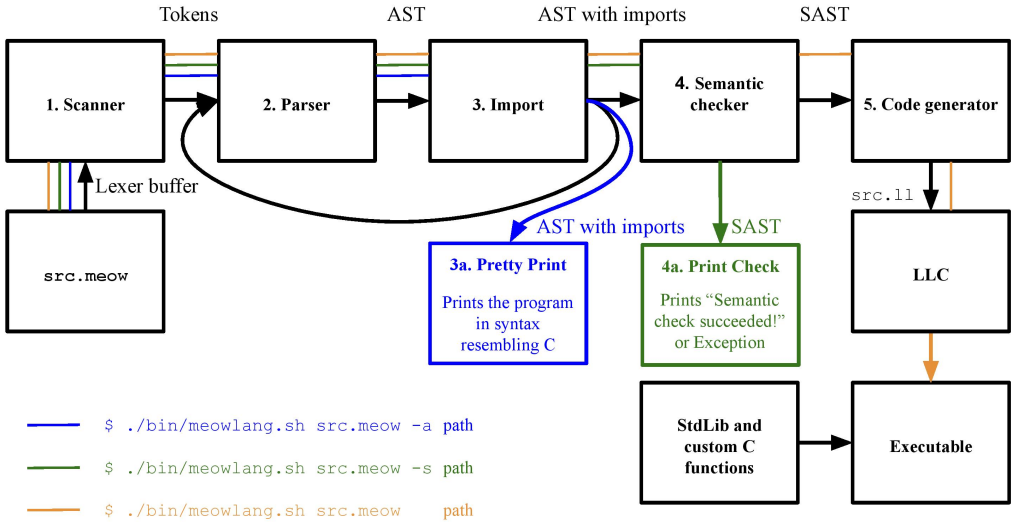
Our project log includes 242 commits. Our full git history can be found in **Appendix, 3.4 Git History**.

5 Architectural Design

The main program that is used to run the compiler is the `bin/meowlang.sh` script. This script supports several different "paths" of execution that allow a user to specify a source file path and optionally a flag to choose whether to pretty-print the abstract syntax tree (`-a`, blue path), semantically check their program (`-s`, green path), generate llvm code (`-c`) or complete the full pipeline and execute their program (no flag specified, orange path). Note that this script should only be used after running `make` to build the compiler, creating the `meowlang.native` executable.

```
$ ./bin/meowlang.sh [source-file] [-a|-s|-c]
```

The flow of compiling a `Meowlang` program is as follows:



1. A `.meow` file is opened and streamed into a lexer buffer. The lexer buffer contents are passed to the scanner, which tokenizes the lexemes.
2. The tokens are passed to the parser, which utilizes an unambiguous grammar to return an abstract syntax tree representation of the program.
3. The abstract syntax tree (AST) is passed to the import module which recursively opens other source `.meow` files and their imports to ultimately returns a new abstract syntax tree with imported classes and functions merged in.
4. The AST with imports is passed to the semantic checker, with returns a semantically-checked abstract syntax tree (SAST).
5. The SAST is passed to the code generator, which generates LLVM code and stores it in a `.ll` file.
6. The `.ll` file is linked with compiled C source files containing custom functions to create a `.exe` file that can be run.

5.1 Scanner

Files: `scanner.mll`

The scanner takes in a lexer buffer of the program and tokenizes it into literals and keywords reserved for specifying structure and scope (like braces), module imports, objects and classes, arrays, functions, binary operators, flow control, and identifiers for variables and for data types (since `Meowlang` is a statically typed language). Whitespace and comments are discarded. The scanner raises a `SyntaxError` if any string is not terminated at EOF.

The scanner was implemented by all members of the team.

5.2 Parser

Files: `ast.ml`, `parser.mly`, `pretty.ml`

The `Meowlang` parser is implemented using ocamlyacc. The parser receives the tokenized program from the scanner and uses our context-free grammar to parse the tokens into types of import, declarations, statements, and expressions, accounting for precedence and associativity. When a token or a set of tokens is matched with a pattern, an expression that matches a type listed in `ast.ml` is returned with arguments (if they exist). The parser thus returns a syntactically checked abstract syntax tree (AST), or returns a parsing error if it encounters a sequence of tokens not accepted by the grammar.

The parser was implemented by all members of the team.

5.3 Imports

Files: `ast.ml`, `import.ml`, `sast.ml`

The AST returned by the parser is passed to the `Import.ml` module, which adds imports recursively to the main program's AST. Beginning with the imports of the main source file, the program opens each import and calls upon the scanner and parser to produce the AST for that source file. It then looks at imports within that source file, and so on. As it recursively traverses the imports, the program populates a hash table of `.meow` source files imported, with the module's file path as the key and the AST of the file contents as the value. This approach allows circular imports to be easily identified; more specifically, the recursion base-case is provided when either (a) a module has already been imported or (b) the module has no imports. Note that as each import is processed, the import module checks to ensure that the module naming convention is correct and that the imported module exists as a file in the directory of the main source file. After the hash table is populated, all new classes and functions for each import in the hash table are appended to the classes and functions in the original AST. The import list for the new AST is an empty list, since all imports have been processed.

The import parser and semantic checker was implemented by Carolyn, Meg, and Lauren.

5.4 Semantic checker

Files: `semant.ml`, `sast.ml`, `exceptions.ml`

The semantic checker receives an AST and returns a semantically-checked abstract syntax tree (SAST) matching types specified in `sast.ml` and using exception types and messages specified in `exceptions.ml`. The semantic checker uses hash tables to store functions, classes, and variables, checks that the program has exactly one `Main` function, and checks for duplicate identifiers for any type, among many other expression-specific and statement-specific type checks. Along the way, it ensures that arguments and operands match the data type expected for all expressions, function calls, method calls, and assignment statements. We define a new `environment` type that is passed through each recursive check to keep track of the "context" (i.e., Are we in a class or function? What are the local symbols in scope? Are we in a constructor?). Because the semantic checks were often heavily dependent on the context, this approach was very helpful. See `semant.ml` to see all semantic checks.

Note that the much of the "heavy lifting" for handling classes and methods within the compiler occurs as part of the AST-to-SAST transformation. As part of the processing, all class methods are converted into top-level functions that have one additional argument, the object that the method is being called on. The name of the "new" function is the class name concatenated with the name of the method, separated by a period. For example, a class named `MOUSE` with method `Get_Num_Cookies` would be converted into a function called `MOUSE.Get_Num_Cookies` that takes a `MOUSE` instance as an argument. We also make call-site adjustments where we replace all instances of a method call with the "new" function call, passing the relevant object as an argument.

The semantic checker was implemented by all members of the team.

5.5 Code generator

Files: `codegen.ml`, `lib/custom_casting.c`, `lib/custom_scanf.c`, `lib/custom_strcat.c`, `lib/custom_strcmp.c`

The code generator receives the SAST and generates LLVM intermediate representation (IR) using `LLVM.MOE`, an OCaml API. LLVM IR uses static-single assignment, where there is an unbounded number of registers and each is assigned only once. At the point that the main `translate` function is called, all the class methods and method calls from the original source files have been converted to functions and function calls (see SAST transformation step). This means that there are only two important things that the code generator take care of: (1) for each user-defined class, it generates a new struct type that contains all class instance variables and (2) for each user-defined function in the SAST, build each function. For each function, the code generator (a) allocates stack space for local variables and formal arguments to functions, (b) allocates heap memory for objects and arrays and initializes their contents, as necessary, (c) builds expressions and calls other functions. Note that here we also define the prototypes for several built-in C functions that we later link in, as well as three helper custom C functions found in `src/lib`: `custom_casting.c` and `custom_strcmp.c` and

`custom_strcat.c`.

The code generator was implemented by all members of the team. Custom C functions were written by Lauren and Meg.

6 Test Plan

6.1 Testing Design & Automation

6.1.1 Test Suite

There were three main components in the test suite:

1. test programs
2. shell scripts
3. test outputs

Test programs, found in the `/test/test_programs` directory, consisted of `.meow` files which either had the prefix `fail_` or the prefix `test_`. The files labeled with `fail_` were tests that were expected to output an error message, while the `.meow` files with `test_` prefixes were expected to pass. The `test_program` directory also contains demo programs such as `cat_adventure.meow`, though these are ignored as part of the regression test suite outlined below (generally, because they rely on user-input).

Shell scripts were created for testing automation. Because the `Meowlang` compiler supports (a) pretty-printing the AST, (b) performing semantic checks, and (c) running full pipeline compilation and program execution, we were able to create a regression test suite that tests the AST, semantic checker, and code generation individually in order to better isolate and identify issues in the compiler. For example, if a program passes the AST test, but fails the semantic checker test, we know we introduced a bug in our semantic checker. The script `test_all.sh` runs all of the tests in the regression suite and can be run from the root of the `Meowlang` repository as shown below. More details on how to run the shell scripts can be seen in the `README.md` or on the Github repository for Meowlang.

```
$ ./test/test_all.sh
```

Under the hood, this script compares the output of the compiler with "expected outputs" saved under the `test/test_output/ast`, `test/test_output/semantic` and `test/test_output/full_pipeline` directories. For every source file included in the regression test suite, there is a corresponding file in each of these directories with the suffix `.out`. These `.out` files are then "diff'd" with the actual output when the regression tests run and the program stops and reports the first issue encountered, if there are any.

Our test suite contains 158 tests. The expected output, when all tests pass, is provided in **Appendix, 8.3 Test Output**.

6.1.2 Testing Process

The regression test suite was created in parallel with the coding of semantic checks in `semant.ml`. This allowed us to build test programs for each new semantic check added to ensure that our checks were working as expected.

In order to ensure that our semantic checks wee complete, our team spent time before we started any coding simply brainstorming and developing a long list of the types of checks that we wanted to complete and that made sense for our programming language. For example, we identified the following series of checks for functions:

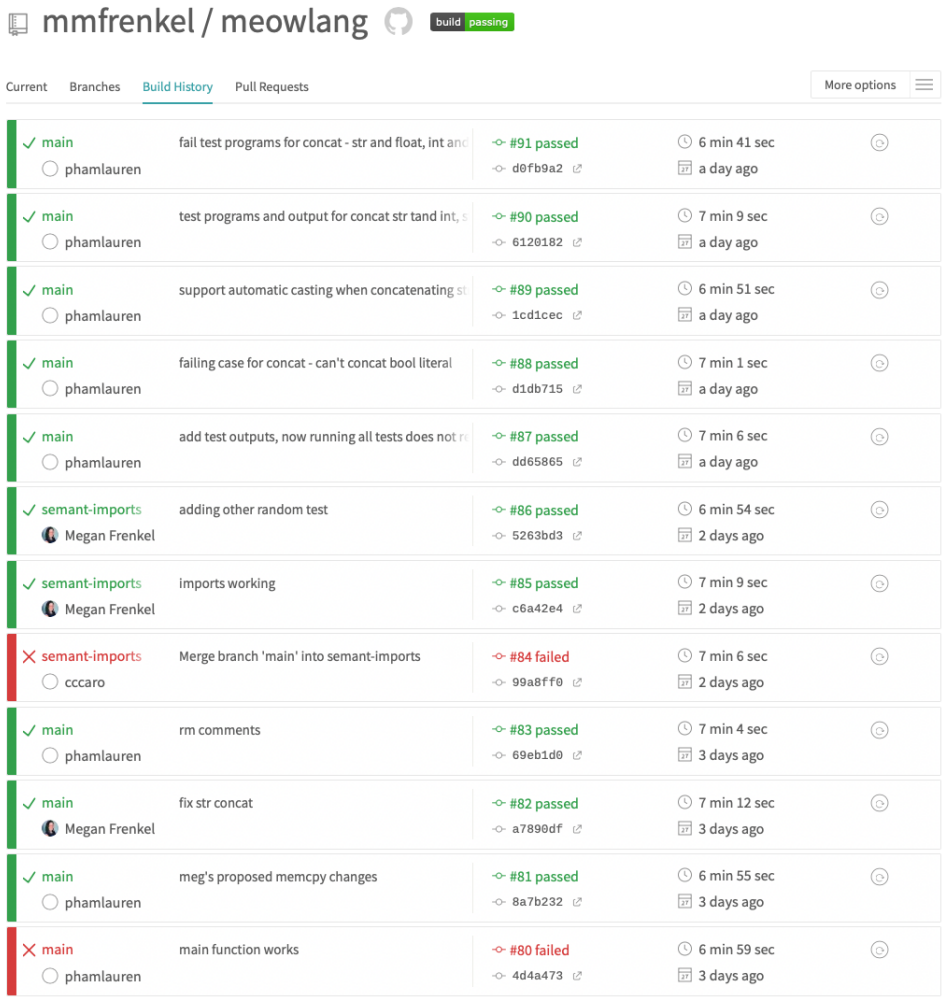
1. Make sure no duplicate function names exist.
2. If a function is called, make sure that the function exists.
3. Check that all arguments required for a function are provided in a function call.
4. Ensure that variable types passed as arguments on function call match types of formal arguments.
5. Check that no user-defined functions clash with built-in functions in language (`Meow` and `Scan`).
6. Make sure that functions with no return type specified do not include any return statements.
7. Make sure that functions that return always return an item of the expected type.

With a long list of semantic checks already in mind, we were able to quickly develop the semantic checker and relevant test programs.

6.1.3 Continuous Integration with TravisCI

In order to ensure that code pushed to our repository (a) passed all regression tests and (b) worked on more than one person's computer, we set up continuous integration with TravisCI. We configured TravisCI to run our `test/test_all.sh` script each time a commit was made to any branch. This way, each person knew when and why our test suite started failing and whose commit was responsible.

Below is a screenshot of our TravisCI console for the project, demonstrating how breaking changes were caught by TravisCI and fixed accordingly.



6.2 Example Source Language Programs & Target Output

Below are four sample programs, with their target LLVM output.

1. The following program, `test_mouse_class.meow`, creates a user-defined class named `MOUSE` and includes a main function that creates a new instance of `MOUSE`, accesses instance variables and calls a class method.

Listing 21: `test/test_programs/test_mouse_class.meow`

```
1 HAI ITZ ME FUNC Main,
2
3     ITZ ME NUMBR jerrys_cookies.
4     MAEK Jerry NEW MOUSE.     PSST no specification of cookies
5
6     jerrys_cookies IZ cookies IN Jerry.
```

```
    cookies IN Jerry IZ 2.

    PURR Give_Cookie IN Jerry WIT 2.
    jerrys_cookies IZ PURR Count_Cookies IN Jerry.

    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 0.

    HAI ITZ ME NUMBR FUNC Count_Cookies,
        GIVE cookies.
    KBYE

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This uses the SUM prefix operator
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE
```

Listing 22: target_output/test_mouse_class_output.ll

```
; ModuleID = 'Meowlang'
source_filename = "Meowlang"

%MOUSE = type { i32 }

define void @MOUSE.Give_Cookie(%MOUSE* %"mouse*", i32 %count_cookies) {
entry:
  %"mouse*1" = alloca %MOUSE*
  store %MOUSE* %"mouse*", %MOUSE** %"mouse*1"
  %count_cookies2 = alloca i32
  store i32 %count_cookies, i32* %count_cookies2
  %"mouse*3" = load %MOUSE*, %MOUSE** %"mouse*1"
  %tmp = getelementptr inbounds %MOUSE, %MOUSE* %"mouse*3", i32 0, i32 0
  %dr = load i32, i32* %tmp
  %count_cookies4 = load i32, i32* %count_cookies2
  %binop_int_tmp = add i32 %dr, %count_cookies4
  %"mouse*5" = load %MOUSE*, %MOUSE** %"mouse*1"
  %tmp6 = getelementptr inbounds %MOUSE, %MOUSE* %"mouse*5", i32 0, i32 0
  store i32 %binop_int_tmp, i32* %tmp6
  ret void
}

```

```
define i32 @MOUSE.Count_Cookies(%MOUSE* %"mouse*") {
entry:
  %"mouse*1" = alloca %MOUSE*
  store %MOUSE* %"mouse*", %MOUSE** %"mouse*1"
  %"mouse*2" = load %MOUSE*, %MOUSE** %"mouse*1"
  %tmp = getelementptr inbounds %MOUSE, %MOUSE* %"mouse*2", i32 0, i32 0
  %dr = load i32, i32* %tmp
  ret i32 %dr
}

define void @main() {
entry:
  %jerrys_cookies = alloca i32
  %Jerry = alloca %MOUSE*
  %malloccall = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32))
  %new_struct = bitcast i8* %malloccall to %MOUSE*
  store %MOUSE* %new_struct, %MOUSE** %Jerry
  %cookies = alloca i32
  %Jerry1 = load %MOUSE*, %MOUSE** %Jerry
  %tmp = getelementptr inbounds %MOUSE, %MOUSE* %Jerry1, i32 0, i32 0
  store i32 0, i32* %tmp
  %Jerry2 = load %MOUSE*, %MOUSE** %Jerry
  %tmp3 = getelementptr inbounds %MOUSE, %MOUSE* %Jerry2, i32 0, i32 0
  %dr = load i32, i32* %tmp3
  store i32 %dr, i32* %jerrys_cookies
  %Jerry4 = load %MOUSE*, %MOUSE** %Jerry
  %tmp5 = getelementptr inbounds %MOUSE, %MOUSE* %Jerry4, i32 0, i32 0
  store i32 2, i32* %tmp5
  %Jerry6 = load %MOUSE*, %MOUSE** %Jerry
  call void @MOUSE.Give_Cookie(%MOUSE* %Jerry6, i32 2)
  %Jerry7 = load %MOUSE*, %MOUSE** %Jerry
  %MOUSE.Count_Cookies_result = call i32 @MOUSE.Count_Cookies(%MOUSE* %Jerry7)
  store i32 %MOUSE.Count_Cookies_result, i32* %jerrys_cookies
  %Jerry8 = load %MOUSE*, %MOUSE** %Jerry
  %0 = bitcast %MOUSE* %Jerry8 to i8*
  tail call void @free(i8* %0)
  ret void
}

declare i32 @printf(i8*, ...)

declare i32 @custom_scanf(i8**)

declare i32 @atoi(i8*)

declare double @atof(i8*)

declare i8* @custom_itoa(i32)
```

```
71
72  declare i8* @custom_ftoa(double)
73
74  declare i32 @custom_strcmp(i8*, i8*)
75
76  declare i8* @custom_strcat(i8*, i8*)
77
78  declare noalias i8* @malloc(i32)
79
80  declare void @free(i8*)
```

2. The following program, `test_simple_math.meow` creates some custom functions to perform some simple mathematical operations.

Listing 23: `test/test_programs/test_simple_math.meow`

```
1
2   HAI ITZ ME NUMBR FUNC Add_Three WIT NUMBR num1 AN NUMBR num2 AN NUMBR num3,
3           PSST Trying to add three numbers
4
5           ITZ ME NUMBR sum.
6           sum IZ SUM OF num1 AN SUM OF num2 AN num3.
7           GIVE sum.
8   KBYE
9
10  HAI ITZ ME NUMBR FUNC Subtract_Two WIT NUMBR num1 AN NUMBR num2,
11          PSST Trying to add three numbers
12
13          ITZ ME NUMBR diff.
14          diff IZ DIFF OF num1 AN num2.
15          GIVE diff.
16  KBYE
17
18  HAI ITZ ME NUMBR FUNC Multiply_Four WIT NUMBR num1 AN NUMBR num2
19          AN NUMBR num3 AN NUMBR num4,
20
21          PSST Trying to add multiply four numbers
22
23          ITZ ME NUMBR product.
24          product IZ PRODUKT OF num1 AN PRODUKT OF num2 AN PRODUKT OF num3 AN num4.
25          GIVE product.
26  KBYE
27
28  HAI ITZ ME NUMBR FUNC Divide_Two WIT NUMBR num1 AN NUMBR num2,
29
30          PSST Trying to divide two numbers
31
32          ITZ ME NUMBR quotient.
33          quotient IZ QUOSHUNT OF num1 AN num2.
```

```
        GIVE quotient.
KBYE


HAI ITZ ME NUMBR FUNC Main,

        ITZ ME NUMBR one IZ 1.
        ITZ ME NUMBR two IZ 2.
        ITZ ME NUMBR three IZ 3.
        ITZ ME NUMBR sum.
        ITZ ME NUMBR diff.
        ITZ ME NUMBR product.
        ITZ ME NUMBR quotient.

        PSST * ATTEMPT TO ADD *
        sum IZ PURR Add_Three WIT one AN two AN three.
        PURR Meow WIT sum.

        PSST * ATTEMPT TO SUBTRACT *
        diff IZ PURR Subtract_Two WIT three AN two.
        PURR Meow WIT diff.

        PSST * ATTEMPT TO MULTIPLY *
        product IZ PURR Multiply_Four WIT one AN two AN diff AN sum.
        PURR Meow WIT product.

        PSST * ATTEMPT TO DIVIDE *
        quotient IZ PURR Divide_Two WIT sum AN two.
        PURR Meow WIT quotient.

        GIVE 0.
KBYE
```

Listing 24: target_output/test_simple_math_output.ll

```
; ModuleID = 'Meowlang'
source_filename = "Meowlang"

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00", align 1
@fmt.1 = private unnamed_addr constant [4 x i8] c"%d\0A\00", align 1
@fmt.2 = private unnamed_addr constant [4 x i8] c"%d\0A\00", align 1
@fmt.3 = private unnamed_addr constant [4 x i8] c"%d\0A\00", align 1

define i32 @main() {
entry:
  %one = alloca i32
  %two = alloca i32
  %three = alloca i32
```

```
14    %sum = alloca i32
15    %diff = alloca i32
16    %product = alloca i32
17    %quotient = alloca i32
18    store i32 1, i32* %one
19    store i32 2, i32* %two
20    store i32 3, i32* %three
21    %three1 = load i32, i32* %three
22    %two2 = load i32, i32* %two
23    %one3 = load i32, i32* %one
24    %Add_Three_result = call i32 @Add_Three(i32 %one3, i32 %two2, i32 %three1)
25    store i32 %Add_Three_result, i32* %sum
26    %sum4 = load i32, i32* %sum
27    %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt, i32 0, i32
        0), i32 %sum4)
28    %two5 = load i32, i32* %two
29    %three6 = load i32, i32* %three
30    %Subtract_Two_result = call i32 @Subtract_Two(i32 %three6, i32 %two5)
31    store i32 %Subtract_Two_result, i32* %diff
32    %diff7 = load i32, i32* %diff
33    %printf8 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0,
        i32 0), i32 %diff7)
34    %sum9 = load i32, i32* %sum
35    %diff10 = load i32, i32* %diff
36    %two11 = load i32, i32* %two
37    %one12 = load i32, i32* %one
38    %Multiply_Four_result = call i32 @Multiply_Four(i32 %one12, i32 %two11, i32 %diff10, i32 %sum9)
39    store i32 %Multiply_Four_result, i32* %product
40    %product13 = load i32, i32* %product
41    %printf14 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.2, i32 0,
        i32 0), i32 %product13)
42    %two15 = load i32, i32* %two
43    %sum16 = load i32, i32* %sum
44    %Divide_Two_result = call i32 @Divide_Two(i32 %sum16, i32 %two15)
45    store i32 %Divide_Two_result, i32* %quotient
46    %quotient17 = load i32, i32* %quotient
47    %printf18 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.3, i32 0,
        i32 0), i32 %quotient17)
48    ret i32 0
49  }
50
51  define i32 @Divide_Two(i32 %num1, i32 %num2) {
52  entry:
53    %num11 = alloca i32
54    store i32 %num1, i32* %num11
55    %num22 = alloca i32
56    store i32 %num2, i32* %num22
57    %quotient = alloca i32
```

```
  %num13 = load i32, i32* %num11
  %num24 = load i32, i32* %num22
  %binop_int_tmp = sdiv i32 %num13, %num24
  store i32 %binop_int_tmp, i32* %quotient
  %quotient5 = load i32, i32* %quotient
  ret i32 %quotient5
}

define i32 @Multiply_Four(i32 %num1, i32 %num2, i32 %num3, i32 %num4) {
entry:
  %num11 = alloca i32
  store i32 %num1, i32* %num11
  %num22 = alloca i32
  store i32 %num2, i32* %num22
  %num33 = alloca i32
  store i32 %num3, i32* %num33
  %num44 = alloca i32
  store i32 %num4, i32* %num44
  %product = alloca i32
  %num15 = load i32, i32* %num11
  %num26 = load i32, i32* %num22
  %num37 = load i32, i32* %num33
  %num48 = load i32, i32* %num44
  %binop_int_tmp = mul i32 %num37, %num48
  %binop_int_tmp9 = mul i32 %num26, %binop_int_tmp
  %binop_int_tmp10 = mul i32 %num15, %binop_int_tmp9
  store i32 %binop_int_tmp10, i32* %product
  %product11 = load i32, i32* %product
  ret i32 %product11
}

define i32 @Subtract_Two(i32 %num1, i32 %num2) {
entry:
  %num11 = alloca i32
  store i32 %num1, i32* %num11
  %num22 = alloca i32
  store i32 %num2, i32* %num22
  %diff = alloca i32
  %num13 = load i32, i32* %num11
  %num24 = load i32, i32* %num22
  %binop_int_tmp = sub i32 %num13, %num24
  store i32 %binop_int_tmp, i32* %diff
  %diff5 = load i32, i32* %diff
  ret i32 %diff5
}

define i32 @Add_Three(i32 %num1, i32 %num2, i32 %num3) {
entry:
```

```
  %num11 = alloca i32
  store i32 %num1, i32* %num11
  %num22 = alloca i32
  store i32 %num2, i32* %num22
  %num33 = alloca i32
  store i32 %num3, i32* %num33
  %sum = alloca i32
  %num14 = load i32, i32* %num11
  %num25 = load i32, i32* %num22
  %num36 = load i32, i32* %num33
  %binop_int_tmp = add i32 %num25, %num36
  %binop_int_tmp7 = add i32 %num14, %binop_int_tmp
  store i32 %binop_int_tmp7, i32* %sum
  %sum8 = load i32, i32* %sum
  ret i32 %sum8
}

declare i32 @printf(i8*, ...)

declare i32 @custom_scanf(i8**)

declare i32 @atoi(i8*)

declare double @atof(i8*)

declare i8* @custom_itoa(i32)

declare i8* @custom_ftoa(double)

declare i32 @custom_strcmp(i8*, i8*)

declare i8* @custom_strcat(i8*, i8*)
```

3. The following program, `test_array_access5.meow` creates several custom `Pet` objects and stores them in an array, from which they are accessed in the main program.

Listing 25: `test/test_programs/test_array_access5.meow`

```

HAI ITZ ME CLASS PET,

    ITZ ME YARN name.
    ITZ ME YARN type.
    ITZ ME NUMBR age.

    HAI ITZ ME YARN FUNC Get_Name,
        GIVE name.
    KBYE

```

```
KBYE


HAI ITZ ME NUMBR FUNC Main,

    ITZ ME NUMBR size IZ 2.
    ITZ ME NUMBR idx IZ 1.
    ITZ ME PET Cat.
    ITZ ME YARN cat_name.

    PSST Make some pets
    MAEK Silvester NEW PET,
        WIT name IZ "Silvester"
        AN type IZ "cat"
        AN age IZ 4.

    MAEK Tank NEW PET,
        WIT name IZ "Tank"
        AN type IZ "dog"
        AN age IZ 2.

    PSST Make an array of objects
    MAEK my_pets NEW BUCKET OF PET HOLDS size.

    my_pets[0] IZ Silvester.

    PSST Make a printer instance an print element at index 'idx'
    cat_name IZ PURR Get_Name IN my_pets[0].
    PURR Meow WIT cat_name.

    BLEEP my_pets.
    BLEEP Silvester.
    BLEEP Tank.
    GIVE 0.

KBYE
```

Listing 26: target_output/test_array_access_output.ll

```
; ModuleID = 'Meowlang'
source_filename = "Meowlang"

%PET = type { i8*, i8*, i32 }

@str = private unnamed_addr constant [4 x i8] c"cat\00", align 1
@str.1 = private unnamed_addr constant [10 x i8] c"Silvester\00", align 1
@str.2 = private unnamed_addr constant [4 x i8] c"dog\00", align 1
@str.3 = private unnamed_addr constant [5 x i8] c"Tank\00", align 1
```

```
@fmt = private unnamed_addr constant [4 x i8] c"%s\0A\00", align 1

define i8* @PET.Get_Name(%PET* %"pet*") {
entry:
  %"pet*1" = alloca %PET*
  store %PET* %"pet*", %PET** %"pet*1"
  %"pet*2" = load %PET*, %PET** %"pet*1"
  %tmp = getelementptr inbounds %PET, %PET* %"pet*2", i32 0, i32 0
  %dr = load i8*, i8** %tmp
  ret i8* %dr
}

define i32 @main() {
entry:
  %size = alloca i32
  %idx = alloca i32
  %Cat = alloca %PET*
  %cat_name = alloca i8*
  store i32 2, i32* %size
  store i32 1, i32* %idx
  %Silvester = alloca %PET*
  %malloccall = tail call i8* @malloc(i32 ptrtoint (%PET* getelementptr (%PET, %PET* null, i32 1) to i32))
  %new_struct = bitcast i8* %malloccall to %PET*
  store %PET* %new_struct, %PET** %Silvester
  %age = alloca i32
  %Silvester1 = load %PET*, %PET** %Silvester
  %tmp = getelementptr inbounds %PET, %PET* %Silvester1, i32 0, i32 2
  store i32 4, i32* %tmp
  %type = alloca i8*
  %Silvester2 = load %PET*, %PET** %Silvester
  %tmp3 = getelementptr inbounds %PET, %PET* %Silvester2, i32 0, i32 1
  store i8* getelementptr inbounds ([4 x i8], [4 x i8]* @str, i32 0, i32 0), i8** %tmp3
  %name = alloca i8*
  %Silvester4 = load %PET*, %PET** %Silvester
  %tmp5 = getelementptr inbounds %PET, %PET* %Silvester4, i32 0, i32 0
  store i8* getelementptr inbounds ([10 x i8], [10 x i8]* @str.1, i32 0, i32 0), i8** %tmp5
  %Tank = alloca %PET*
  %malloccall6 = tail call i8* @malloc(i32 ptrtoint (%PET* getelementptr (%PET, %PET* null, i32 1) to i32)
    )
  %new_struct7 = bitcast i8* %malloccall6 to %PET*
  store %PET* %new_struct7, %PET** %Tank
  %age8 = alloca i32
  %Tank9 = load %PET*, %PET** %Tank
  %tmp10 = getelementptr inbounds %PET, %PET* %Tank9, i32 0, i32 2
  store i32 2, i32* %tmp10
  %type11 = alloca i8*
  %Tank12 = load %PET*, %PET** %Tank
  %tmp13 = getelementptr inbounds %PET, %PET* %Tank12, i32 0, i32 1
```

```
  store i8* getelementptr inbounds ([4 x i8], [4 x i8]* @str.2, i32 0, i32 0), i8** %tmp13
  %name14 = alloca i8*
  %Tank15 = load %PET*, %PET** %Tank
  %tmp16 = getelementptr inbounds %PET, %PET* %Tank15, i32 0, i32 0
  store i8* getelementptr inbounds ([5 x i8], [5 x i8]* @str.3, i32 0, i32 0), i8** %tmp16
  %my_pets = alloca %PET**
  %size17 = load i32, i32* %size
  %mallocsize = mul i32 %size17, ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32)
  %malloccall18 = tail call i8* @malloc(i32 %mallocsize)
  %create_heap_array = bitcast i8* %malloccall18 to %PET**
  store %PET** %create_heap_array, %PET*** %my_pets
  %Silvester19 = load %PET*, %PET** %Silvester
  %arr_ptr = load %PET**, %PET*** %my_pets
  %element_ptr = getelementptr %PET*, %PET** %arr_ptr, i32 0
  store %PET* %Silvester19, %PET** %element_ptr
  %arr_ptr20 = load %PET**, %PET*** %my_pets
  %element_ptr21 = getelementptr %PET*, %PET** %arr_ptr20, i32 0
  %array_entry = load %PET*, %PET** %element_ptr21
  %PET.Get_Name_result = call i8* @PET.Get_Name(%PET* %array_entry)
  store i8* %PET.Get_Name_result, i8** %cat_name
  %cat_name22 = load i8*, i8** %cat_name
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt, i32 0, i32
    0), i8* %cat_name22)
  %my_pets23 = load %PET**, %PET*** %my_pets
  %0 = bitcast %PET** %my_pets23 to i8*
  tail call void @free(i8* %0)
  %Silvester24 = load %PET*, %PET** %Silvester
  %1 = bitcast %PET* %Silvester24 to i8*
  tail call void @free(i8* %1)
  %Tank25 = load %PET*, %PET** %Tank
  %2 = bitcast %PET* %Tank25 to i8*
  tail call void @free(i8* %2)
  ret i32 0
}

declare i32 @printf(i8*, ...)

declare i32 @custom_scanf(i8**)

declare i32 @atoi(i8*)

declare double @atof(i8*)

declare i8* @custom_itoa(i32)

declare i8* @custom_ftoa(double)

declare i32 @custom_strcmp(i8*, i8*)
```

```
104
105 declare i8* @custom_strcat(i8*, i8*)
106
107 declare noalias i8* @malloc(i32)
108
109 declare void @free(i8*)
```

4. The following program, `test_scan2.meow` demonstrates our custom string concatenation and scanf functionality by asking a user for their favorite color.

Listing 27: `test/test_programs/test_scan2.meow`

```
1
2  HAI ITZ ME FUNC Main,
3
4          ITZ ME YARN favorite_color.
5
6          PSST Reading in content
7          PURR Meow WIT "Please tell me your favorite color: ".
8          PURR Scan WIT favorite_color.
9
10         PURR Meow WIT CAT "Your favorite color is: " AN favorite_color.
11
12         BLEEP favorite_color.
13 KBYE
```

Listing 28: `target_output/test_scan_output.ll`

```
1  ; ModuleID = 'Meowlang'
2  source_filename = "Meowlang"
3
4  @str = private unnamed_addr constant [37 x i8] c"Please tell me your favorite color: \00", align 1
5  @fmt = private unnamed_addr constant [4 x i8] c"%s\0A\00", align 1
6  @str.1 = private unnamed_addr constant [25 x i8] c"Your favorite color is: \00", align 1
7  @fmt.2 = private unnamed_addr constant [4 x i8] c"%s\0A\00", align 1
8
9  define void @main() {
10 entry:
11   %favorite_color = alloca i8*
12   %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt, i32 0, i32
       0), i8* getelementptr inbounds ([37 x i8], [37 x i8]* @str, i32 0, i32 0))
13   %custom_scanf = call i32 @custom_scanf(i8** %favorite_color)
14   %favorite_color1 = load i8*, i8** %favorite_color
15   %strcat_call = call i8* @custom_strcat(i8* getelementptr inbounds ([25 x i8], [25 x i8]* @str.1, i32 0,
       i32 0), i8* %favorite_color1)
16   %printf2 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.2, i32 0,
       i32 0), i8* %strcat_call)
17   %favorite_color3 = load i8*, i8** %favorite_color
18   tail call void @free(i8* %favorite_color3)
```

```
  ret void
}

declare i32 @printf(i8*, ...)

declare i32 @custom_scanf(i8**)

declare i32 @atoi(i8*)

declare double @atof(i8*)

declare i8* @custom_itoa(i32)

declare i8* @custom_ftoa(double)

declare i32 @custom_strcmp(i8*, i8*)

declare i8* @custom_strcat(i8*, i8*)

declare void @free(i8*)
```

6.3 Test Suites

The full test suite (source programs and corresponding expected outputs) can be found in the **Appendix, Section 8.2 Test Programs**.

7 Lessons Learned

- **Meg**: Our team setup our regression test suite relatively early in the project and at the time I really underestimated how helpful it would be. Being able to successfully run our regression test suite after making some potentially breaking changes was great piece of mind that the new code changes did not unintentionally introduce new bugs.
- **Michelle**: I learned that a big project such as this one is not as intimidating after breaking up everything into little small steps. Pair programming was a really helpful way to get unstuck with coding. With awesome team members working together, so much was accomplished in such little time!
- **Carolyn**: I learned that your teammates can make or break the project. We were very fortunate to have some very capable members on our team. Careful planning and frequent check-ins played a big part in the success of the project. I also learned that I should account for incidentals or delays in estimating completion time.
- **Lauren**: In many of my previous coding-heavy CS classes, I spent a lot of my time frantically throwing things at the wall to see what would stick because I wasn't confident enough to try to really understand. Because so much is accomplished with such little code in OCaml and in each step of the compiler, that approach is even more inefficient than it already is on average, so I was forced to study MicroC and

to completely understand everything before I worked on our own compiler. As a result, I was able to enjoy the learning process and gain a confidence in myself that I didn't have before. I am grateful to Professor Edwards, PLT, and my team for that!

8 Appendix

8.1 Source Code

This appendix section contains the `Meowlang` compiler source code.

Listing 29: `bin/meowlang.sh`

```
#!/bin/bash
LLC="llc"                               # LLVM compiler
CC="cc"                                 # C compiler

# get the path of the file, regardless of directory
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" &> /dev/null && pwd )"
MEOWLANG="$DIR/../src/meowlang.native"

# Paths to the relevant .o file to link into compiler
CUSTOM_SCAN="$DIR/../src/custom_scanf.o"
CUSTOM_STRCMP="$DIR/../src/custom_strcmp.o"
CUSTOM_STRCAT="$DIR/../src/custom_strcat.o"
CUSTOM_CASTING="$DIR/../src/custom_casting.o"

if [[ $# -le 0 ]]
then
        echo -e "Usage: ./meowlang.sh program_name.meow [flags]"
        echo -e "-A (compile and run)"
        echo -e "-c (compile and print)"
        echo -e "-s (semantic check)"
        echo -e "-a (ast pretty-print)"
        exit 1
fi

cd $(dirname "${1}")
name=$(basename $1 .meow)

if [[ $2 == "-a" ]]
then
        $MEOWLANG -f $(basename $1) -a
elif [[ $2 == "-s" ]]
then
        $MEOWLANG -f $(basename $1) -s
elif [[ $2 == "-c" ]]
then
        $MEOWLANG -f $(basename $1) -c
else
        # build and run everything
        $MEOWLANG -f $(basename $1) -c > "$name.ll"

        if [[ $? -eq 0 ]];
```

```
        then
                $LLC -relocation-model=pic "$name.ll" > "$name.s" &&
                $CC -o "$name.exe" "$name.s" $CUSTOM_SCAN $CUSTOM_STRCMP $CUSTOM_STRCAT $CUSTOM_CASTING  &&
                "./$name.exe"
        fi
fi
```

Listing 30: `Makefile`

```
.PHONY : build_complier build_and_test compress

build_complier :
    cd ./src && $(MAKE) clean && $(MAKE) && cd ../

build_and_test: build_complier
    ./test/test_all.sh

compress : build_complier
    cd ./src && $(MAKE) clean && cd ../.. && tar czf meowlang.tar.gz ./meowlang
```

Listing 31: `src/Makefile`

```
# Run 'make' to build meowlang.native

.PHONY : all
all : meowlang.native custom_scanf.o custom_casting.o custom_strcmp.o custom_strcat.o

meowlang.native : parser.mly scanner.mll ast.ml semant.ml codegen.ml
    opam config exec -- \
    ocamlbuild -use-ocamlfind meowlang.native

.PHONY : clean
clean :
    ocamlbuild -clean
    rm -rf ocamlllvm *.diff parser.output custom_scanf.o custom_casting.o custom_strcmp.o custom_strcat.o

custom_scanf.o : ./lib/custom_scanf.c
    cc -c ./lib/custom_scanf.c

custom_casting.o : ./lib/custom_casting.c
    cc -c ./lib/custom_casting.c

custom_strcmp.o : ./lib/custom_strcmp.c
    cc -c ./lib/custom_strcmp.c

custom_strcat.o : ./lib/custom_strcat.c
    cc -c ./lib/custom_strcat.c
```

Listing 32: src/_tags

```
# Include the llvm and llvm.analysis packages while compiling
true: package(llvm), package(llvm.analysis)

# Enable almost all compiler warnings
true : warn(+a-4)

# Enable use of Str (non-standard lib)
true: use_str
```

Listing 33: src/ast.ml

```
(* Abstract Syntax Tree *)

type op = Add | Sub | Mult | Div | Equal | Neq | Less | Greater | And | Or | Concat | Increment | Decrement

type uop = Not

type array_size =
    ILiteralArraySize of int
  | VariableArraySize of string

type typ = Int | Bool | Float | String | Void | Obtype of string | Arrtype of array_size * typ

type import = string

type expr =
    ILiteral of int
  | Fliteral of string
  | BoolLit of bool
  | StringLit of string
  | Id of string
  | Cast of typ * expr
  | Binop of expr * op * expr
  | Unop of uop * expr
  | Assign of expr * expr
  | FunctionCall of string * expr list
  | MethodCall of expr * string * expr list
  | NewArray of string * typ * array_size * expr list
  | Noexpr
  | NewInstance of string * typ * expr list
  | ClassAccess of expr * string
  | ArrayAccess of string * expr

type bind_var = typ * string * expr
type bind_formals = typ * string

type stmt =
```

```
    Block of stmt list
  | Expr of expr
  | Return of expr
  | If of expr * stmt * stmt
  | For of op * expr * expr * expr * stmt
  | Dealloc of expr
  | ClassAssign of expr * string * expr
  | ArrayAssign of string * expr * expr

type func_decl = {
    typ : typ;
    fname : string;
    formals : bind_formals list;
    locals : bind_var list;
    body : stmt list;
  }

type class_decl = {
    cname : string;
    cvars : bind_var list;
    cfuncs : func_decl list;
}

type program = import list * func_decl list * class_decl list
```

Listing 34: src/codegen.ml

```
(*****************************************************************************)
(* Code generation: translate takes a semantically checked AST and produces   *)
(* LLVM IR. Note: This code inspired by codegen.ml of the                     *)
(* MicroC Compiler by S Edwards (PLT, Spring 2021)                            *)
(*****************************************************************************)
module L = Llvm
module A = Ast
open Exceptions
open Sast

module StringMap = Map.Make(String)

(* Hash table of structs representing user-defined classes *)
let struct_types:(string, L.lltype) Hashtbl.t = Hashtbl.create 10
let struct_field_idx:(string, int) Hashtbl.t = Hashtbl.create 10
let local_variables:(string,  L.llvalue) Hashtbl.t = Hashtbl.create 10
let global_functions:(string, (L.llvalue * sfunc_decl)) Hashtbl.t = Hashtbl.create 10

(* Finds a struct by its original class name *)
let find_struct_by_cls cls_name =
  try Hashtbl.find struct_types cls_name
```

```
22    with Not_found ->
23      let msg = Printf.sprintf "struct for class %s unknown" cls_name
24      in raise (UnknownStruct(msg))
25
26  (* Return the value for a variable, else raise error *)
27  let lookup_variable var_name env =
28    try Hashtbl.find env var_name
29    with _ ->
30      let msg = Printf.sprintf "codegen error: %s variable: %s" undeclared_msg var_name
31      in raise (VariableNotFound(msg))
32
33  (* Return the value for a function name or argument *)
34  let lookup_function func_name =
35    try Hashtbl.find global_functions func_name
36    with _ ->
37      let msg = Printf.sprintf "codegen error: %s function: %s" undeclared_msg func_name
38      in raise (FunctionNotFound(msg))
39
40  let lookup_index cls_name field_name =
41    try Hashtbl.find struct_field_idx (cls_name ^ "." ^ field_name)
42    with _ ->
43      let msg = Printf.sprintf "codegen error: %s.%s" cls_name field_name
44      in raise (InstanceVariableNotFound(msg))
45
46  let create_built_in return_typ args n m =
47    let func_t = L.function_type return_typ args in
48    L.declare_function n func_t m
49
50  let context    = L.global_context ()
51  let the_module = L.create_module context "Meowlang"
52
53  (* Easier to not have to rewrite these types*)
54  let i1_t      = L.i1_type     context
55  let i8_t      = L.i8_type     context
56  let i32_t     = L.i32_type    context
57  let float_t   = L.double_type context
58  let void_t    = L.void_type   context
59  let str_t     = L.pointer_type i8_t
60
61  (* Finds the LLVM type corresponding to the Meowlang type *)
62  let rec ltype_of_typ = function
63      A.Int           -> i32_t
64    | A.Bool          -> i1_t
65    | A.Float         -> float_t
66    | A.Void          -> void_t
67    | A.String        -> str_t
68    | A.Obtype(c)     -> L.pointer_type (find_struct_by_cls c)
69    | A.Arrtype(_, t) -> L.pointer_type (ltype_of_typ t)
```

```
70
71  (* Creates a function prototype *)
72  let create_func_prototype fdecl =
73    let name =  fdecl.sfname
74    and return_typ = ltype_of_typ fdecl.styp
75    and formal_types =
76      Array.of_list (List.map (fun (t,_) -> ltype_of_typ t) fdecl.sformals)
77    in
78    let ftype = L.function_type return_typ formal_types in
79    L.define_function name ftype the_module, fdecl
80
81  (* Declare and setup required hash tables for a struct based on an object *)
82  let codegen_struct cls =
83    (* save new struct to known struct list *)
84    let struct_t = L.named_struct_type context cls.scname in
85    Hashtbl.add struct_types cls.scname struct_t;
86
87    let var_name_list = List.map (fun (_, n, _) -> n) cls.scvars
88    and inst_var_ltyps = List.map (fun (t, _, _) -> ltype_of_typ t) cls.scvars in
89    List.iteri (
90      fun idx name -> Hashtbl.add struct_field_idx (cls.scname ^ "." ^ name) idx
91    ) var_name_list;
92    L.struct_set_body struct_t (Array.of_list inst_var_ltyps) false
93
94  (* Provides the c-equivalent formatting string for a given expr *)
95  let format (typ, _) =
96    match typ with
97      A.Int    -> "%d\n"
98    | A.Float  -> "%g\n"
99    | A.String -> "%s\n"
100   | A.Bool   -> "%d\n"
101   | _ -> raise (NotYetSupported("formatting for type not yet supported"))
102
103   let add_local (typ, local_name, _) env builder =
104     let new_local = L.build_alloca (ltype_of_typ typ) local_name builder
105     in Hashtbl.add env local_name new_local
106
107 (*********************************************************)
108 (* Build Function: Fill in the body of the each function *)
109 (*********************************************************)
110 let build_function fdecl =
111
112   (*  First create the prototypes for build in functions (I/O and casting) *)
113   let printf_func = (* variatic, so needs its own distinct builder *)
114     let printf_t = L.var_arg_function_type i32_t [| str_t |] in
115     L.declare_function "printf" printf_t the_module
116
117   and scanf_func = create_built_in i32_t [| L.pointer_type str_t |] "custom_scanf" the_module
```

```
118   and atoi_func = create_built_in i32_t [| str_t |] "atoi" the_module
119   and atof_func = create_built_in float_t [| str_t |] "atof" the_module
120   and itoa_func = create_built_in str_t [| i32_t |] "custom_itoa" the_module
121   and ftoa_func = create_built_in str_t [| float_t |] "custom_ftoa" the_module
122   and strcmp_func = create_built_in i32_t [| str_t ; str_t |] "custom_strcmp" the_module
123   and strcat_func = create_built_in str_t [| str_t ; str_t |] "custom_strcat" the_module
124   in
125
126   let (the_function, _) = Hashtbl.find global_functions fdecl.sfname in
127   let builder = L.builder_at_end context (L.entry_block the_function) in
128
129   (* Construct the functions locals (formal + local vars) *)
130   let add_formal acc (typ, formal_name) param =
131     (* set name of param value to corresponding formal name *)
132     L.set_value_name formal_name param;
133
134     (* allocate space stack for formal *)
135     let new_formal = L.build_alloca (ltype_of_typ typ) formal_name builder in
136     ignore (L.build_store param new_formal builder);  (* store %param %new_formal *)
137     Hashtbl.add local_variables formal_name new_formal;
138     (formal_name, new_formal) :: acc (* to make compiler happy *)
139   in
140
141   (* Clear existing locals and start over *)
142   Hashtbl.clear local_variables;
143   let params = (Array.to_list (L.params the_function)) in
144   ignore(List.fold_left2 add_formal [] fdecl.sformals params);
145   List.iter (fun l -> add_local l local_variables builder) fdecl.slocals;
146
147   (* Formatting expressions, for printf-ing *)
148   let format_str fmt = L.build_global_stringptr fmt "fmt" builder in
149
150   (*********************************************)
151   (* Helper function for building up expressions  *)
152   (*********************************************)
153   let rec expr builder ((_, e) : sexpr) env =
154     match e with
155       SILiteral i  -> L.const_int i32_t i
156     | SBoolLit b   -> L.const_int i1_t (if b then 1 else 0)
157     | SFliteral l  -> L.const_float_of_string float_t l
158     | SStringLit s -> L.build_global_stringptr s "str" builder
159     | SId var      -> L.build_load (lookup_variable var env) var builder
160     | SNoexpr      -> L.const_int i32_t 0
161
162     | SCast(t, (typ, e))  ->
163       let rhs = expr builder (typ, e) env
164       and llvm_typ = ltype_of_typ t in
165       (match typ with
```

```
166         A.Float when t = A.Int -> L.build_fptosi rhs llvm_typ "cast_v" builder
167       | A.Int when t = A.Float -> L.build_uitofp rhs llvm_typ "cast_v" builder
168       | A.String when t = A.Float ->
169           L.build_call atof_func [| rhs |] "atof_call" builder
170       | A.String when t = A.Int ->
171           L.build_call atoi_func [| rhs |] "atoi_call" builder
172       | A.Int when t = A.String ->
173           L.build_call itoa_func [| rhs |] "itoa_call" builder
174       | A.Float when t = A.String ->
175           L.build_call ftoa_func [| rhs |] "ftoa_call" builder
176       | _ -> raise (NotYetSupported("codegen: cast operation not yet supported")))
177
178     | SAssign ((_, lhs), e)   ->
179       let rhs = expr builder e env in
180       let lhs =
181         match lhs with
182         (* Used in assigning variables; not used in assigning class members *)
183           SId(var) -> lookup_variable var env
184         | SClassAccess(A.Obtype(cname), ((_, SId(_)) as v), inst_v) ->
185             let index = lookup_index cname inst_v
186             and load_tmp = expr builder v env in
187             L.build_struct_gep load_tmp index "tmp" builder
188         | SArrayAccess(v, e) ->
189             let index_args = [|expr builder e env|] in
190             let arr = L.build_load (lookup_variable v env) "arr_ptr" builder in
191             L.build_gep arr index_args "element_ptr" builder
192
193         | _ -> raise (NotYetSupported("codegen: assignment op not yet supported"))
194       in ignore(L.build_store rhs lhs builder); rhs
195
196     | SUnop(_, ((_, _) as e)) ->
197       let e' = expr builder e env in
198       L.build_not e' "tmp" builder
199
200     (* Binary operation between two integers *)
201     | SBinop(((A.Int, _) as e1), op, ((A.Int, _) as e2)) ->
202       let lhs = expr builder e1 env
203       and rhs = expr builder e2 env in
204       (match op with
205         A.Add       -> L.build_add
206       | A.Sub       -> L.build_sub
207       | A.Mult      -> L.build_mul
208       | A.Div       -> L.build_sdiv
209       | A.Equal     -> L.build_icmp L.Icmp.Eq
210       | A.Neq       -> L.build_icmp L.Icmp.Ne
211       | A.Less      -> L.build_icmp L.Icmp.Slt
212       | A.Greater   -> L.build_icmp L.Icmp.Sgt
213       | A.Increment -> L.build_add
```

```
214       | A.Decrement -> L.build_sub
215       | _         ->
216         let msg = "found binary operation not supported for two integers"
217         in raise (NotYetSupported(msg))
218       ) lhs rhs "binop_int_tmp" builder
219
220     (* String concatenation *)
221     | SBinop(((A.String, _) as e1), A.Concat, ((A.String, _) as e2)) ->
222       let lhs = expr builder e1 env
223       and rhs = expr builder e2 env in
224       L.build_call strcat_func [| lhs ; rhs |] "strcat_call" builder
225
226     | SBinop(((A.String, _) as e1), A.Concat, ((_, _) as e2)) ->
227       let lhs = expr builder e1 env
228       and rhs = expr builder (A.String, SCast(A.String, e2)) env in
229       L.build_call strcat_func [| lhs ; rhs |] "strcat_call" builder
230
231     | SBinop(((_, _) as e1), A.Concat, ((A.String, _) as e2)) ->
232       let lhs = expr builder (A.String, SCast(A.String, e1)) env
233       and rhs = expr builder e2 env in
234       L.build_call strcat_func [| lhs ; rhs |] "strcat_call" builder
235
236     (* String comparison *)
237     | SBinop(((A.String, _) as e1), A.Equal, ((A.String, _) as e2)) ->
238       let lhs = expr builder e1 env
239       and rhs = expr builder e2 env in
240       L.build_call strcmp_func [| lhs ; rhs |] "stcmp_call" builder
241
242     (* Binary operation between one or more floats *)
243     | SBinop(((A.Float as t), (_ as v1)), op, ((A.Int as o), (_ as v2)))
244     | SBinop(((A.Int as t), (_ as v1)), op, ((A.Float as o, (_ as v2))))
245     | SBinop(((A.Float as t), (_ as v1)), op, ((A.Float as o), (_ as v2))) ->
246       let build_cast v =
247         L.build_uitofp v float_t "cast_v" builder
248       in
249       let lhs =
250         let tmp_l = expr builder (A.Float, v1) env in
251         if t = A.Int then build_cast tmp_l
252         else tmp_l
253       and rhs =
254         let tmp_r = expr builder (A.Float, v2) env in
255         if o = A.Int then build_cast tmp_r
256         else tmp_r
257       in
258       (match op with
259         A.Add     -> L.build_fadd
260       | A.Sub     -> L.build_fsub
261       | A.Mult    -> L.build_fmul
```

```
262       | A.Div     -> L.build_fdiv
263       | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
264       | A.Neq     -> L.build_fcmp L.Fcmp.One
265       | A.Less    -> L.build_fcmp L.Fcmp.Olt
266       | A.Greater -> L.build_fcmp L.Fcmp.Ogt
267       | _         ->
268         let msg = "found binary operation not supported for one or more floats"
269       in raise (NotYetSupported(msg))
270       ) lhs rhs "binop_float_tmp" builder
271
272     (* Binary operation for two booleans *)
273     | SBinop(((A.Bool, _) as e1), op, ((A.Bool, _) as e2)) ->
274       let lhs = expr builder e1 env
275       and rhs = expr builder e2 env in
276       (match op with
277           A.Or    -> L.build_or
278         | A.And   -> L.build_and
279         | A.Neq   -> L.build_icmp L.Icmp.Ne
280         | A.Equal -> L.build_icmp L.Icmp.Eq
281         | _       ->
282           let msg = "found binary operation not supported for two boolean values"
283           in raise (NotYetSupported(msg))
284       ) lhs rhs "binop_bool_tmp" builder
285
286     (* Call to built in printf function *)
287     | SFunctionCall ("Meow", [arg]) ->
288       L.build_call printf_func [| format_str (format arg) ; (expr builder arg env) |] "printf" builder
289
290     | SFunctionCall ("Scan", [(_, SId(arg))]) ->
291       L.build_call scanf_func [| (lookup_variable arg env) |] "custom_scanf" builder
292
293     (* Call a user-defined function *)
294     | SFunctionCall (fname, args) ->
295       let (fdef, fdecl) = lookup_function fname in
296       let llargs = List.rev (List.map (fun a -> expr builder a env) (List.rev args))
297       and result = (
298         match fdecl.styp with
299           A.Void -> ""
300         | _ -> fname ^ "_result"
301       ) in
302       L.build_call fdef (Array.of_list llargs) result builder
303
304     (* Create a new struct instance of class c with variable name n*)
305     | SNewInstance(v, c, constructor_exprs) ->
306       (match c with
307         A.Obtype (cname) as cls -> (
308         (* add new variable to local vars; a pointer to malloc'd item *)
309         add_local (cls, v, None) env builder;
```

```
310         (* assign the malloc to the new variable *)
311         let rhs = L.build_malloc (find_struct_by_cls cname) "new_struct" builder
312         and lhs = lookup_variable v env in
313         ignore(L.build_store rhs lhs builder);
314
315         (* now the tricky nonsense of the constructor; since we want the default args
316         to be calculatable against each other, must build a separate symbol table *)
317         let constructor_vars:(string, L.llvalue) Hashtbl.t = Hashtbl.create 10 in
318         (* add all local entries... *)
319         let _ = Hashtbl.iter (fun k v -> Hashtbl.add constructor_vars k v) local_variables in
320         let build_constructor v (typ, e) =
321           (* add it to a "constructor" symbol table *)
322           (match e with
323               SAssign((_, SClassAccess(_, (_, _), id)), _) ->
324               ignore(add_local (typ, id, e) constructor_vars builder);
325               (expr builder (typ, e) constructor_vars) :: v (* build it! *)
326             | _ ->
327               let msg = "codegen: class constructor pattern not yet supported"
328               in raise (NotYetSupported(msg)))
329         in
330         ignore(List.fold_left build_constructor [] constructor_exprs); rhs
331         )
332       | _ -> raise (ObjectCreationInvalid("codegen: cannot create instance of anything but Obtype")))
333
334     | SClassAccess(A.Obtype(cname), v, inst_v) ->
335       let tmp_value = expr builder v env
336       and index = lookup_index cname inst_v in
337       let deref = L.build_struct_gep tmp_value index "tmp" builder in
338       L.build_load deref "dr" builder
339
340     | SNewArray (v, i_typ, n, setup_exprs) ->
341       (* add new variable to local vars; a pointer to malloc'd item *)
342       add_local (A.Arrtype(n, i_typ), v, None) env builder;
343       (* determine value of n *)
344       let n_val =
345         match n with
346           A.ILiteralArraySize i  -> L.const_int i32_t i
347         | A.VariableArraySize id -> expr builder (A.Int, SId(id)) env
348       and ar_typ = ltype_of_typ i_typ
349       in
350       let rhs = L.build_array_malloc ar_typ n_val "create_heap_array" builder
351       and lhs = lookup_variable v env in
352       ignore(L.build_store rhs lhs builder);
353
354       (* now do the setup expressions, where the array contents are assigned *)
355       (* this is like a constructor for a new array *)
356       ignore(List.map (fun e -> expr builder e env) (List.rev setup_exprs)); rhs
357
```

```
    | SArrayAccess (v, e) ->
      let index_args = [|expr builder e env|]
      and arr = L.build_load (lookup_variable v env) "arr_ptr" builder in
      let gep = L.build_gep arr index_args "element_ptr" builder in
      L.build_load gep "array_entry" builder

    | _ -> raise (NotYetSupported("found expr or functions not yet supported"))
  in

  (* LLVM insists each basic block end with exactly one "terminator"
     instruction that transfers control. This function runs "instr builder"
     if the current block does not already have a terminator. *)
  let add_terminal builder instr =
    match L.block_terminator (L.insertion_block builder) with
      Some _ -> ()
    | None -> ignore (instr builder) in

  (*****************************************************)
  (* Helper function for building up statements        *)
  (* Returns the builder for the statement's successor *)
  (*****************************************************)
  let rec stmt builder = function
      SBlock sl   -> List.fold_left stmt builder sl
    | SExpr e     -> ignore(expr builder e local_variables); builder
    | SReturn e   -> ignore(
      match fdecl.styp with
        (* Special "return nothing" instr *)
        A.Void -> L.build_ret_void builder
        (* Build return statement *)
      | _ -> L.build_ret (expr builder e local_variables) builder
      ); builder

    | SIf (predicate, then_stmt, else_stmt) ->
      let bool_val = expr builder predicate local_variables in
      let merge_bb = L.append_block context "merge" the_function in
      let build_br_merge = L.build_br merge_bb in (* partial function *)

      let then_bb = L.append_block context "then" the_function in
      add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
      build_br_merge;

      let else_bb = L.append_block context "else" the_function in
      add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
      build_br_merge;

      ignore(L.build_cond_br bool_val then_bb else_bb builder);
      L.builder_at_end context merge_bb

```

```
406     | SFor (inc_decrement, index, opt_index_assignment, termination_comparison, loop_body) ->
407       (* perform the index assignment if it exists *)
408       ignore(expr builder opt_index_assignment local_variables);
409
410       let pred_bb = L.append_block context "for" the_function in
411       ignore(L.build_br pred_bb builder);
412
413       let body_bb = L.append_block context "for_body" the_function
414       (* create the SBinop expr for incrementing and decrementing *)
415       and binop = (A.Int, SBinop(index, inc_decrement, (A.Int, SILiteral 1))) in
416       add_terminal (stmt
417         (L.builder_at_end context body_bb)
418         (* append assigning index inc/decrement to the end of the loop body *)
419         (SBlock [loop_body ; SExpr(A.Int, SAssign(index, binop))])
420       )
421         (L.build_br pred_bb);
422
423       let pred_builder = L.builder_at_end context pred_bb in
424       let bool_val = expr pred_builder termination_comparison local_variables in
425
426       let merge_bb = L.append_block context "merge" the_function in
427       ignore(L.build_cond_br bool_val body_bb merge_bb pred_builder);
428       L.builder_at_end context merge_bb
429
430     | SClassAssign (A.Obtype(cname), v, inst_v, e) ->
431       let rhs = expr builder e local_variables
432       and lhs =
433         let index = lookup_index cname inst_v
434         and load_tmp = expr builder v local_variables in
435         L.build_struct_gep load_tmp index "tmp" builder
436       in
437       ignore(L.build_store rhs lhs builder); builder
438
439     | SArrayAssign (v, idx_e, (t, assign_e)) ->
440       (* Utilize code already created to handle assignment expressions *)
441       let converted_expr =
442         (t, SAssign((t, SArrayAccess(v, idx_e)), (t, assign_e)))
443       in
444       ignore(expr builder converted_expr local_variables); builder
445
446     | SDealloc (v) ->
447       let tmp_value = expr builder v local_variables in
448       ignore(L.build_free (tmp_value) builder); builder
449
450     | _ -> raise (NotYetSupported("complex stmts not yet supported"))
451   in
452
453   (* Build the code for each statement in the function *)
```

```
  let builder = stmt builder (SBlock fdecl.sbody) in

  (* Add a return if the last block falls off the end *)
  add_terminal builder (
    match fdecl.styp with
      A.Void -> L.build_ret_void
    | A.Float -> L.build_ret (L.const_float float_t 0.0)
    | t -> L.build_ret (L.const_int (ltype_of_typ t) 0)
  )

(*****************************************************************************)
(* Translate : Takes a Sast.program and produces an Llvm.module              *)
(*                                                                           *)
(* Note that all "actionable" items in the program are now functions         *)
(* (there are no longer class methods/etc). Classes are useful because       *)
(* they provide the struct definition corresponding to the class, but        *)
(* all class methods have been lifted to global scope in the semantic        *)
(* checking step.                                                            *)
(*****************************************************************************)
let translate (_, functions, classes) =

  (* Create the relevant structs, based on the class definitions *)
  List.iter codegen_struct classes;

  (* Create each function's prototype *)
  (* this so we can call it even before we've created its body. *)
  (* this function builds up a map of function_name: prototype  *)
  (* Maps {function_name: (func_prototype, func_decl)} *)
  let add_functions_to_map fdecl =
    Hashtbl.add global_functions fdecl.sfname (create_func_prototype fdecl) in
  List.iter add_functions_to_map functions;

  (* Build each function *)
  List.iter build_function functions;
  the_module
```

Listing 35: `src/exceptions.ml`

```
(************ Exceptions ************)

(* Expression/Statement  Exceptions *)
exception DuplicateIdentifier of string
exception IllegalAssignment
exception VariableNotFound of string
exception IllegalBinaryOp of string
exception IllegalUnaryOp of string
exception VariableAssignmentError of string
exception ControlFlowIllegalArgument of string
```

```
11 exception CastUnnecessary of string
12
13 (* Function Exceptions *)
14 exception MissingMainFunction of string
15 exception FunctionNotFound of string
16 exception FunctionArgumentLengthMismatch of string
17 exception ArgumentTypeMismatch of string
18 exception ReturnFromVoidFunction of string
19 exception ReturnTypeInvalid of string
20
21 (* Object Exceptions *)
22 exception ClassNotFound of string
23 exception InstanceVariableNotFound of string
24 exception InvalidDealloc of string
25 exception InvalidClassMemberAssignment of string
26 exception ClassMethodNotFound of string
27 exception MethodArgumentLengthMismatch of string
28 exception InvalidMethodCall of string
29 exception ObjectCreationInvalid of string
30 exception ObjectConstructorInvalid of string
31 exception ObjectInstanceVariableInvalid of string
32 exception InstanceVariableAccessInvalid of string
33 exception UnknownStruct of string
34
35 (* Array Exceptions *)
36 exception InvalidArraySizeSpecified of string
37 exception InvalidArrayItem of string
38 exception InvalidArrayAccess of string
39 exception ExcessArrayInput of string
40 exception InvalidArrayAssignment of string
41
42 (* Other *)
43 exception ImportNotFound of string
44 exception DuplicateImport of string
45 exception NotYetSupported of string
46 exception InternalError of string
47
48
49 (* Message Templates for Exceptions *)
50 let dup_func_msg                = "duplicate function name, or conflict with built-in:"
51 let dup_class_msg               = "duplicate class name:"
52 let dup_method_msg              = "duplicate class method name:"
53 let dup_instance_var_msg        = "duplicate instance variable name identified in class declaration:"
54 let dup_formal_msg              = "duplicate formal name :"
55 let dup_local_var_msg           = "duplicate local variable name :"
56 let dup_form_local_msg          = "duplicate identifers (formal and local) name :"
57 let dup_form_instance_msg       = "duplicate identifier (formal and local conflict with instance vars)"
58 let undeclared_msg              = "undeclared identifier: "
```

```
59 let assignment_typ_mismatch     = "variables can only be assigned to items of the expected type"
60 let missing_main_func_msg       = "all programs must have a 'Main' function"
61 let func_arg_num_mismatch       =  "expected different number of arguments for function: "
62 let meth_arg_num_mismatch       =  "expected different number of arguments for method: "
63 let expr_type_mismatch          = "expression type mismatch: " (* expected _ but got _ in expression _ *)
64 let op_type_mismatch_inc_dec    = "operation type mismatch: expected Increment or Decrement but got type"
65 let op_type_mismatch_boolean    = "operation type mismatch: expected Boolean but got type"
66 let op_type_mismatch_int        = "operation type mismatch: expected Integer but got type"
67 let op_type_mismatch_loop_term  = "operation type mismatch: expected <, >, =, != as loop termination condition:"
68 let class_method_unknown        = "method does not exist for this class:"
69 let invalid_method_call         = "methods can only be called on objects:"
70 let invalid_array_size_msg      = "arrays sizes must be integer literals or variables only:"
71 let invalid_object_creation     = "you can only create objects from classes"
72 let invalid_instance_var_access = "instance variables only exist in classes"
73 let invalid_array_item_msg      = "array elements must be of specified type for the array"
74 let excess_array_item           = "array contents exceeded specified array size"
75 let invalid_deallocation_msg    = "BLEEP (free) can only be called on variables of object or array type that have
       been allocated:"
76 let invalid_cls_member_assign   = "you must assign valid instance variables within a class with items of valid type
       : tried to assign"
77 let member_assign_cls_only      = "you can only assign instance variables for class objects:"
78 let array_access_array_only     = "array indexing is only available for array types"
79 let array_access_integer        = "arrays can only be indexed with integer types (>= 0):"
80 let array_access_out_of_bounds  = "index exceeds array size"
81 let object_constructor_error    = "to assign instance variables on object creation, you must use assignment
       expressions"
82 let object_constructor_types    = "you may only assign instance variables that are defined within the class and
       that are of the correct type"
83 let use_of_this_outside_class   = "use of HERE ('this') keyword can only be used inside of a class to refer to its
       own methods/variables"
84 let class_access_msg            = "class access is performed on an object identified by a variable"
85 let method_call_expr            = "attempting to call method on invalid expression; must be variable representing
       an object type or an indexed array of objects"
86 let return_from_void_func       = "attempting to return from a void function"
87 let return_type_invalid         = "attempting to return value that doesn't match function return type"
88 let scan_error                  = "built in scan function takes only the id of a string as an argument"
```

Listing 36: `src/import.ml`

```
1 (*****************************************************************************)
2 (* Performs import-specific semantic checks on the AST,                      *)
3 (*  Scans and parses each import file, Produces AST which is the importing   *)
4 (* file's AST appended by the ASTs of the import files appended              *)
5 (*****************************************************************************)
6 open Exceptions
7
8 let imported_asts:(string, Ast.program) Hashtbl.t = Hashtbl.create 10
9
```

```
let unique_list import_list =
  let unique =
    List.fold_left (
      fun acc item -> if List.mem item acc then acc else item :: acc
    ) [] import_list
  in
  if (List.length import_list == List.length unique) then true
  else raise (DuplicateImport("Duplicate import in .meow file"))

(* Check that the import exists; only files in the cwd are supported *)
let check_import_exists import =
  if (Sys.file_exists import) then ()
  else raise (ImportNotFound("Could not find import"))

(* Check that import name is capitalized filename without the extension *)
(* I.e. to import hello_world.meow: GIMME HELLO_WORLD? *)
let valid_import_name import =
  let accepted_regex = Str.regexp "['A-Z']['A-Z', '_', '0-9']*" in
  if Str.string_match accepted_regex import 0 then true else false

(* Convert string representing module into the module path *)
let mangle_import_name import =
  if valid_import_name import then
    let cwd = Sys.getcwd ()
    and adjusted_name = String.lowercase_ascii import
    in Printf.sprintf "%s/%s.meow" cwd adjusted_name
  else
    let msg = "illegal import name " ^ import
    in raise (ImportNotFound (msg))

let import_ast import_path =
  let channel_in = open_in import_path in
  let lexbuf = Lexing.from_channel channel_in in
  let ast = Parser.program Scanner.token lexbuf in
  Hashtbl.add imported_asts import_path ast; ast

(* Pull in the import; this reads in the file, parses into AST
   and adds the file to the *)

let rec do_import import =
  (* get the import as a path*)
  let real_import_path = mangle_import_name import in

  (* if we've already pulled in the import, stop recursion here *)
  if Hashtbl.mem imported_asts real_import_path then ()
  else (
    (* if it exists, get the ast; then recursively get other ASTs *)
    check_import_exists real_import_path;
```

```
    let new_ast = import_ast real_import_path in
    match new_ast with
      ([], _, _) -> ()
    | (import_list, _, _) ->
        if unique_list import_list then
          List.iter do_import import_list
        else ()
  )

(*********************************************)
(* Add imports recursively to main file      *)
(*********************************************)
let add_imports (imports, functions, classes) filename =
  (* populate hash table of asts that we need to import *)
  let base_path =
    Printf.sprintf "%s/%s" (Sys.getcwd ()) filename
  in Hashtbl.add imported_asts base_path ([], [], []);

  if unique_list imports then (
    List.iter do_import imports;
    (* add all the new classes and functions to the list of existing ones *)
    let functions' =
      Hashtbl.fold (fun _ (_, funcs, _) acc ->
          List.fold_left (fun acc func -> func :: acc) acc funcs
      ) imported_asts functions

    and classes' =
      Hashtbl.fold (fun _ (_, _, cls_list) acc ->
        List.fold_left (fun acc cls -> cls :: acc) acc cls_list
      ) imported_asts classes
    in
    ([], functions', classes')
  )
  else ([], [], []) (* make compiler happy; an exception will be raised *)
```

Listing 37: src/meowlang.ml

```
(*
    Main CLI for running the compiler. Command line arguments specify behavior.
*)
open Pretty
type action = Ast | Sast | LLVM_IR | Compile
let filename = ref ""
let _ =
    (* Figure out the action to take *)
    let action = ref Ast in
    let set_action a () = action := a in
    let set_filename name = filename := name in
```

```
    let speclist = [
        ("-a", Arg.Unit (set_action Ast), "Print the AST");
        ("-s", Arg.Unit (set_action Sast), "Perform Semantic Checks on the SAST");
        ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated LLVM IR");
        ("-c", Arg.Unit (set_action Compile), "Check and print the generated LLVM IR");
        ("-f", Arg.String (set_filename), "Submit file to compile")
    ] in
    let usage_msg = "usage: ./meowlang.native [-a|-s|-l|-c] -f [file.meow]" in
    Arg.parse speclist print_endline usage_msg;

    (* Open the specified file *)
    let channel_in = open_in !filename in
    let lexbuf = Lexing.from_channel channel_in in
    let ast = Parser.program Scanner.token lexbuf in
    let ast_with_imports = Import.add_imports ast !filename in

    (* Perform action based on specification *)
    match !action with
      Ast -> print_string (string_of_program ast_with_imports)
    | _ ->
        let sast = Semant.check ast_with_imports in
        match !action with
          Ast     -> ()
        | Sast    -> print_string "Semantic check succeded!\n"
        | LLVM_IR -> print_string (Llvm.string_of_llmodule (Codegen.translate sast))
        | Compile -> let m = Codegen.translate sast in
            Llvm_analysis.assert_valid_module m;
            print_string (Llvm.string_of_llmodule m)
```

Listing 38: src/parser.mly

```
%{
open Ast
%}

%token RETURN MODULE IMPORT CALL FUNCTION DEF COMP CLASS NEW FREE MAKE HERE SIZE
%token SEMI LPAREN RPAREN LBRACE RBRACE COMMA PLUS MINUS TIMES DIVIDE ASSIGN LBRACKET RBRACKET
%token NOT EQ NEQ LT GT AND OR CONCAT CONTAINS IN
%token IF THEN ELSE FOR INCREMENT DECREMENT INT BOOL FLOAT STRING ARRAY
%token <int> ILIT
%token <bool> BLIT
%token <string> ID FLIT SLIT
%token EOF

%start program
%type <Ast.program> program

%nonassoc NOELSE
```

```
%nonassoc ELSE
%nonassoc ARRAY
%right ASSIGN
%right CONCAT
%right OR AND
%right EQ NEQ
%right LT GT
%right PLUS MINUS
%right TIMES DIVIDE
%right NOT

%%

program:
  decls EOF                   { $1 }

decls:
    imports udf_udcs          { (List.rev $1, fst $2, snd $2) }
  | udf_udcs                  { ([], fst $1, snd $1)          }

imports:
    import                   { [$1] }
  | imports import           { $2 :: $1 }

import:
  MODULE ID IMPORT           { $2 }

udf_udcs:
   /* nothing */             { ([], [])                 }
| udf_udcs fdecl             { (($2 :: fst $1), snd $1) }
| udf_udcs cdecl             { (fst $1, ($2 :: snd $1)) }

fdecls:
    fdecl                    { [$1]     }
  | fdecls fdecl             { $2 :: $1 }

fdecl:
    LBRACE DEF return_type FUNCTION ID formals_opt RPAREN vdecls stmt_list RBRACE
      {
        {
          typ = $3;
          fname = $5;
          formals = List.rev $6;
          locals = List.rev $8;
          body = List.rev $9;
        }
      }

```

```
return_type:
    /* nothing */            { Void }
  | typ                      { $1   }

formals_opt:
    /* nothing */            { [] }
  | LPAREN formal_list       { $2 }

formal_list:
    typ ID                   { [($1,$2)]      }
  | formal_list COMMA typ ID { ($3,$4) :: $1  }

typ:
    INT                      { Int                              }
  | BOOL                     { Bool                             }
  | FLOAT                    { Float                            }
  | STRING                   { String                           }
  | ID                       { Obtype ($1)                      }
  /* This is to support arrays as arguments to functions; -1 is just a 'dummy' value,
     since ocaml doesn't allow null for integer values */
  | ARRAY CONTAINS typ       { Arrtype (ILiteralArraySize(-1), $3) }

vdecls:
    /* nothing */             { []       }
  | vdecls vdecl              { $2 :: $1 }

vdecl:
    DEF typ ID SEMI                      { ($2, $3, Noexpr) }
  | DEF typ ID ASSIGN expr SEMI          { ($2, $3, $5)     }
  | DEF typ ID ASSIGN function_call SEMI { ($2, $3, $5)     }

stmt_list:
    /* nothing */             { []        }
  | stmt_list stmt            { $2 :: $1  }

stmt:
    expr SEMI                                  { Expr($1)           }
  | function_call SEMI                         { Expr($1)           }
  | RETURN expr SEMI                           { Return($2)         }
  | LBRACE stmt_list RBRACE                    { Block(List.rev $2) }
  | array_decl SEMI                            { Expr($1)           }
  | c_instance SEMI                            { Expr($1)           }
  | expr IF THEN LBRACE stmt_list RBRACE %prec NOELSE
      { If($1, Block(List.rev $5), Block([]))          }
  | expr IF THEN LBRACE stmt_list RBRACE ELSE LBRACE stmt_list RBRACE
      { If($1, Block(List.rev $5), Block(List.rev $9)) }
  | FOR expr INCREMENT expr_opt COMMA expr LBRACE stmt_list RBRACE
      { For(Increment, $2, $4, $6, Block(List.rev $8)) }
```

```
114   | FOR expr DECREMENT expr_opt COMMA expr LBRACE stmt_list RBRACE
115       { For(Decrement, $2, $4, $6, Block(List.rev $8)) }
116   | ID IN ID ASSIGN expr SEMI                    { ClassAssign(Id($3), $1, $5)}
117   | ID LBRACKET expr RBRACKET ASSIGN expr SEMI    { ArrayAssign($1, $3, $6)    }
118   | FREE ID SEMI                                 { Dealloc(Id($2))            }
119   | ID ASSIGN function_call SEMI                  { Expr(Assign(Id($1), $3))   }
120
121 expr:
122     ILIT                    { ILiteral($1)               }
123   | FLIT                    { Fliteral($1)               }
124   | BLIT                    { BoolLit($1)                }
125   | SLIT                    { StringLit($1)              }
126   | ID                      { Id($1)                     }
127   | ID ASSIGN typ expr      { Assign(Id($1), Cast($3, $4))}
128   | ID ASSIGN expr          { Assign(Id($1), $3)         }
129   | PLUS expr COMMA expr    { Binop($2, Add,   $4)       }
130   | MINUS expr COMMA expr   { Binop($2, Sub,   $4)       }
131   | TIMES expr COMMA expr   { Binop($2, Mult,  $4)       }
132   | DIVIDE expr COMMA expr  { Binop($2, Div,   $4)       }
133   | EQ expr COMMA expr      { Binop($2, Equal, $4)       }
134   | NEQ expr COMMA expr     { Binop($2, Neq,   $4)       }
135   | LT expr COMP expr       { Binop($2, Less,  $4)       }
136   | GT expr COMP expr       { Binop($2, Greater, $4)     }
137   | AND expr COMMA expr     { Binop($2, And,   $4)       }
138   | OR expr COMMA expr      { Binop($2, Or,    $4)       }
139   | NOT expr                { Unop(Not, $2)              }
140   | LPAREN expr RPAREN      { $2                         }
141   | CONCAT expr COMMA expr  { Binop($2, Concat, $4)      }
142   | ID IN ID                { ClassAccess(Id($3), $1)    }
143   | ID LBRACKET expr RBRACKET { ArrayAccess($1, $3)       }
144
145  function_call:
146     CALL ID                         { FunctionCall($2, [])          }
147   | CALL ID IN expr                 { MethodCall($4, $2, [])        }
148   | CALL ID IN HERE                 { MethodCall(Id("this"), $2, []) }
149   | CALL ID LPAREN args_opt         { FunctionCall($2, $4)          }
150   | CALL ID IN ID LPAREN args_opt   { MethodCall(Id($4), $2, $6)    }
151   | CALL ID IN HERE LPAREN args_opt { MethodCall(Id("this"), $2, $6) }
152
153 expr_opt:
154   /* nothing */             { Noexpr }
155   | expr                    { $1     }
156
157 args_opt:
158   | args_list               { List.rev $1 }
159
160 args_list:
161     expr                    { [$1]     }
```

```
  | args_list COMMA expr      { $3 :: $1 }

/* Array Specification */

array_decl:
    MAKE ID NEW ARRAY CONTAINS typ SIZE array_size_typ RPAREN LPAREN args_opt { NewArray($2, $6, $8, $11)}
  | MAKE ID NEW ARRAY CONTAINS typ SIZE array_size_typ                        { NewArray($2, $6, $8, []) }

array_size_typ:
    ILIT                    { ILiteralArraySize($1) }
  | ID                      { VariableArraySize($1) }

/* Classes */

cdecl:
    LBRACE DEF CLASS ID RPAREN vdecls methods RBRACE
    {
      {
        cname = $4;
        cvars = $6;
        cfuncs = $7;
      }
    }

methods:
    /* nothing */           { [] }
  | fdecls                  { $1 }

 /* Class Instantiation */

c_instance:
    MAKE ID NEW typ                 { NewInstance($2, $4, []) }
  | MAKE ID NEW typ RPAREN class_opt { NewInstance($2, $4, $6) }

class_opt:
    /* nothing */           { [] }
  | LPAREN copt_list        { $2 }

copt_list:
    expr                    { [$1]     }
  | copt_list COMMA expr    { $3 :: $1 }
```

Listing 39: src/pretty.ml

```
(*
   Pretty-printing functions that allow you to print a meowlang program in the
   near-equivalent C program. Note that a large amount of this code was derived
   and adjusted from the MicroC Compiler provided by S. Edwards (Spring 2021).
```

```
*)
open Ast

let string_of_op = function
    Add -> "+"
  | Sub -> "-"
  | Mult -> "*"
  | Div -> "/"
  | Equal -> "=="
  | Neq -> "!="
  | Less -> "<"
  | Greater -> ">"
  | And -> "&&"
  | Or -> "||"
  | Concat -> "+"
  | Increment -> "++"
  | Decrement -> "--"

let string_of_modules = function
  l -> "include \"" ^ l ^ "\""

let string_of_uop = function
  Not -> "!"

let string_of_array_size = function
  ILiteralArraySize i -> string_of_int i
| VariableArraySize s -> s

let rec string_of_typ = function
  Int -> "int"
| Bool -> "bool"
| Float -> "float"
| String -> "char *"
| Void -> ""
| Obtype(s) -> "class " ^s
| Arrtype(size, typ) ->
    match size with
      ILiteralArraySize i when i = -1 -> string_of_typ typ ^ " []"
    | _ -> string_of_typ typ ^ " [" ^ string_of_array_size size ^ "]"

let string_of_array_size = function
  ILiteralArraySize(l) -> string_of_int l
| VariableArraySize(s) -> s

let rec string_of_expr = function
    ILiteral(l) -> string_of_int l
  | StringLit(l) -> "\"" ^ l ^ "\""
  | Fliteral(l) -> l
```

```
53    | BoolLit(true) -> "true"
54    | BoolLit(false) -> "false"
55    | Cast(typ, e) -> "(" ^ string_of_typ typ ^ ") " ^ string_of_expr e
56    | Id(s) -> s
57    | Binop(e1, o, e2) ->
58        string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
59    | Unop(o, e) -> string_of_uop o ^ string_of_expr e
60    | Assign(v, e) -> string_of_expr v ^ " = " ^ string_of_expr e
61    | FunctionCall(f, el) ->
62        (match f with
63            "Meow" -> "printf" ^ "(\"%X\\n\", " ^ String.concat ", " (List.map string_of_expr el) ^ ")"
64          | _      -> f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")")
65    | MethodCall(ob, f, el) ->
66      string_of_expr ob ^ "." ^ f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
67    | NewArray(i, typ, s, contents) ->
68        string_of_typ typ ^ " [" ^ string_of_array_size s ^ "] " ^ i ^ " = [ " ^
69        String.concat ", " (List.map string_of_expr contents) ^ " ]"
70    | Noexpr -> ""
71    | NewInstance(var, c, []) -> string_of_typ c ^ " " ^ var
72    | NewInstance(var, c, exprs) -> string_of_typ c ^ " " ^ var ^ "(" ^
73        String.concat "" (List.map (fun e -> string_of_expr e ^ ", ") exprs) ^ ")"
74    | ClassAccess(ob, el) -> string_of_expr ob ^ "." ^ el
75    | ArrayAccess(var, e) -> var ^ "[" ^ string_of_expr e ^ "]"
76
77  let rec string_of_stmt = function
78      Expr(expr) -> "\t" ^ string_of_expr expr ^ ";\n"
79    | Return(expr) -> "\treturn " ^ string_of_expr expr ^ ";\n"
80    | Block(stmts) ->
81      "\t{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "\t}\n"
82    | If(e, s, Block([])) -> "\tif (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
83    | If(e, s1, s2) ->  "\tif (" ^ string_of_expr e ^ ")" ^
84      string_of_stmt s1 ^ "\telse\t" ^ string_of_stmt s2
85    | For(o, e1, e_opt, e2, s) ->
86        "\tfor (" ^string_of_expr e_opt ^ " " ^ string_of_expr e1 ^ string_of_op o ^
87        " " ^ string_of_expr e2 ^ ") {\n\t\t" ^ string_of_stmt s ^ "\t}\n"
88    | Dealloc(e) -> "\tfree(" ^ string_of_expr e ^ ");\n"
89    | ClassAssign(e1, s2, e2) -> "\t" ^ string_of_expr e1 ^ "." ^ s2 ^ " = " ^ string_of_expr e2 ^ ";\n"
90    | ArrayAssign(s, e1, e2) -> "\t" ^ s ^ "[" ^ string_of_expr e1 ^ "] = " ^ string_of_expr e2 ^";\n"
91
92  let string_of_vdecl (t, id, expr) =
93      match expr with
94      | Noexpr -> "\t" ^ string_of_typ t ^ " " ^ id ^ ";\n"
95      | _ -> "\t" ^ string_of_typ t ^ " " ^ id ^ " = " ^ string_of_expr expr ^ ";\n"
96
97  let string_of_formals (t, id) = string_of_typ t ^ " " ^ id
98
99  let format_params fdecl =
100   "(" ^ String.concat ", " (List.map string_of_formals fdecl.formals) ^ ")\n"
```

```
101
102 let string_of_fdecl fdecl  =
103   let return_string = (
104     match fdecl.typ with
105       Void -> ""
106     | _ -> string_of_typ fdecl.typ ^ " ") in
107
108   return_string ^ fdecl.fname ^ format_params fdecl ^
109   "{\n" ^
110     String.concat "" (List.map string_of_vdecl fdecl.locals) ^
111     String.concat "" (List.map string_of_stmt fdecl.body) ^
112   "}\n"
113
114 let string_of_mdecl fdecl =
115   let indent = "\t" in
116
117   let return_string = (
118     match fdecl.typ with
119         Void -> ""
120       | _ -> string_of_typ fdecl.typ ^ " ") in
121
122   indent ^ return_string ^ fdecl.fname ^ format_params fdecl ^ indent ^ "{\n" ^
123   String.concat indent (List.map string_of_vdecl fdecl.locals) ^
124   String.concat indent (List.map string_of_stmt fdecl.body) ^ indent ^ "}\n"
125
126 let string_of_cdecl cdecl =
127     "Class " ^ cdecl.cname ^ " {\n\n" ^
128     String.concat "" (List.map string_of_vdecl cdecl.cvars) ^ "\n" ^
129     String.concat "" (List.map string_of_mdecl cdecl.cfuncs) ^ "\n" ^
130     "}\n"
131
132 (* Main function that pretty-prints an AST *)
133 let string_of_program (imports, funcs, classes) =
134   String.concat "" (List.map string_of_modules imports) ^ "\n\n" ^
135   String.concat "\n" (List.map string_of_fdecl funcs) ^ "\n" ^
136   String.concat "\n" (List.map string_of_cdecl classes)
```

Listing 40: `src/sast.ml`

```
1 (* Semantically Checked AST *)
2 open Ast
3
4 type sexpr = typ * sx
5 and sx =
6     SILiteral of int
7   | SFliteral of string
8   | SBoolLit of bool
9   | SStringLit of string
```

```
  | SCast of typ * sexpr
  | SId of string
  | SBinop of sexpr * op * sexpr
  | SUnop of uop * sexpr
  | SAssign of sexpr * sexpr
  | SFunctionCall of string * sexpr list
  | SNoexpr
  | SNewInstance of string * typ * sexpr list
  | SClassAccess of typ * sexpr * string
  | SNewArray of string * typ * array_size * sexpr list
  | SArrayAccess of string * sexpr

  type sbind_var = typ * string * sexpr

type sstmt =
    SBlock of sstmt list
  | SExpr of sexpr
  | SReturn of sexpr
  | SIf of sexpr * sstmt * sstmt
  | SFor of op * sexpr * sexpr * sexpr * sstmt
  | SDealloc of sexpr
  | SClassAssign of typ * sexpr * string * sexpr
  | SArrayAssign of string * sexpr * sexpr

type sfunc_decl = {
    styp : typ;
    sfname : string;
    sformals : bind_formals list;
    slocals : bind_var list; (* this is still a bind_var because exprs get moved to body *)
    sbody : sstmt list;
  }

type sclass_decl = {
    scname : string;
    scvars : sbind_var list;
    scfuncs : sfunc_decl list;
}

type program = import list * sfunc_decl list * sclass_decl list
```

Listing 41: `src/scanner.mll`

```
(* Ocamllex scanner for MicroC

Support for strings through read_string taken from OCaml documentation
found https://dev.realworldocaml.org/parsing-with-ocamllex-and-menhir.html
*)
{
```

```
7     open Parser
8
9     exception SyntaxError of string
10 }
11
12 let digit = ['0' - '9']
13 let digits = digit+
14 let identifier = ['a'-'z' 'A'-'Z'] ['a'-'z' 'A'-'Z' '0'-'9' '_']*
15 let whitespace = [' ' '\t' '\r' '\n']
16
17 (*
18   Special rule for float; note that because periods are used as a semicolon
19   in this langauge, we have to treat the input "2." differently than usual. In
20   this language, "2." is interpreted as an integer and "2.0" is a float.
21 *)
22 let float = digits '.'  ((digit+ | ( ['e' 'E'] ['+' '-']? digits)) | (digit* ( ['e' 'E'] ['+' '-']? digits )))
23
24 rule token = parse
25 (* Whitespace/Comments *)
26   whitespace      { token lexbuf }
27 | "PSST"          { comment lexbuf }
28
29 (* Structural *)
30 | "HAI"           { LBRACE }
31 | "KBYE"          { RBRACE }
32 | "WIT"           { LPAREN }
33 | ","             { RPAREN }
34 | "AN"            { COMMA }
35 | "."             { SEMI }
36
37 (* Module/Imports *)
38 | "GIMME"         { MODULE }
39 | "?"             { IMPORT }
40
41 (* Objects/Classes *)
42 | "MAEK"          { MAKE }
43 | "NEW"           { NEW }
44 | "BLEEP"         { FREE }
45 | "CLASS"         { CLASS }
46 | "IN"            { IN }
47 | "HERE"          { HERE }
48
49 (* Functions *)
50 | "PURR"          { CALL }
51 | "FUNC"          { FUNCTION }
52 | "GIVE"          { RETURN }
53
54 (* Arrays/Indexing *)
```

```
| "["             { LBRACKET }
| "]"             { RBRACKET }
| "HOLDS"         { SIZE     }

(* Operators *)
| "SUM OF"        { PLUS }
| "DIFF OF"       { MINUS }
| "PRODUKT OF"    { TIMES }
| "QUOSHUNT OF"   { DIVIDE }
| "ITZ ME"        { DEF }
| "IZ"            { ASSIGN }
| "SAEM"          { EQ }
| "DIFFRINT"      { NEQ }
| "SMALLR"        { LT }
| "BIGGR"         { GT }
| "BOTH OF"       { AND }
| "EITHER OF"     { OR }
| "NOT"           { NOT }
| "CAT"           { CONCAT }
| "THAN"          { COMP }
| "OF"            { CONTAINS }
| "UPPIN"         { INCREMENT }
| "NERFIN"        { DECREMENT }

(* Flow Control *)
| "O RLY?"        { IF }
| "YA RLY"        { THEN }
| "NO WAI"        { ELSE }
| "IM IN YR LOOP" { FOR }

(* Data Types *)
| "YARN"          { STRING }
| "NUMBR"         { INT }
| "BOO"           { BOOL }
| "NUMBAR"        { FLOAT }
| "AYE"           { BLIT(true)  }
| "NAY"           { BLIT(false) }
| "BUCKET"        { ARRAY }
| digits as lxm   { ILIT(int_of_string lxm) }
| float as lxm    { FLIT(lxm) }
| identifier as lxm { ID(lxm) }
| '"'             { read_string (Buffer.create 17) lexbuf }  (* String *)
| eof { EOF }
| _ as char { raise (SyntaxError("Illegal character: '" ^ Char.escaped char ^ "'")) }

and read_string buf =
  parse
  | '"'           { SLIT(Buffer.contents buf) }
```

```
  | [^ '"']+
    {
        Buffer.add_string buf (Lexing.lexeme lexbuf);
        read_string buf lexbuf
    }
  | eof { raise (SyntaxError ("String is not terminated")) }

and comment = parse
  "\n" { token lexbuf }
| _    { comment lexbuf }
```

Listing 42: src/semant.ml

```
(*****************************************************************************)
(* Performs semantic checks on the AST, producing a new SAST                 *)
(* Also responsible for converting classes into structs and functions that   *)
(* are more easily processed in the LLVM step (i.e., call site adjustments   *)
(* and lifing methods to global space with new struct arguments)             *)
(*****************************************************************************)
open Exceptions
open Ast
open Sast
open Pretty

module StringMap = Map.Make(String)

type environment = {
  in_class : bool;
  class_name: string;
  constructor: bool;
  obj_name: string; (* empty string if constructor if false *)
  mutable function_name: string;
  mutable returns: typ;
  symbols : (string, Ast.typ) Hashtbl.t;
}

(* Hashtable of valid functions and classes to be used globally *)
let function_tbl:(string, Ast.func_decl) Hashtbl.t = Hashtbl.create 10
let class_tbl:(string, Ast.class_decl) Hashtbl.t = Hashtbl.create 10

(* Return function by name; Raise exception if it doesn't exist *)
let find_function fname =
  try Hashtbl.find function_tbl fname
  with Not_found -> raise (FunctionNotFound (undeclared_msg ^ fname))

(* Return class by name; Raise exception if it doesn't exist *)
let find_class cname =
  try Hashtbl.find class_tbl cname
```

```
36    with Not_found -> raise (ClassNotFound (undeclared_msg ^ cname))
37
38  (* Return variable by name; Raise exception if it doesn't exist *)
39    let find_variable tbl vname =
40      try Hashtbl.find tbl vname
41      with Not_found -> raise (VariableNotFound (undeclared_msg ^ vname))
42
43  (* Takes a class name to produce the variable name given to objects
44     when passed as parameters to method flipped into functions. *)
45  let mangled_obj_varname cname =
46    String.lowercase_ascii (cname ^ "*")
47
48  (* Converts a method name to a function name *)
49  let m_to_f_name cls_n m_n =
50    cls_n ^ "." ^ m_n
51
52  (* Raise an exception if the given types are not the same *)
53  let check_matching_types typ1 typ2 err =
54    (* with arrays we don't care if the sizes match; they're just implemented as pointers *)
55    match typ1 with
56      Arrtype(_, arr_typ_1) ->
57        (match typ2 with
58          Arrtype(_, arr_typ_2) when arr_typ_1 = arr_typ_2 -> typ1
59        | _ -> raise err)
60    | _ when typ1 = typ2 -> typ1
61    | _ -> raise err
62
63  (* Helper function to check for duplicates of anything *)
64  let find_duplicate items exception_msg =
65    let rec helper = function
66        n1 :: n2 :: _ when n1 = n2 ->
67          let msg = Printf.sprintf "%s %s" exception_msg n1
68          in raise (DuplicateIdentifier (msg))
69      | _ :: t -> helper t
70      | [] -> ()
71    in helper (List.sort compare items)
72
73  (* Find the type of something, given a symbol table *)
74  let find_type_of_id symbol_tbl id =
75    try Hashtbl.find symbol_tbl id
76    with Not_found ->
77      let msg = Printf.sprintf "%s %s" undeclared_msg id
78      in raise (VariableNotFound (msg))
79
80  (*Find the class method by method name and class type *)
81  let find_class_method cname mname =
82    let cls = find_class cname in
83    let cls_methods = List.fold_left (
```

```
84        fun m cls_method -> StringMap.add cls_method.fname cls_method m
85      ) StringMap.empty (cls.cfuncs)
86    in
87    try StringMap.find mname cls_methods
88    with Not_found ->
89      let msg = Printf.sprintf "%s %s" class_method_unknown (m_to_f_name cname mname)
90      in raise (ClassMethodNotFound(msg))
91
92  let instance_variables_of_cls cls_name =
93    let cls = find_class cls_name in
94    List.fold_left (fun acc (_, name, _) -> name  :: acc) [] cls.cvars
95
96  (*** Checks for duplicates ****)
97  let check_duplicates functions classes =
98    (* duplicates in function names *)
99    find_duplicate (List.map (fun f -> f.fname) functions) dup_func_msg;
100   (* duplicates in class names*)
101   find_duplicate (List.map (fun c -> c.cname) classes) dup_class_msg;
102   (* duplicates in methods within a class*)
103   List.iter (
104     fun cls ->
105       find_duplicate (List.map (fun m -> m.fname) cls.cfuncs) dup_method_msg
106   ) classes;
107   ()
108
109 (* Add built in functions to the list of functions *)
110 let add_built_ins existing_funcs =
111   (*  printf function, corresponding to Meow in Meowlang *)
112   let printf = {
113     typ = Void;
114     fname = "Meow";
115     formals = [(String, "x")];
116     locals = [];
117     body = []
118   }
119   and custom_scanf = {
120     typ = Int;
121     fname = "Scan";
122     formals = [(String, "x")];
123     locals = [];
124     body = []
125   }
126   in
127   let e_funs = printf :: existing_funcs in
128   custom_scanf :: e_funs
129
130 (* return boolean value to represent if return type is expected for a function *)
131 let has_return_value func =
```

```
132   match func.typ with
133     Void -> false
134   | _ -> true
135
136
137 (*************************************)
138 (* Main function for checking semantics *)
139 (* of expressions                       *)
140 (*************************************)
141 let rec semant_expr expr env =
142
143   (* Checks that argument types match formal var types *)
144   let check_arg_type formal_param arg_expr =
145     let (actual_type, arg_expr') = semant_expr arg_expr env
146     and expected_type = fst formal_param in
147     let msg = Printf.sprintf "%s, expected typ %s, but got %s"
148         (string_of_expr (arg_expr)) (string_of_typ expected_type) (string_of_typ actual_type)
149     in
150     ignore(check_matching_types expected_type actual_type (ArgumentTypeMismatch(msg)));
151     (actual_type, arg_expr')
152   in
153
154   match expr with
155     ILiteral i  -> (Int, SILiteral i)
156   | Fliteral f  -> (Float, SFliteral f)
157   | BoolLit b   -> (Bool, SBoolLit b)
158   | StringLit s -> (String, SStringLit s)
159   | Noexpr      -> (Void, SNoexpr)
160
161   | Cast(typ, e) as ex ->
162     let (typ1, e') = semant_expr e env in
163     let _ =
164       match typ with
165         Int when (typ1 = Float || typ1 = String) -> ()
166       | Float when (typ1 = Int || typ1 = String) -> ()
167       | String when (typ1 = Int || typ1 = Float) -> ()
168       | _ when typ = typ1 && (typ1 = Int || typ1 = Float) ->
169           raise (CastUnnecessary("cast is redundant here: " ^ string_of_expr ex))
170       | _  ->
171         let msg =  Printf.sprintf "cast not currently supported from %s to %s. See: %s"
172                   (string_of_typ typ1) (string_of_typ typ) (string_of_expr ex)
173         in raise(NotYetSupported(msg))
174     in (typ, SCast(typ, (typ1, e')))
175
176   | Binop (e1, op, e2) as ex ->
177     (* Binary operations work with operands of the same type *)
178     let (typ1, e1') = semant_expr e1 env
179     and (typ2, e2') = semant_expr e2 env in
```

```
180     let same_type = typ1 = typ2 in
181     let end_typ = match op with
182         Add | Sub | Mult | Div | Increment | Decrement when same_type && typ1 = Int -> Int
183       | Add | Sub | Mult | Div when (typ1 = Float || typ1 = Int) && (typ2 = Float || typ2 = Int) -> Float
184       | Equal | Neq when same_type && (typ1 = Float || typ1 = Int || typ1 = String || typ1 = Bool) -> Bool
185       | Equal | Neq when (typ1 = Float || typ1 = Int) && (typ2 = Float || typ2 = Int) -> Bool
186       | Less | Greater when (typ1 = Float || typ1 = Int) && (typ2 = Float || typ2 = Int) -> Bool
187       | And | Or when same_type && typ1 = Bool -> Bool
188       | Concat when (typ1 = String && (typ2 = String || typ2 == Int || typ2 == Float)) ||
189                     (typ2 = String && (typ1 == Int || typ1 == Float)) -> String
190       | _  ->
191         let msg = Printf.sprintf "unexpected types in binary op (%s and %s): %s"
192               (string_of_typ typ1)  (string_of_typ typ2) (string_of_expr ex)
193         in raise (IllegalBinaryOp(msg))
194     in (end_typ, SBinop((typ1, e1'), op, (typ2, e2')))
195
196   | Unop (op, e) as ex ->
197     (* Only one type of Unop supported *)
198     let (vtype, e') = semant_expr e env in
199     if vtype == Bool then (vtype, SUnop(op, (vtype, e')))
200     else raise (IllegalUnaryOp (string_of_expr ex))
201
202   | Id id ->
203     (* if we are in a class, adjust the ID to be a class access, of a "<ClassName>*" parameter *)
204     let typ = find_type_of_id env.symbols id in
205       if env.in_class then
206         let mangled_name =
207           if env.constructor then env.obj_name
208           else mangled_obj_varname env.class_name
209         in
210         if List.mem id (instance_variables_of_cls env.class_name) then
211           (typ, SClassAccess(Obtype(env.class_name), (typ, SId(mangled_name)), id))
212         else (typ, SId id)
213       else (typ, SId id)
214
215   | Assign (v, e) as ex ->
216     (* Check that the expr produces the same type as the variable it is assigned to *)
217     let _ =
218       (match v with
219         Id (id) -> id
220       | _ -> raise (VariableAssignmentError("assignment is performed on a variable")))
221     in
222     let (var_typ, id) = semant_expr v env
223     and (ret_type, e') = semant_expr e env in
224     let err =
225       let msg = Printf.sprintf "%s. expected %s, got %s here: %s"
226             assignment_typ_mismatch (string_of_typ var_typ) (string_of_typ ret_type) (string_of_expr ex)
227       in VariableAssignmentError(msg)
```

```
228     in (check_matching_types var_typ ret_type err, SAssign((var_typ, id), (ret_type, e')))
229
230   | FunctionCall (fname, args) ->
231     (* 1. Make sure function exists *)
232     let func = find_function fname in
233
234     (* 2. Check that param length is equal to the num args provided *)
235     if List.length args != List.length func.formals then
236       raise (FunctionArgumentLengthMismatch (func_arg_num_mismatch ^ fname))
237     else
238     (* 3. Check that the arguments passed are of the expected type *)
239       let args' =
240         (match fname with
241         (* this is a work around for handling the built in print function, which can accept multiple types *)
242         | "Meow" ->
243           (match args with
244             arg :: []  -> [semant_expr arg env]
245           | _ -> raise (FunctionArgumentLengthMismatch("built in string function takes 1 argument only\n")))
246         | "Scan" ->
247           (match args with
248             arg :: []  ->
249             let (typ, arg') = semant_expr arg env in
250             (match arg with
251               Id(_) when typ = String -> [(typ, arg')]
252             | _ -> raise (ArgumentTypeMismatch(scan_error)))
253           | _ -> raise (FunctionArgumentLengthMismatch(scan_error)))
254         | _ -> List.map2 check_arg_type func.formals args)
255       in
256       (func.typ, SFunctionCall(fname, args'))
257
258   | MethodCall (obj_expr, meth_name, args) as ex ->
259     (* 1. Check that the object exists in the symbol table  *)
260     let (v_type, obj_expr') =
261       match obj_expr with
262         Id("this") ->
263           if env.in_class = false then
264             let msg = Printf.sprintf "%s, but found in func %s"
265                 use_of_this_outside_class env.function_name
266             in raise (InvalidMethodCall(msg))
267           else (Obtype(env.class_name), SId(mangled_obj_varname env.class_name))
268       (* call method on variable for object, or array element that is object only *)
269       | Id(_) | ArrayAccess(_, _) -> semant_expr obj_expr env
270       | _ ->
271           let msg = Printf.sprintf "%s, found %s"
272             method_call_expr (string_of_expr ex)
273           in raise(InvalidMethodCall(msg))
274     in
275
```

```
276     (* 2. Check that the method exists within the class and get it *)
277     let cname =
278       match v_type with
279       | Obtype object_type -> object_type
280       | _ ->
281           let msg = Printf.sprintf "%s found %s instead of Objtype in %s"
282             invalid_method_call (string_of_typ v_type) (string_of_expr ex)
283           in raise (InvalidMethodCall(msg))
284     in
285     let meth = find_class_method cname meth_name in
286
287     (* 3. Check that param length is equal to the num args provided *)
288     if List.length args != List.length meth.formals then
289       let msg = Printf.sprintf "%s %s (got %s, expected %s)"
290             meth_arg_num_mismatch  meth_name (string_of_int (List.length args))
291             (string_of_int (List.length meth.formals))
292       in raise (MethodArgumentLengthMismatch(msg))
293     else
294     (* 4. Check that the arguments passed are of the expected type *)
295       let args' =
296         try List.map2 check_arg_type meth.formals args
297         with ArgumentTypeMismatch(s) ->
298           let msg = Printf.sprintf "method %s received arg of unexpected type: %s" meth_name s
299           in raise(ArgumentTypeMismatch(msg))
300       in
301       (* 5. Convert to function call; add the object from step 1 as the first argument *)
302       (meth.typ, SFunctionCall(m_to_f_name cname meth_name, (v_type, obj_expr') :: args'))
303
304   | NewArray (arr_name, arr_typ, arr_size, expr_list) as ex ->
305     (* 1. Check to make sure that size is integer *)
306     let array_size_typ =
307       match arr_size with
308         ILiteralArraySize i ->
309           (* 2. check integer literal against expr_list length *)
310           let num_items = List.length expr_list in
311             if num_items > i then
312               raise (ExcessArrayInput(excess_array_item ^ " " ^ string_of_int num_items))
313             else if i < 1 then
314               raise (InvalidArraySizeSpecified("size of array must be integer > 0"))
315             else (); Int
316       | VariableArraySize s -> find_type_of_id env.symbols s
317     in
318     if array_size_typ != Int then
319       raise (InvalidArraySizeSpecified(invalid_array_size_msg ^ string_of_expr ex ))
320     else
321       let expr_list' = List.fold_left (fun acc e -> semant_expr e env :: acc) [] expr_list
322       (* 3. if expr_list, check that the expressions match the content type of array *)
323       (* This is also a good opportunity to "massage" these array expressions into   *)
```

```
324       (* assignment statements after the array is allocated *)
325       and massage_array_args idx (expr_typ, expr) =
326         let msg = Printf.sprintf "%s, expected: %s, got: %s in %s"
327           invalid_array_item_msg (string_of_typ arr_typ) (string_of_typ expr_typ) (string_of_expr ex)
328         and result =
329           (expr_typ, SAssign((arr_typ, SArrayAccess(arr_name, (Int, SILiteral(idx)))), (expr_typ, expr)))
330         in
331         match expr_typ with
332           Obtype(_) when expr_typ == arr_typ -> result
333         | _ when expr_typ = arr_typ -> result
334         | _ -> raise (InvalidArrayItem(msg))
335       in
336       let array_constructor = List.mapi massage_array_args (List.rev expr_list') in
337
338       (* make sure to add the allocated obj to symbol table so it can be referenced *)
339       Hashtbl.add env.symbols arr_name (Arrtype(arr_size, arr_typ));
340       (Arrtype(arr_size, arr_typ), SNewArray(arr_name, arr_typ, arr_size, array_constructor))
341
342   | NewInstance (obj_name, typ, expr_list) as ex ->
343     (* 1. You can only create an "instance" of something that is type Objtype *)
344     let (cname, cls) =
345       match typ with
346         (* 2. Check that the class of new instance actually exists - valid *)
347         Obtype o -> (o, find_class o)
348       | _ ->
349           let msg = Printf.sprintf "%s, found: %s" invalid_object_creation (string_of_expr ex)
350           in raise (ObjectCreationInvalid(msg))
351     in
352
353     (*****************************************************************************)
354     (* Modifies the list of expressions provided upon creation of a new class *)
355     (* instance to incorporate both default and explicit values for instance  *)
356     (* variables. This approach, namely creating a new env, allows for the    *)
357     (* following in a default constructor, where two instance vars can build  *)
358     (* the value of a third. The main strategy is to build up semantic expr   *)
359     (* for the defaults, then append custom args. This allows the LLVM code   *)
360     (* to always set the defaults, then override with the custom, if necessary*)
361     (*                                                                       *)
362     (* HAI ITZ ME CLASS Example                                              *)
363     (*       ITZ ME NUMBR num1 IZ 2.                                         *)
364     (*        ITZ ME NUMBR num2 IZ 5.                                        *)
365     (*       ITZ ME NUMBR sum IZ SUM OF num1 AN num2.                        *)
366     (*        ...                                                            *)
367     (*    KBYE                                                               *)
368     (*****************************************************************************)
369     let mangled_constructor_args =
370       let construct_env = {
371         (* Create a new environment representing constructor *)
```

```
372         in_class = true;  (* turns on auto ClassAccess for instance vars *)
373         class_name = cname;
374         constructor = true; (* tells us we are working on a constructor scenario *)
375         obj_name = obj_name;
376         function_name = "";
377         returns = Void;
378         symbols = Hashtbl.create 10;
379       } in
380       (* Create a list of assignment expressions for default args *)
381       let default_vars =
382         let set_default_vars acc (typ, id, expr) =
383           let (typ', e') = semant_expr expr construct_env in
384           (match e' with
385               SNoexpr -> acc
386             | _ ->
387               let lhs = (typ, SClassAccess(Obtype(cname), (Obtype(cname), SId(obj_name)), id))
388               and rhs = (typ', e') in
389               Hashtbl.add construct_env.symbols id typ;
390               (typ, SAssign(lhs, rhs)) :: acc)
391         in
392         List.fold_left set_default_vars [] (List.rev cls.cvars) in
393
394       (* This tricky code is meant to allow someone to assign instance vars
395         by name, using assignment-like expressions, to create a new class instance *)
396       let check_constructor_arg acc expr =
397         (* make sure that all variables/types in assignment are part of class *)
398         (match expr with
399           | Assign(Id(id), e) ->
400               let (typ, e') = semant_expr e env in
401               let cvars = List.map (fun (typ, name, _) -> (typ, name)) cls.cvars in
402               if List.mem (typ, id) cvars then
403                 (* Convert the assignment statement into an assignment with class access *)
404                 let lhs = (typ, SClassAccess(Obtype(cname), (Obtype(cname), SId(obj_name)), id))
405                 and rhs = (typ, e') in
406                 (* Append the new expression to the growing list *)
407                 (typ, SAssign(lhs, rhs)) :: acc
408               else
409                 let msg = Printf.sprintf "%s, found: %s in allocation of new %s"
410                           object_constructor_types (string_of_expr expr) cname
411                 in raise (ObjectConstructorInvalid(msg))
412           | _ ->
413             let msg = Printf.sprintf "%s, found: %s in allocation of new %s"
414                     object_constructor_error (string_of_expr expr) cname
415             in raise(ObjectConstructorInvalid(msg)))
416         in
417       List.fold_left check_constructor_arg default_vars expr_list in
418
419     (* Make sure to add the allocated obj to symbol table so it can be referenced *)
```

```
420     Hashtbl.add env.symbols obj_name (Obtype (cname));
421     (* Return SAST for new instance with modified constructor! *)
422     (typ, SNewInstance(obj_name, typ, List.rev mangled_constructor_args))
423
424   | ClassAccess (v, class_var) as ex ->
425     (* 1. Check that the object exists in the symbol table  *)
426     let obj_name =
427       (match v with
428           Id (id) -> id
429         | _ -> raise (InstanceVariableAccessInvalid(class_access_msg)))
430     in
431     let (typ, identifer) = semant_expr v env in
432     (* 2. You can only "access" instance variables of type Obtype *)
433     let cname =
434       (match typ with
435           Obtype o -> o
436         | _ ->
437           let msg = Printf.sprintf "%s, found: %s" invalid_instance_var_access (string_of_expr ex)
438           in raise (InstanceVariableAccessInvalid(msg)))
439     in
440     (* 3. Check that the instance variable exists within the class *)
441     let cls = find_class cname in
442     let cvars = List.map (fun (_, name, _) -> name) cls.cvars in
443     if List.mem class_var cvars then
444       let rec find_typ n vars =
445         match vars with
446           [] -> raise (InternalError("unexpectedly could not determine type of class variable\n"))
447         | (typ, name, _) :: t -> if name = n then typ else find_typ n t
448       in
449       (find_typ class_var cls.cvars, (SClassAccess(Obtype(cname), (typ, identifer), class_var)))
450     else
451       let msg = Printf.sprintf "%s, instance of class %s, has no member %s"
452                 obj_name cname class_var
453       in raise (InstanceVariableNotFound(msg))
454
455   | ArrayAccess (array_id, e) as ex ->
456     (* 1. Check that the array exists in the symbol table  *)
457     let typ = find_type_of_id env.symbols array_id in
458
459     (* 2. You can only have index access to items of type ArrType *)
460     let (_, contents_typ) =
461       match typ with
462       | Arrtype (ILiteralArraySize(sz), arr_typ) ->
463         (match e with
464             (* Test if array access is out of bounds; note this is only possible
465                 to do when the array hasn't been passed as a parameter, in which i < 0 *)
466             ILiteral i ->
467               if ( sz > 0 && i >= sz ) || i < 0 then
```

```
468                 let msg = Printf.sprintf "%s, using index %s in %s, an array of size %s"
469                           array_access_out_of_bounds (string_of_int i) array_id (string_of_int sz)
470                 in raise (InvalidArrayAccess(msg))
471               else (sz, arr_typ)
472           | _ -> (sz, arr_typ))
473       | Arrtype (VariableArraySize(_), arr_typ) -> (0, arr_typ) (* this 0 is to make the compiler happy *)
474       | _ ->
475         let msg = Printf.sprintf "%s; attempting index on '%s' of type %s"
476                   array_access_array_only array_id (string_of_typ typ)
477         in raise (InvalidArrayAccess(msg))
478     in
479     (* 3. Check to make sure that the array is going to be indexed by an integer typ *)
480     let (typ', e') = semant_expr e env in
481     (match typ' with
482       Int -> (contents_typ, SArrayAccess ((array_id), (typ', e')))
483     | _ ->
484       let msg = Printf.sprintf "%s found index expression %s of type %s"
485                 array_access_integer (string_of_expr ex) (string_of_typ typ')
486       in raise (InvalidArrayAccess(msg)))
487
488 (*************************************)
489 (* Main function for checking semantics *)
490 (* of statements                        *)
491 (*************************************)
492 let rec semant_stmt stmt env =
493
494   (* if/else branching: checks that termination expr is a Boolean *)
495   let check_bool_expr e =
496     let (t', e') = semant_expr e env in
497       if t' == Bool then (t', e')
498       else
499         let msg = Printf.sprintf "%s %s %s"
500           op_type_mismatch_boolean (string_of_typ t') (string_of_expr e)
501         in raise (ControlFlowIllegalArgument(msg))
502   and
503
504   (* for looping: checks that op is Increment or Decrement *)
505   check_control_op op =
506     match op with
507       Increment -> op
508     | Decrement -> op
509     | _ -> raise (ControlFlowIllegalArgument(op_type_mismatch_inc_dec ^ string_of_op op))
510   and
511
512   (* for looping: checks that index is an Integer *)
513   check_control_index e =
514     let (t', e') = semant_expr e env in
515       if t' == Int then (t', e')
```

```
516       else
517         let msg = Printf.sprintf "%s %s %s"
518           op_type_mismatch_int (string_of_typ t') (string_of_expr e)
519         in raise (ControlFlowIllegalArgument(msg))
520   and
521
522   (* for looping: second expr is optional or index assignment *)
523   check_index_assignment e =
524     let (t', e') = semant_expr e env in
525     match e' with
526       SNoexpr -> (t', e')
527     | SAssign(_, _) -> (t', e')
528     | _ ->
529       let msg = Printf.sprintf "index assignment expected in expression: %s" (string_of_expr e)
530       in raise (ControlFlowIllegalArgument(msg))
531   and
532
533   (* for looping: checks that loop termination is a binary operation of < or > *)
534   check_loop_termination e =
535     let (t', e') = semant_expr e env in
536       match e' with
537         SBinop(_, op, _) ->
538           (match op with
539             Less | Greater | Equal | Neq -> (t', e')
540           | _ ->
541             let msg = Printf.sprintf "%s %s" op_type_mismatch_loop_term (string_of_expr e)
542             in raise (ControlFlowIllegalArgument(msg)))
543       | _ ->
544         let msg = Printf.sprintf "%s expected binary operation in loop: %s"
545           expr_type_mismatch (string_of_expr e)
546         in raise (ControlFlowIllegalArgument(msg))
547   in
548
549   match stmt with
550     Expr e    -> SExpr(semant_expr e env)
551
552   | Return e  ->
553     if env.returns = Void then
554         let msg = Printf.sprintf "%s; see function %s" return_from_void_func env.function_name
555         in raise (ReturnFromVoidFunction(msg))
556     else
557         let (typ, e') = semant_expr e env in
558         let msg = Printf.sprintf "%s; expected: %s, got: %s in function %s"
559           return_type_invalid (string_of_typ env.returns) (string_of_typ typ) env.function_name
560         in
561         ignore(check_matching_types typ env.returns (ReturnTypeInvalid(msg)));
562         SReturn((typ, e'))
563
```

```
564   | If (e, stmt1, stmt2) ->
565     SIf(check_bool_expr e,
566     semant_stmt stmt1 env,
567     semant_stmt stmt2 env)
568
569   | For (op, e1, e2, e3, stmt) ->
570     SFor(check_control_op op,         (* increment or decrement *)
571         check_control_index e1,       (* index *)
572         check_index_assignment e2,    (* optional index assignment *)
573         check_loop_termination e3,    (* termination condition *)
574         semant_stmt stmt env)         (* loop body *)
575
576   | Block b ->
577     let rec check_stmt_list = function
578         [Return _ as s] -> [semant_stmt s env]
579       | Block sl :: ss  -> check_stmt_list (sl @ ss) (* Flatten blocks *)
580       | s :: ss         ->
581         let fst = semant_stmt s env
582         and lst = check_stmt_list ss in
583         fst :: lst
584       | []              -> []
585     in SBlock(check_stmt_list b)
586
587   | Dealloc (id) as s ->
588     (* Check that the dealloced item is of type ObjType or ArrType *)
589     let id =
590       (match id with
591         Id(v) -> v
592         | _ -> raise (InvalidDealloc(invalid_deallocation_msg ^ string_of_stmt s)))
593     in
594     let typ = find_type_of_id env.symbols id in
595     (match typ with
596       Obtype _ | Arrtype _ | String -> SDealloc(typ, SId(id))
597     | _ ->
598       let msg = Printf.sprintf "%s %s is of typ %s"
599         invalid_deallocation_msg id (string_of_typ typ)
600       in raise (InvalidDealloc(msg)))
601
602   | ClassAssign (id, instance_var, e) ->
603     (* 1. id must correspond to an ObjType *)
604     let (typ, identifier) = semant_expr id env in
605     (match typ with
606       Obtype (cname) ->
607         let cls = find_class cname in
608         let cvars = List.map (fun (typ, name, _) -> (typ, name)) cls.cvars in
609         let (vtype, e') = semant_expr e env in
610
611         (* 2. the instance variable must exist in the class and the
```

```
612         item being assigned must be of the correct type *)
613         if List.mem (vtype, instance_var) cvars then
614           SClassAssign(Obtype(cname), (typ, identifier), instance_var, (vtype, e'))
615         else
616           let msg = Printf.sprintf "%s %s to %s.%s"
617                 invalid_cls_member_assign (string_of_typ typ) cname instance_var
618           in raise (InvalidClassMemberAssignment(msg))
619     | _ ->
620       let msg = Printf.sprintf "%s %s is of type %s"
621           member_assign_cls_only (string_of_expr id) (string_of_typ typ)
622       in raise (InvalidClassMemberAssignment(msg)))
623
624   | ArrayAssign (id, idx_e, e) ->
625     (* 1. make sure that the variable is an array type *)
626     let typ = find_type_of_id env.symbols id in
627     (match typ with
628       Arrtype (_, ty) ->
629         (* 2. make sure that the idx_e expression yields an integer *)
630         let (idx_typ, idx_e') = semant_expr idx_e env in
631         (match idx_typ with
632           Int ->
633             (* 3. make sure that the expression type being assigned matches array content type *)
634             let (exp_typ, e') = semant_expr e env in
635               if exp_typ = ty then
636                 SArrayAssign(id, (idx_typ, idx_e'), (exp_typ, e'))
637               else
638                 let msg = Printf.sprintf "%s, found '%s' of type %s"
639                           invalid_array_item_msg (string_of_expr e) (string_of_typ exp_typ)
640                 in raise (InvalidArrayAssignment(msg))
641           | _ ->
642             let msg = Printf.sprintf "%s found index expression '%s' of type %s"
643                     array_access_integer (string_of_expr idx_e) (string_of_typ idx_typ)
644             in raise (InvalidArrayAssignment(msg)))
645     | _ ->
646       let msg = Printf.sprintf "%s; %s is not an array" array_access_array_only id
647       in raise (InvalidArrayAssignment(msg)))
648
649
650 (*************************************)
651 (* Checks that a function body is       *)
652 (* semantically correct.                *)
653 (*************************************)
654 let check_function_body func env =
655   (*
656     1. Build local symbol table of variables for this scope
657   *)
658   List.iter (fun (typ, name) -> Hashtbl.add env.symbols name typ) func.formals;
659   List.iter (fun (typ, name, _) -> Hashtbl.add env.symbols name typ) func.locals;
```

```
660
661   (*
662     2. Refactoring step: moves assignments in func.locals into function body
663        beginning analogous to make the following change in a c function :
664           int value = 2;   -> int value;
665                               value = 2;
666   *)
667   let create_assignment_stmt build local_var_bind =
668     (match local_var_bind with
669       (_, _, Noexpr)  -> build
670     | (_, name, expr) -> Expr(Assign(Id(name), expr)) :: build) in
671   let new_assignments =  List.fold_left create_assignment_stmt [] func.locals in
672   let adjusted_body = List.rev new_assignments @ func.body in
673
674   (* 3. Build up the SAST Tree for the Function *)
675   semant_stmt (Block adjusted_body) env
676
677 (*************************************************)
678 (* Main function for checking to ensure contents of *)
679 (* function/method is semantically valid.           *)
680 (*************************************************)
681 let check_function_or_method func env =
682   (* 1. Get a list of formal names and local variable names then
683      2. Check for duplicate formal and duplicate local variable names on their own
684      3. Check for duplicates in formals and locals together
685      4. If there are other variables in env (i.e., instance vars) also check those *)
686   let list_formal_names = List.fold_left (fun acc (_, name) -> name :: acc) [] func.formals
687   and list_locals_names = List.fold_left (fun acc (_, name, _) -> name  :: acc) [] func.locals
688   and list_other_names = Hashtbl.fold (fun k _ acc -> k :: acc) env.symbols []
689   in
690   find_duplicate (list_formal_names) dup_formal_msg;
691   find_duplicate (list_locals_names) dup_local_var_msg;
692   find_duplicate (list_formal_names @ list_locals_names) dup_form_local_msg;
693   find_duplicate (list_formal_names @ list_locals_names @ list_other_names) dup_form_instance_msg;
694
695   (* 4. Step to make LLVM code happy; main function must be 'main' not 'Main' *)
696   let adjusted_function_name f = if f.fname = "Main" then "main" else f.fname
697   in env.function_name <- func.fname;
698
699   (* 5. Check contents of function body, producting SAST version *)
700   let checked_func = {
701     styp = func.typ;
702     sfname = adjusted_function_name func;
703     sformals = func.formals;
704     slocals  = func.locals;
705     sbody = match check_function_body func env with
706         SBlock(stmt_list) -> stmt_list
707       | _ -> raise (InternalError ("unexpected_error: no block in function body"))
```

```
  } in
  checked_func

(**************************************)
(* Checks that class definition is    *)
(* semantically correct.              *)
(**************************************)
 let check_class cls =
  (* instance variables defined cannot be duplicated within a single cls *)
  let list_instance_vars = List.fold_left (fun acc (_, name, _) -> name  :: acc) [] cls.cvars
  in
  find_duplicate (list_instance_vars) dup_instance_var_msg;

  (* create a new environment for the class, will contain instance vars *)
  let env = {
    in_class = true;
    class_name = cls.cname;
    constructor = false;
    obj_name = "";
    function_name = "";
    returns = Void;
    symbols = Hashtbl.create 10;
  } in

  (* Make sure that default values for instance variables make sense *)
  let eval_instance_var (typ, name, expr) =
    (match expr with
      (* no default value for instance var provided *)
      Noexpr -> Hashtbl.add env.symbols name typ; (typ, name, (Void, SNoexpr))
    | _ ->
      (* check assignment statement types *)
      let (typ', expr') = semant_expr expr env in
      if typ != typ' then
        let msg = Printf.sprintf "%s %s, expected %s"
          object_constructor_types (string_of_typ typ') (string_of_typ typ)
        in raise (ObjectInstanceVariableInvalid(msg))
      else
        Hashtbl.add env.symbols name typ; (typ, name, (typ', expr')))
  in
  let instance_vars_evaluated = List.map eval_instance_var (List.rev cls.cvars) in

  (* check the methods in each class, passing a symbol tbl
     that includes the instance vars *)
  let checked_cls = {
    scname = cls.cname;
    scvars = instance_vars_evaluated;
    scfuncs = List.map (
      fun f ->
```

```
756         env.returns <- f.typ;
757         check_function_or_method f env
758     ) cls.cfuncs;
759   } in
760   checked_cls
761
762 (*************************************************)
763 (* Wrapper for function that checks of function is  *)
764 (* semantically correct.                         *)
765 (*************************************************)
766 let check_function func =
767
768   (* create a new environment for the function scope *)
769   let env = {
770     in_class = false;
771     class_name = "";
772     constructor = false;
773     obj_name = "";
774     returns = func.typ;
775     function_name = func.fname;
776     symbols = Hashtbl.create 10;
777   }
778   in check_function_or_method func env
779
780 (*************************************************)
781 (* Moves a method within a class into the global      *)
782 (* space by renaming it "<Class_Name>.<MethodName>"   *)
783 (* and providing the Objtype(Class_Name) as the first *)
784 (* argument, an argument named "<Class_Name>*"        *)
785 (*************************************************)
786 let lift_methods_to_global_space cls =
787   let lift_method m =
788     {
789       styp = m.styp;
790       sfname = m_to_f_name cls.scname m.sfname;
791       sformals = (Obtype(cls.scname), mangled_obj_varname cls.scname) :: m.sformals;
792       slocals  = m.slocals;
793       sbody = m.sbody;
794     }
795   in
796   List.map lift_method cls.scfuncs
797
798 (*****************************************************************************)
799 (* Entry point for Semantic Checker, transforming AST to SAST                *)
800 (*****************************************************************************)
801 let check (_, functions, classes) =
802
803   (* 1. add built in functions to list of functions and
```

```
    2. Check for any duplicate function, method and class names *)
  let functions' = add_built_ins functions in
  check_duplicates functions' classes;

  (* 3. Since functions/classes are global, create maps of functions, classes *)
  List.iter (fun func -> Hashtbl.add function_tbl func.fname func) functions';
  List.iter (fun cls-> Hashtbl.add class_tbl cls.cname cls) classes;

  (* 4. Make sure that a "main" function exists, and if so, continue with
       creating a list of checked functions, converted to SAST form *)
  if Hashtbl.mem function_tbl "Main" then
    let semant_classes = List.map check_class classes
    and semant_funcs = List.map check_function functions in

    (* 5. The combined functions represent the "lifted" class methods and usual functions *)
    let combined_functions =
      List.fold_left (
        fun fs cls -> lift_methods_to_global_space cls @ fs
      ) semant_funcs semant_classes
    in
    ([], combined_functions, semant_classes)

  else raise (MissingMainFunction (missing_main_func_msg))
```

Listing 43: src/lib/custom_casting.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>


char *custom_itoa(int val)
{
    int n_chars_in_val = 1; /* need at least one for null terminator */
    char *buffer;
    int stored = val;

    /* find number of characters we need */
    while (val != 0) {
            val /= 10;
            n_chars_in_val++;
        }

    /* actually create the buffer */
    buffer = malloc(sizeof(char) * n_chars_in_val);
    if (!buffer) {
        printf("%s\n", strerror(errno));
```

```
        return NULL;
    }

    /* ... and add the values to it */
    if (sprintf(buffer, "%d", stored) != n_chars_in_val - 1) {
        printf("could not cast int to string\n");
        free(buffer);
        exit(1);
    }

    return buffer;
}

char *custom_ftoa(double val)
{
    char *buffer;
    int sz;

    sz = asprintf(&buffer, "%f", val);

    if (sz < 0) {
        /* memory allocation unsuccessful */
        printf("could not cast float to string\n");
        exit(1);
    }
    return buffer;
}
```

Listing 44: src/lib/custom_scanf.c

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>

int custom_scanf(char **buf_ptr)
{
    int size_of_buffer = 10;
    int size = 0;
    char c;
    int n_characters = 0;
    char *buffer;

    buffer = malloc(sizeof(char) * size_of_buffer);
    if (!buffer) {
        printf("%s\n", strerror(errno));
        return -1;
    }
```

```
19
20      while ((c = getchar()) != EOF && (c != '\n')) {
21          buffer[n_characters] = c;
22          n_characters++;
23
24          /* if buffer is full then realloc */
25          if (n_characters == size_of_buffer) {
26              size_of_buffer *= 2;
27              buffer = realloc(buffer, sizeof(char) * size_of_buffer);
28              if (!buffer) {
29                  printf("%s\n", strerror(errno));
30                  return -1;
31              }
32          }
33      }
34      buffer[n_characters] = '\0';
35      *buf_ptr = buffer;
36      return n_characters;
37  }
```

Listing 45: src/lib/custom_strcat.c

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <errno.h>
4   #include <string.h>
5
6   char *custom_strcat(char *lhs, char *rhs)
7   {
8     /* the size of the concatenated string */
9     int size_of_buffer = strlen(lhs) + strlen(rhs) + 1;
10
11    /* malloc space for the new string */
12      char *buffer;
13      buffer = malloc(sizeof(char) * size_of_buffer);
14      if (!buffer) {
15          printf("%s\n", strerror(errno));
16          return "\0";
17      }
18
19    memcpy(buffer, lhs, strlen(lhs));
20    memcpy(buffer+strlen(lhs), rhs, strlen(rhs));
21    /* null terminate the buffer */
22      buffer[strlen(lhs) + strlen(rhs)]= '\0';
23    return buffer;
24  }
```

Listing 46: src/lib/custom_strcmp.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

/**
 *  Returns 0 if the strings are different, and 1 if they match.
 *  Note that this is the reverse of the ordering strncmp.
 */
int custom_strcmp(char *str1, char *str2)
{
    if (strncmp(str1, str2, strlen(str1)))
        return 0;
    return 1;
}
```

8.2 Test Programs

8.2.1 Test Scripts

This section includes helper scripts used for testing purposes.

Listing 47: test/run_regression_tests.sh

```
#!/bin/bash
# Use this file to perform regression tests on the meowlang compiler.
# This file is designed to be used from the project root directory and run as:
# ./test/run_regression_tests.sh [run_type]

WORKING_DIR="./test/test_programs"
COMPLIER="../../src/meowlang.native"

# helper function to print usage if incorrect args are passed
Usage() {
        echo "Usage ./test/run_regression_tests.sh [-a|-s|-c] [files]"
        echo "Test type must be specified: "
        echo "  -a (AST: Scanner/Parser Only)"
        echo "  -s (Semantic: Run Semantic Checks)"
        echo "  -c (Compile: Compile to LLVM, Execute)"
        echo "[files]: optional; list of .meow files, default uses all files"
        exit 1
}

# checks to make sure that a source file that should pass still passes
# and compares old result with the new result. must call only after
# the test_type_dir and the run_type global variables are set by the script
Check() {
```

```
    test_file=$1      # name of the test file to run
    should_pass=$2    # do we expect the test to pass?

    base_name=$(basename $file .meow)                                    # e.g., "test_conditionals"
    actual_output="$base_name.out"                                       # e.g., "test_conditionals.out"
    expected_output="../test_output/$test_type_dir/$base_name.out"       # e.g., ./test/test_output/ast/
test_conditions.out

    echo -e "\n*** Running $base_name ($test_type_dir) *** " | tee -a $global_log

    # run the relevant test
    if [ $run_type == "-a" ] || [ $run_type == "-s" ]
    then
            # semantic and ast checks are simple to test
            $COMPLIER $run_type -f $test_file &> $actual_output
    else
            cd ../..
            # testing full compilation requires help from test script
            # note that these two tests require input from stdin (they ask questions)
            # here is a work around to provide input so no human intervention is required
            if [[ "$test_file" == *"scan"* ]]; then
                    echo "some value" | ./test/test_single_program.sh "$base_name.meow" &> ./test/test_programs
/$actual_output
            elif [[ "$test_file" == *"_imports.meow" ]]; then
                    echo "tester" | ./test/test_single_program.sh "$base_name.meow" &> ./test/test_programs/
$actual_output
            else
                    ./test/test_single_program.sh "$base_name.meow" &> ./test/test_programs/$actual_output
            fi
            cd $WORKING_DIR
    fi

    # see if the result is what we expected
    if [[ $should_pass && $? -ne 0 ]];
    then
            echo "TEST FAILED: Test unexpectedly failed" | tee -a $global_log
            exit 1
    elif [[ !$should_pass && $? -eq 0 ]];
    then
            echo "TEST FAILED: Test unexpectedly passed" | tee -a $global_log
            exit 1
    fi

    # if we don't have the output file yet, just return
    if [[ ! -f "$expected_output" ]]; then
            echo "$expected_output does not yet exist, so cannot calculate diff."
            rm -f $actual_output
            return 0
```

```
69          fi
70          # run this twice to be sure to easily capture output in global log
71          diff -b $actual_output $expected_output >> $global_log
72          diff -b $actual_output $expected_output
73          if [ $? -ne 0 ];
74          then
75                  echo "TEST FAILED: Result did not match expected output" | tee -a $global_log
76                  exit 1
77          fi
78
79          # report result and remove the old files
80          echo "TEST PASSED" | tee -a $global_log
81          rm -f $actual_output
82  }
83
84  # ------------------- ENTRY POINT TO REGRESSION TESTS ---------------------- #
85
86  n_tests_completed=0
87  global_log="../global_log.out"
88
89  cd $WORKING_DIR
90
91  if [[ $# -le 0 ]]
92  then
93          echo "Test-type must be specified as a command line argument"
94          Usage
95  else
96          run_type=$1
97  fi
98
99  # find the test type (e.g., AST, Semantic Checks, Or Whole Shabang)
100 if [ $run_type == "-a" ]
101 then
102         echo "****************************************"
103         echo " RUNNING CHECKS ON ABSTRACT SYNTAX TREE" | tee -a $global_log
104         echo "****************************************"
105         test_type_dir="ast"
106 elif [ $run_type == '-s' ]
107 then
108         echo "****************************************"
109         echo "     RUNNING CHECKS ON SEMANTICS" | tee -a $global_log
110         echo "****************************************"
111         test_type_dir="semantic"
112 elif [ $run_type == "-c" ]
113 then
114         echo "****************************************"
115         echo "    RUNNING CHECKS ON FULL PIPELINE" | tee -a $global_log
116         echo "****************************************"
```

```
        test_type_dir="full_pipeline"
else
        echo "Command line arg $1 is not yet a supported test type"
        Usage
fi

# (1) Get test files to run regression tests on
# if someone passes a specific file or list of files, just run those

if [ $# -ge 2 ]
then
        files=$@
else
        files=$(find . -type f -name "*.meow")
fi

# remove old global log, if it still exists
rm -f $global_log

# (2) Run each file ("test-*.meow" means test should succeed; each "fail-*.meow should fail")
for file in $files
do
        case $file in
            ./test*)
                    Check $file $true
                        let n_tests_completed++
                        ;;

                ./fail*)
                        Check $file $false
                        let n_tests_completed++
                        ;;
                *)
                        echo ""
                        echo "Skipping accessory file $file..." ;;
        esac
done

echo -e "\n*** $n_tests_completed successful $test_type_dir tests completed! Good to go! ***" 2>&1 | tee -a
    $global_log
```

Listing 48: `test/test_all.sh`

```
#!/bin/bash
# Runs all tests

echo "************************************************************"
echo "                   STARTING ALL CHECKS!                     "
```

```
echo "************************************************************"

./test/run_regression_tests.sh -a
if [ $? -ne 0 ]
then
    echo "AST Checks Failed!"
    exit 1
fi

./test/run_regression_tests.sh -s
if [ $? -ne 0 ]
then
    echo "Semantic Checks Failed!"
    exit 1
fi

./test/run_regression_tests.sh -c
if [ $? -ne 0 ]
then
    echo "Full Pipeline Checks Failed!"
    exit 1
fi

echo ""
echo "************************************************************"
echo "                    ALL CHECKS PASS!                        "
echo "************************************************************"
```

Listing 49: `test/test_single_program.sh`

```

LLC="llc"                           # LLVM compiler
CC="cc"                             # C compiler
REL_DIR="../../src"                 # src directory relative to test dir
MEOWLANG="$REL_DIR/meowlang.native"   # Meowlang compiler
TEST_PROGRAM_PATH="test/test_programs"

# Paths to the relevant .o file to link into compiler
CUSTOM_SCAN="$REL_DIR/custom_scanf.o"
CUSTOM_STRCMP="$REL_DIR/custom_strcmp.o"
CUSTOM_STRCAT="$REL_DIR/custom_strcat.o"
CUSTOM_CASTING="$REL_DIR/custom_casting.o"

if [[ $# -le 0 ]]
then
        echo -e "Test file must be provided as a command line arg \n"
        echo "./test/test_single_program <program_name.meow>"
        echo "Add --keep flag to keep all files after run is complete"
```

```
fi

cd $TEST_PROGRAM_PATH

# build everything
name=$(basename $1 .meow)

$MEOWLANG -f "$1" -c > "$name.ll"

if [[ $? -eq 0 ]];
then
        $LLC -relocation-model=pic "$name.ll" > "$name.s" &&
        $CC -o "$name.exe" "$name.s" $CUSTOM_SCAN $CUSTOM_STRCMP $CUSTOM_STRCAT $CUSTOM_CASTING  &&
        "./$name.exe"
fi

# clean up, unless --keep flag is provided
if [[ $# -lt 2 || $2 != "--keep" ]]
then
        rm -f "$name.ll" "$name.s" "$name.exe"
fi

cd ../.. # go back up to where you where before
```

8.2.2 Test `.meow` Files + Expected Output

This appendix section contains all `.meow` test programs for the `Meowlang` compiler.

Listing 50: `test/test_programs/fail_array_access1.meow`

```
HAI ITZ ME FUNC Main,

   ITZ ME YARN index IZ "index?".
   ITZ ME YARN result.

    PSST You cannot access elements with not an int variable or literal
    MAEK greetings NEW BUCKET OF YARN HOLDS 3,
        WIT "string 1"
        AN "string 2".

   result IZ greetings[index].
   BLEEP greetings.

KBYE
```

Listing 51: `test/test_output/ast/fail_array_access1.out`

```
1
2
3  Main()
4  {
5      char * index = "index?";
6      char * result;
7      char * [3] greetings = [ "string 1", "string 2" ];
8      result = greetings[index];
9      free(greetings);
10 }
```

Listing 52: test/test_output/semantic/fail_array_access1.out

```
1  Fatal error: exception Exceptions.InvalidArrayAccess("arrays can only be indexed with integer types (>= 0)
       : found index expression greetings[index] of type char *")
```

Listing 53: test/test_output/full_pipeline/fail_array_access1.out

```
1  Fatal error: exception Exceptions.InvalidArrayAccess("arrays can only be indexed with integer types (>= 0)
       : found index expression greetings[index] of type char *")
```

Listing 54: test/test_programs/fail_array_access2.meow

```
1
2  HAI ITZ ME FUNC Main,
3
4     ITZ ME NUMBAR index IZ 1.0.
5     ITZ ME YARN result.
6
7      PSST You cannot access elements with not an int variable or literal
8      MAEK greetings NEW BUCKET OF YARN HOLDS 3,
9          WIT "string 1"
10         AN "string 2".
11
12    result IZ greetings[index].
13    BLEEP greetings.
14
15 KBYE
```

Listing 55: test/test_output/ast/fail_array_access2.out

```
1
2
3  Main()
4  {
5      float index = 1.0;
6      char * result;
7      char * [3] greetings = [ "string 1", "string 2" ];
```

```
    result = greetings[index];
    free(greetings);
}
```

Listing 56: test/test_output/semantic/fail_array_access2.out

```
Fatal error: exception Exceptions.InvalidArrayAccess("arrays can only be indexed with integer types (>= 0)
    : found index expression greetings[index] of type float")
```

Listing 57: test/test_output/full_pipeline/fail_array_access2.out

```
Fatal error: exception Exceptions.InvalidArrayAccess("arrays can only be indexed with integer types (>= 0)
    : found index expression greetings[index] of type float")
```

Listing 58: test/test_programs/fail_array_access3.meow

```

HAI ITZ ME FUNC Main,

   ITZ ME YARN result.

    PSST You cannot access elements with not an int variable or literal
    MAEK greetings NEW BUCKET OF YARN HOLDS 3,
        WIT "string 1"
        AN "string 2".

   result IZ greetings["bad index!"].
   BLEEP greetings.

KBYE
```

Listing 59: test/test_output/ast/fail_array_access3.out

```


Main()
{
    char * result;
    char * [3] greetings = [ "string 1", "string 2" ];
    result = greetings["bad index!"];
    free(greetings);
}
```

Listing 60: test/test_output/semantic/fail_array_access3.out

```
Fatal error: exception Exceptions.InvalidArrayAccess("arrays can only be indexed with integer types (>= 0)
    : found index expression greetings["bad index!"] of type char *")
```

Listing 61: test/test_output/full_pipeline/fail_array_access3.out

```
Fatal error: exception Exceptions.InvalidArrayAccess("arrays can only be indexed with integer types (>= 0)
    : found index expression greetings["bad index!"] of type char *")
```

Listing 62: test/test_programs/fail_array_access4.meow

```
HAI ITZ ME FUNC Main,

   ITZ ME YARN result IZ "A string!".

   PSST You cannot index something that is not an array type
   result IZ result[0].

KBYE
```

Listing 63: test/test_output/ast/fail_array_access4.out

```


Main()
{
    char * result = "A string!";
    result = result[0];
}
```

Listing 64: test/test_output/semantic/fail_array_access4.out

```
Fatal error: exception Exceptions.InvalidArrayAccess("array indexing is only available for array types;
    attempting index on 'result' of type char *")
```

Listing 65: test/test_output/full_pipeline/fail_array_access4.out

```
Fatal error: exception Exceptions.InvalidArrayAccess("array indexing is only available for array types;
    attempting index on 'result' of type char *")
```

Listing 66: test/test_programs/fail_array_access5.meow

```
HAI ITZ ME FUNC Main,

   ITZ ME NUMBR result IZ 100.

   PSST You cannot index something that is not an array type
   result IZ result[0].

KBYE
```

Listing 67: test/test_output/ast/fail_array_access5.out

```

Main()
{
    int result = 100;
    result = result[0];
}
```

Listing 68: test/test_output/semantic/fail_array_access5.out

```
Fatal error: exception Exceptions.InvalidArrayAccess("array indexing is only available for array types;
    attempting index on 'result' of type int")
```

Listing 69: test/test_output/full_pipeline/fail_array_access5.out

```
Fatal error: exception Exceptions.InvalidArrayAccess("array indexing is only available for array types;
    attempting index on 'result' of type int")
```

Listing 70: test/test_programs/fail_array_access6.meow

```

HAI ITZ ME FUNC Main,

   ITZ ME YARN result.

    MAEK greetings NEW BUCKET OF YARN HOLDS 3,
        WIT "string 1"
        AN "string 2".

   PSST because array was specified with size as integer literal, it can
   PSST determine that this index is out of bounds.
   result IZ greetings[10].
   BLEEP greetings.

KBYE
```

Listing 71: test/test_output/ast/fail_array_access6.out

```

Main()
{
    char * result;
    char * [3] greetings = [ "string 1", "string 2" ];
    result = greetings[10];
    free(greetings);
}
```

Listing 72: test/test_output/semantic/fail_array_access6.out

```
Fatal error: exception Exceptions.InvalidArrayAccess("index exceeds array size, using index 10 in
    greetings, an array of size 3")
```

Listing 73: test/test_output/full_pipeline/fail_array_access6.out

```
Fatal error: exception Exceptions.InvalidArrayAccess("index exceeds array size, using index 10 in
    greetings, an array of size 3")
```

Listing 74: test/test_programs/fail_array_assignment1.meow

```
1
2  HAI ITZ ME FUNC Main,
3
4     ITZ ME YARN index IZ 0.
5     ITZ ME NUMBR value IZ 1.
6
7      MAEK greetings NEW BUCKET OF YARN HOLDS 3,
8          WIT "string 1"
9          AN "string 2".
10
11    PSST setting the variable at this index of wrong type
12    greetings[index] IZ value.
13    BLEEP greetings.
14 KBYE
```

Listing 75: test/test_output/ast/fail_array_assignment1.out

```
1
2
3  Main()
4  {
5      char * index = 0;
6      int value = 1;
7      char * [3] greetings = [ "string 1", "string 2" ];
8      greetings[index] = value;
9      free(greetings);
10 }
```

Listing 76: test/test_output/semantic/fail_array_assignment1.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
    expected type. expected char *, got int here: index = 0")
```

Listing 77: test/test_output/full_pipeline/fail_array_assignment1.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
    expected type. expected char *, got int here: index = 0")
```

Listing 78: test/test_programs/fail_array_assignment2.meow

```
1
2  HAI ITZ ME FUNC Main,
3
4     ITZ ME YARN value IZ "value".
5     ITZ ME NUMBAR index IZ QUOSHUNT OF 10.0 AN 3.
6
7      MAEK greetings NEW BUCKET OF YARN HOLDS 3,
8          WIT "string 1"
9          AN "string 2".
10
11    PSST setting the variable at this index of wrong type
12    greetings[index] IZ value.
13    BLEEP greetings.
14 KBYE
```

Listing 79: test/test_output/ast/fail_array_assignment2.out

```
1
2
3  Main()
4  {
5      char * value = "value";
6      float index = 10.0 / 3;
7      char * [3] greetings = [ "string 1", "string 2" ];
8      greetings[index] = value;
9      free(greetings);
10 }
```

Listing 80: test/test_output/semantic/fail_array_assignment2.out

```
1  Fatal error: exception Exceptions.InvalidArrayAssignment("arrays can only be indexed with integer types
       (>= 0): found index expression 'index' of type float")
```

Listing 81: test/test_output/full_pipeline/fail_array_assignment2.out

```
1  Fatal error: exception Exceptions.InvalidArrayAssignment("arrays can only be indexed with integer types
       (>= 0): found index expression 'index' of type float")
```

Listing 82: test/test_programs/fail_array_assignment3.meow

```
1
2  HAI ITZ ME FUNC Main,
3
4     ITZ ME YARN value IZ "value".
5     ITZ ME NUMBR index IZ QUOSHUNT OF 10 AN 3.
6
7      MAEK greetings NEW BUCKET OF YARN HOLDS 3,
```

```
        WIT "string 1"
        AN "string 2".

    PSST setting the variable at this index of wrong type
    greetings[0] IZ index.
    BLEEP greetings.
KBYE
```

Listing 83: test/test_output/ast/fail_array_assignment3.out

```


Main()
{
    char * value = "value";
    int index = 10 / 3;
    char * [3] greetings = [ "string 1", "string 2" ];
    greetings[0] = index;
    free(greetings);
}
```

Listing 84: test/test_output/semantic/fail_array_assignment3.out

```
Fatal error: exception Exceptions.InvalidArrayAssignment("array elements must be of specified type for the
     array, found 'index' of type int")
```

Listing 85: test/test_output/full_pipeline/fail_array_assignment3.out

```
Fatal error: exception Exceptions.InvalidArrayAssignment("array elements must be of specified type for the
     array, found 'index' of type int")
```

Listing 86: test/test_programs/fail_array_assignment4.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME YARN value IZ "value".

    PSST you cannot do an index assignment on something other than arrays
    value[0] IZ "new value".

KBYE
```

Listing 87: test/test_output/ast/fail_array_assignment4.out

```


Main()
{
```

```
    char * value = "value";
    value[0] = "new value";
}
```

Listing 88: test/test_output/semantic/fail_array_assignment4.out

```
Fatal error: exception Exceptions.InvalidArrayAssignment("array indexing is only available for array types
    ; value is not an array")
```

Listing 89: test/test_output/full_pipeline/fail_array_assignment4.out

```
Fatal error: exception Exceptions.InvalidArrayAssignment("array indexing is only available for array types
    ; value is not an array")
```

Listing 90: test/test_programs/fail_array_element_not_defined.meow

```

HAI ITZ ME FUNC Main,

    PSST size variable is not yet defined
    MAEK animals NEW BUCKET OF NUMBR HOLDS 10,
        WIT i_am_not_defined
        AN "are"
        AN "strings not ints".

   BLEEP animals.

KBYE
```

Listing 91: test/test_output/ast/fail_array_element_not_defined.out

```


Main()
{
    int [10] animals = [ i_am_not_defined, "are", "strings not ints" ];
    free(animals);
}
```

Listing 92: test/test_output/semantic/fail_array_element_not_defined.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier:  i_am_not_defined")
```

Listing 93: test/test_output/full_pipeline/fail_array_element_not_defined.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier:  i_am_not_defined")
```

Listing 94: test/test_programs/fail_array_holds_none.meow

```
1
2  HAI ITZ ME FUNC Main,
3
4      PSST integer literal values
5      MAEK animals NEW BUCKET OF YARN HOLDS 0.
6
7     BLEEP animals.
8
9  KBYE
```

Listing 95: test/test_output/ast/fail_array_holds_none.out

```
1
2
3  Main()
4  {
5      char * [0] animals = [  ];
6      free(animals);
7  }
```

Listing 96: test/test_output/semantic/fail_array_holds_none.out

```
1  Fatal error: exception Exceptions.InvalidArraySizeSpecified("size of array must be integer > 0")
```

Listing 97: test/test_output/full_pipeline/fail_array_holds_none.out

```
1  Fatal error: exception Exceptions.InvalidArraySizeSpecified("size of array must be integer > 0")
```

Listing 98: test/test_programs/fail_array_mixed_types.meow

```
1
2  HAI ITZ ME FUNC Main,
3
4      PSST You cannot mix types
5      MAEK animals NEW BUCKET OF NUMBR HOLDS 3,
6          WIT 1
7          AN 2
8          AN "this is wrong".
9
10    BLEEP animals.
11
12 KBYE
```

Listing 99: test/test_output/ast/fail_array_mixed_types.out

```
1
2
```

```
Main()
{
    int [3] animals = [ 1, 2, "this is wrong" ];
    free(animals);
}
```

Listing 100: test/test_output/semantic/fail_array_mixed_types.out

```
Fatal error: exception Exceptions.InvalidArrayItem("array elements must be of specified type for the array
    , expected: int, got: char * in int [3] animals = [ 1, 2, "this is wrong" ]")
```

Listing 101: test/test_output/full_pipeline/fail_array_mixed_types.out

```
Fatal error: exception Exceptions.InvalidArrayItem("array elements must be of specified type for the array
    , expected: int, got: char * in int [3] animals = [ 1, 2, "this is wrong" ]")
```

Listing 102: test/test_programs/fail_array_size_not_defined.meow

```

HAI ITZ ME FUNC Main,

    PSST size variable is not yet defined
    MAEK animals NEW BUCKET OF NUMBR HOLDS size,
        WIT "these all"
        AN "are"
        AN "strings not ints".

   BLEEP animals.

KBYE
```

Listing 103: test/test_output/ast/fail_array_size_not_defined.out

```


Main()
{
    int [size] animals = [ "these all", "are", "strings not ints" ];
    free(animals);
}
```

Listing 104: test/test_output/semantic/fail_array_size_not_defined.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier:  size")
```

Listing 105: test/test_output/full_pipeline/fail_array_size_not_defined.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier:  size")
```

Listing 106: test/test_programs/fail_array_size_wrong_type.meow

```
HAI ITZ ME FUNC Main,

        PSST size of array is not an integer
        ITZ ME YARN star IZ "hello".

        MAEK cool_array NEW BUCKET OF NUMBR HOLDS star,
            WIT 1 AN 2 AN 3.
KBYE
```

Listing 107: test/test_output/ast/fail_array_size_wrong_type.out

```


Main()
{
    char * star = "hello";
    int [star] cool_array = [ 1, 2, 3 ];
}
```

Listing 108: test/test_output/semantic/fail_array_size_wrong_type.out

```
Fatal error: exception Exceptions.InvalidArraySizeSpecified("arrays sizes must be integer literals or
    variables only:int [star] cool_array = [ 1, 2, 3 ]")
```

Listing 109: test/test_output/full_pipeline/fail_array_size_wrong_type.out

```
Fatal error: exception Exceptions.InvalidArraySizeSpecified("arrays sizes must be integer literals or
    variables only:int [star] cool_array = [ 1, 2, 3 ]")
```

Listing 110: test/test_programs/fail_array_syntax1.meow

```

HAI ITZ ME FUNC Main,

    PSST creation of array only works with integer or variable specified as size
    PSST cannot accept a complex expression
    MAEK animals NEW BUCKET OF NUMBR HOLDS SUM OF 3 AN 5.

KBYE
```

Listing 111: test/test_output/ast/fail_array_syntax1.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 112: test/test_output/semantic/fail_array_syntax1.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 113: test/test_output/full_pipeline/fail_array_syntax1.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 114: test/test_programs/fail_array_syntax2.meow

```

HAI ITZ ME FUNC Main,

    PSST Forgot array type
    MAEK animals NEW BUCKET HOLDS 3.

KBYE
```

Listing 115: test/test_output/ast/fail_array_syntax2.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 116: test/test_output/semantic/fail_array_syntax2.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 117: test/test_output/full_pipeline/fail_array_syntax2.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 118: test/test_programs/fail_array_syntax3.meow

```

HAI ITZ ME FUNC Main,

    PSST Forgot array name
    MAEK NEW BUCKET OF NUMBR HOLDS 3.

KBYE
```

Listing 119: test/test_output/ast/fail_array_syntax3.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 120: test/test_output/semantic/fail_array_syntax3.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 121: test/test_output/full_pipeline/fail_array_syntax3.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 122: test/test_programs/fail_array_syntax4.meow

```
HAI ITZ ME FUNC Main,

    PSST array sizes are integer values
    MAEK animals NEW BUCKET OF NUMBR HOLDS 3.0,
        WIT "these all"
        AN "are"
        AN "strings not ints".

KBYE
```

Listing 123: test/test_output/ast/fail_array_syntax4.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 124: test/test_output/semantic/fail_array_syntax4.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 125: test/test_output/full_pipeline/fail_array_syntax4.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 126: test/test_programs/fail_array_too_many_elements.meow

```
HAI ITZ ME FUNC Main,

    PSST too many elements added to array with static size
    MAEK animals NEW BUCKET OF YARN HOLDS 2,
        WIT "these all"
        AN "are"
        AN "strings not ints".

   BLEEP animals.

KBYE
```

Listing 127: test/test_output/ast/fail_array_too_many_elements.out

```


Main()
{
    char * [2] animals = [ "these all", "are", "strings not ints" ];
    free(animals);
}
```

Listing 128: test/test_output/semantic/fail_array_too_many_elements.out

```
Fatal error: exception Exceptions.ExcessArrayInput("array contents exceeded specified array size 3")
```

Listing 129: test/test_output/full_pipeline/fail_array_too_many_elements.out

```
Fatal error: exception Exceptions.ExcessArrayInput("array contents exceeded specified array size 3")
```

Listing 130: test/test_programs/fail_array_wrong_types1.meow

```

HAI ITZ ME FUNC Main,

    PSST You cannot mix types
    MAEK animals NEW BUCKET OF NUMBR HOLDS 3,
        WIT "these all"
        AN "are"
        AN "strings not ints".

   BLEEP animals.

KBYE
```

Listing 131: test/test_output/ast/fail_array_wrong_types1.out

```


Main()
{
    int [3] animals = [ "these all", "are", "strings not ints" ];
    free(animals);
}
```

Listing 132: test/test_output/semantic/fail_array_wrong_types1.out

```
Fatal error: exception Exceptions.InvalidArrayItem("array elements must be of specified type for the array
    , expected: int, got: char * in int [3] animals = [ "these all", "are", "strings not ints" ]")
```

Listing 133: test/test_output/full_pipeline/fail_array_wrong_types1.out

```
Fatal error: exception Exceptions.InvalidArrayItem("array elements must be of specified type for the array
    , expected: int, got: char * in int [3] animals = [ "these all", "are", "strings not ints" ]")
```

Listing 134: test/test_programs/fail_array_wrong_types2.meow

```

HAI ITZ ME FUNC Main,

   ITZ ME YARN value1 IZ "hello".
```

```
    ITZ ME YARN value2 IZ "goodbye".

     PSST You cannot mix types
     MAEK greetings NEW BUCKET OF NUMBR HOLDS 3,
         WIT value1
         AN value2.

    BLEEP greetings.

KBYE
```

Listing 135: test/test_output/ast/fail_array_wrong_types2.out

```


Main()
{
    char * value1 = "hello";
    char * value2 = "goodbye";
    int [3] greetings = [ value1, value2 ];
    free(greetings);
}
```

Listing 136: test/test_output/semantic/fail_array_wrong_types2.out

```
Fatal error: exception Exceptions.InvalidArrayItem("array elements must be of specified type for the array
    , expected: int, got: char * in int [3] greetings = [ value1, value2 ]")
```

Listing 137: test/test_output/full_pipeline/fail_array_wrong_types2.out

```
Fatal error: exception Exceptions.InvalidArrayItem("array elements must be of specified type for the array
    , expected: int, got: char * in int [3] greetings = [ value1, value2 ]")
```

Listing 138: test/test_programs/fail_assign.meow

```
HAI ITZ ME FUNC Main,

    PSST expr produces different types than the variable it is assigned to
    ITZ ME NUMBR num IZ 2.
    ITZ ME NUMBR count IZ 3.
    ITZ ME YARN total IZ SUM OF num AN count.

KBYE
```

Listing 139: test/test_output/ast/fail_assign.out

```

```

```
Main()
{
    int num = 2;
    int count = 3;
    char * total = num + count;
}
```

Listing 140: test/test_output/semantic/fail_assign.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
    expected type. expected char *, got int here: total = num + count")
```

Listing 141: test/test_output/full_pipeline/fail_assign.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
    expected type. expected char *, got int here: total = num + count")
```

Listing 142: test/test_programs/fail_bad_import_names.meow

```
GIMME colors?
GIMME shapes?


HAI ITZ ME FUNC Main,
    ITZ ME YARN green.
    ITZ ME YARN red.
KBYE
```

Listing 143: test/test_output/ast/fail_bad_import_names.out

```
Fatal error: exception Exceptions.ImportNotFound("illegal import name colors")
```

Listing 144: test/test_output/semantic/fail_bad_import_names.out

```
Fatal error: exception Exceptions.ImportNotFound("illegal import name colors")
```

Listing 145: test/test_output/full_pipeline/fail_bad_import_names.out

```
Fatal error: exception Exceptions.ImportNotFound("illegal import name colors")
```

Listing 146: test/test_programs/fail_binop1.meow

```
HAI ITZ ME FUNC Main,
    PSST Addition of operands with different types

    ITZ ME YARN string IZ "Value".
    ITZ ME NUMBR value IZ 2.

    ITZ ME NUMBR sum IZ SUM OF string AN value.
KBYE
```

Listing 147: test/test_output/ast/fail_binop1.out

```


Main()
{
    char * string = "Value";
    int value = 2;
    int sum = string + value;
}
```

Listing 148: test/test_output/semantic/fail_binop1.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (char * and int): string
    + value")
```

Listing 149: test/test_output/full_pipeline/fail_binop1.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (char * and int): string
    + value")
```

Listing 150: test/test_programs/fail_binop2.meow

```
HAI ITZ ME FUNC Main,
    PSST Subtraction of operands with different types

    ITZ ME YARN string IZ "Value".
    ITZ ME NUMBAR value IZ 2.0.

    ITZ ME NUMBR diff IZ DIFF OF string AN value.

KBYE
```

Listing 151: test/test_output/ast/fail_binop2.out

```


Main()
{
    char * string = "Value";
    float value = 2.0;
    int diff = string - value;
}
```

Listing 152: test/test_output/semantic/fail_binop2.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (char * and float):
    string - value")
```

Listing 153: test/test_output/full_pipeline/fail_binop2.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (char * and float):
    string - value")
```

Listing 154: test/test_programs/fail_binop3.meow

```
HAI ITZ ME FUNC Main,
    PSST Multiplicaton of operands with different types

    ITZ ME YARN string IZ "Value".
    ITZ ME NUMBR value IZ 2.

    ITZ ME NUMBR product IZ PRODUKT OF string AN value.
KBYE
```

Listing 155: test/test_output/ast/fail_binop3.out

```


Main()
{
    char * string = "Value";
    int value = 2;
    int product = string * value;
}
```

Listing 156: test/test_output/semantic/fail_binop3.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (char * and int): string
    * value")
```

Listing 157: test/test_output/full_pipeline/fail_binop3.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (char * and int): string
    * value")
```

Listing 158: test/test_programs/fail_binop4.meow

```
HAI ITZ ME FUNC Main,
    PSST Division of operands with different types

    ITZ ME YARN string IZ "Value".
    ITZ ME NUMBR integer IZ 2.

    ITZ ME NUMBR quotient IZ QUOSHUNT OF string AN integer.
KBYE
```

Listing 159: test/test_output/ast/fail_binop4.out

```

Main()
{
    char * string = "Value";
    int integer = 2;
    int quotient = string / integer;
}
```

Listing 160: test/test_output/semantic/fail_binop4.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (char * and int): string / integer")
```

Listing 161: test/test_output/full_pipeline/fail_binop4.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (char * and int): string / integer")
```

Listing 162: test/test_programs/fail_binop5.meow

```
HAI ITZ ME FUNC Main,
    PSST Equals

    ITZ ME BOO boolean IZ AYE.
    ITZ ME NUMBR value IZ 2.

    PURR Meow WIT SAEM value AN boolean.

KBYE
```

Listing 163: test/test_output/ast/fail_binop5.out

```

Main()
{
    bool boolean = true;
    int value = 2;
    printf("%X\n", value == boolean);
}
```

Listing 164: test/test_output/semantic/fail_binop5.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (int and bool): value == boolean")
```

Listing 165: test/test_output/full_pipeline/fail_binop5.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (int and bool): value ==
    boolean")
```

Listing 166: test/test_programs/fail_binop6.meow

```
HAI ITZ ME FUNC Main,
    PSST Different cannot occur between different types

    ITZ ME BOO boolean IZ AYE.
    ITZ ME NUMBR value IZ 2.

    DIFFRINT value AN boolean.

KBYE
```

Listing 167: test/test_output/ast/fail_binop6.out

```


Main()
{
    bool boolean = true;
    int value = 2;
    value != boolean;
}
```

Listing 168: test/test_output/semantic/fail_binop6.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (int and bool): value !=
    boolean")
```

Listing 169: test/test_output/full_pipeline/fail_binop6.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (int and bool): value !=
    boolean")
```

Listing 170: test/test_programs/fail_binop7.meow

```
HAI ITZ ME FUNC Main,
    PSST You cannot perform less than operation on two different types

    ITZ ME NUMBR X IZ 2.
    ITZ ME YARN Y IZ "Hello".
    SMALLR X THAN Y.

KBYE
```

Listing 171: test/test_output/ast/fail_binop7.out

```

Main()
{
    int X = 2;
    char * Y = "Hello";
    X < Y;
}
```

Listing 172: test/test_output/semantic/fail_binop7.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (int and char *): X < Y")
```

Listing 173: test/test_output/full_pipeline/fail_binop7.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (int and char *): X < Y")
```

Listing 174: test/test_programs/fail_binop8.meow

```
HAI ITZ ME FUNC Main,
    PSST Greater than can only occur between integer/float types

    ITZ ME NUMBR X IZ 2.
    ITZ ME YARN Y IZ "Hello".
    BIGGR X THAN Y.


KBYE
```

Listing 175: test/test_output/ast/fail_binop8.out

```

Main()
{
    int X = 2;
    char * Y = "Hello";
    X > Y;
}
```

Listing 176: test/test_output/semantic/fail_binop8.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (int and char *): X > Y")
```

Listing 177: test/test_output/full_pipeline/fail_binop8.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (int and char *): X > Y")
```

Listing 178: test/test_programs/fail_binop9.meow

```
HAI ITZ ME FUNC Main,
    PSST And occurs between only two boolean values

    ITZ ME YARN X IZ "Candles".
    ITZ ME YARN Y IZ "Fires".
    BOTH OF X AN Y.

KBYE
```

Listing 179: test/test_output/ast/fail_binop9.out

```


Main()
{
    char * X = "Candles";
    char * Y = "Fires";
    X && Y;
}
```

Listing 180: test/test_output/semantic/fail_binop9.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (char * and char *): X &&
    Y")
```

Listing 181: test/test_output/full_pipeline/fail_binop9.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (char * and char *): X &&
    Y")
```

Listing 182: test/test_programs/fail_binop10.meow

```
HAI ITZ ME FUNC Main,
    PSST Or can only occur between two boolean values

    ITZ ME NUMBR X IZ 2.
    ITZ ME YARN Y IZ "Hello".
    EITHER OF X AN Y.

KBYE
```

Listing 183: test/test_output/ast/fail_binop10.out

```


Main()
{
```

```
    int X = 2;
    char * Y = "Hello";
    X || Y;
}
```

Listing 184: test/test_output/semantic/fail_binop10.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (int and char *): X || Y"
    )
```

Listing 185: test/test_output/full_pipeline/fail_binop10.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (int and char *): X || Y"
    )
```

Listing 186: test/test_programs/fail_binop11.meow

```
HAI ITZ ME FUNC Main,
    PSST Concatenation can only occur between two strings

    ITZ ME BOO X IZ AYE.
    ITZ ME YARN Y IZ "Hello".
    CAT X AN Y.


KBYE
```

Listing 187: test/test_output/ast/fail_binop11.out

```


Main()
{
    bool X = true;
    char * Y = "Hello";
    X + Y;
}
```

Listing 188: test/test_output/semantic/fail_binop11.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (bool and char *): X + Y"
    )
```

Listing 189: test/test_output/full_pipeline/fail_binop11.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (bool and char *): X + Y"
    )
```

Listing 190: test/test_programs/fail_cast_object_to_string.meow

```
1
2  HAI ITZ ME NUMBR FUNC Main,
3
4          PSST Creating some local variables
5          ITZ ME YARN value IZ "happy".
6
7          MAEK Jerry NEW MOUSE.
8
9          PSST Assigning local varibles values
10         Jerry IZ MOUSE value.
11
12         PSST Test printing an integer
13         PURR Meow WIT count.
14
15         GIVE 0.
16 KBYE
17
18 HAI ITZ ME CLASS MOUSE,
19
20     ITZ ME NUMBR cookies IZ 0.
21
22     HAI ITZ ME NUMBR FUNC Count_Cookies,
23         GIVE cookies.
24     KBYE
25
26     HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
27         PSST This uses the SUM prefix operator
28         cookies IZ SUM OF cookies AN count_cookies.
29     KBYE
30
31 KBYE
```

Listing 191: test/test_output/ast/fail_cast_object_to_string.out

```
1
2
3  int Main()
4  {
5      char * value = "happy";
6      class MOUSE Jerry;
7      Jerry = (class MOUSE) value;
8      printf("%X\n", count);
9      return 0;
10 }
11
12 Class MOUSE {
13
```

```
    int cookies = 0;

    Give_Cookie(int count_cookies)
    {
    cookies = cookies + count_cookies;
    }
    int Count_Cookies()
    {
    return cookies;
    }

}
```

Listing 192: test/test_output/semantic/fail_cast_object_to_string.out

```
Fatal error: exception Exceptions.NotYetSupported("cast not currently supported from char * to class MOUSE
    . See: (class MOUSE) value")
```

Listing 193: test/test_output/full_pipeline/fail_cast_object_to_string.out

```
Fatal error: exception Exceptions.NotYetSupported("cast not currently supported from char * to class MOUSE
    . See: (class MOUSE) value")
```

Listing 194: test/test_programs/fail_cast_redundant.meow

```

HAI ITZ ME NUMBR FUNC Main,

        PSST Creating some local variables
        ITZ ME NUMBAR value IZ 2.0.
        ITZ ME NUMBAR float_val.

        PSST Assigning local varibles values
        float_val IZ NUMBAR value.

        PSST Test printing an integer
        PURR Meow WIT float_val.

        GIVE 0.
KBYE
```

Listing 195: test/test_output/ast/fail_cast_redundant.out

```


int Main()
{
    float value = 2.0;
```

```
    float float_val;
    float_val = (float) value;
    printf("%X\n", float_val);
    return 0;
}
```

Listing 196: test/test_output/semantic/fail_cast_redundant.out

```
Fatal error: exception Exceptions.CastUnnecessary("cast is redundant here: (float) value")
```

Listing 197: test/test_output/full_pipeline/fail_cast_redundant.out

```
Fatal error: exception Exceptions.CastUnnecessary("cast is redundant here: (float) value")
```

Listing 198: test/test_programs/fail_cast_string_to_object.meow

```

HAI ITZ ME NUMBR FUNC Main,

        PSST Creating some local variables
        ITZ ME YARN value IZ "happy".

        MAEK Jerry NEW MOUSE.

        PSST Assigning local varibles values
        value IZ YARN Jerry.

        PSST Test printing an integer
        PURR Meow WIT count.

        GIVE 0.
KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 0.

    HAI ITZ ME NUMBR FUNC Count_Cookies,
        GIVE cookies.
    KBYE

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This uses the SUM prefix operator
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE
```

Listing 199: test/test_output/ast/fail_cast_string_to_object.out

```
1
2
3  int Main()
4  {
5      char * value = "happy";
6      class MOUSE Jerry;
7      value = (char *) Jerry;
8      printf("%X\n", count);
9      return 0;
10 }
11
12 Class MOUSE {
13
14     int cookies = 0;
15
16     Give_Cookie(int count_cookies)
17     {
18     cookies = cookies + count_cookies;
19     }
20     int Count_Cookies()
21     {
22     return cookies;
23     }
24
25 }
```

Listing 200: test/test_output/semantic/fail_cast_string_to_object.out

```
1  Fatal error: exception Exceptions.NotYetSupported("cast not currently supported from class MOUSE to char
       *. See: (char *) Jerry")
```

Listing 201: test/test_output/full_pipeline/fail_cast_string_to_object.out

```
1  Fatal error: exception Exceptions.NotYetSupported("cast not currently supported from class MOUSE to char
       *. See: (char *) Jerry")
```

Listing 202: test/test_programs/fail_class1.meow

```
1  HAI ITZ ME FUNC Main,
2
3      MAEK Jerry NEW MOUSE.
4      BLEEP Jerry.
5
6  KBYE
7
8  PSST A class definition cannot have multiple of the same instance variables
9  HAI ITZ ME CLASS MOUSE,
```

```
10
11      ITZ ME NUMBR cookies IZ 0.
12      ITZ ME YARN cookies IZ "duplicate".
13
14      HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
15          PSST This uses the SUM prefix operator
16          cookies IZ SUM OF cookies AN count_cookies.
17      KBYE
18
19  KBYE
```

Listing 203: test/test_output/ast/fail_class1.out

```
1
2
3   Main()
4   {
5       class MOUSE Jerry;
6       free(Jerry);
7   }
8
9   Class MOUSE {
10
11      char * cookies = "duplicate";
12      int cookies = 0;
13
14      Give_Cookie(int count_cookies)
15      {
16      cookies = cookies + count_cookies;
17      }
18
19  }
```

Listing 204: test/test_output/semantic/fail_class1.out

```
1   Fatal error: exception Exceptions.DuplicateIdentifier("duplicate instance variable name identified in
        class declaration: cookies")
```

Listing 205: test/test_output/full_pipeline/fail_class1.out

```
1   Fatal error: exception Exceptions.DuplicateIdentifier("duplicate instance variable name identified in
        class declaration: cookies")
```

Listing 206: test/test_programs/fail_class2.meow

```
1   HAI ITZ ME FUNC Main,
2
3       MAEK Jerry NEW MOUSE.
```

```
    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,

    PSST Cannot assign a default value for var that doesn't exist
    ITZ ME NUMBR cookies IZ this_var_doesnt_exist.

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This uses the SUM prefix operator
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE
```

Listing 207: test/test_output/ast/fail_class2.out

```


Main()
{
    class MOUSE Jerry;
    free(Jerry);
}

Class MOUSE {

    int cookies = this_var_doesnt_exist;

    Give_Cookie(int count_cookies)
    {
    cookies = cookies + count_cookies;
    }

}
```

Listing 208: test/test_output/semantic/fail_class2.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier: this_var_doesnt_exist")
```

Listing 209: test/test_output/full_pipeline/fail_class2.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier: this_var_doesnt_exist")
```

Listing 210: test/test_programs/fail_class3.meow

```
HAI ITZ ME FUNC Main,
```

```
2
3       MAEK Jerry NEW MOUSE.
4       BLEEP Jerry.
5
6   KBYE
7
8   HAI ITZ ME CLASS MOUSE,
9
10      PSST Cannot assign a default value for var that doesn't exist
11      ITZ ME NUMBR cookies IZ "this is the wrong type".
12
13      HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
14          PSST This uses the SUM prefix operator
15          cookies IZ SUM OF cookies AN count_cookies.
16      KBYE
17
18  KBYE
```

Listing 211: test/test_output/ast/fail_class3.out

```
1
2
3   Main()
4   {
5       class MOUSE Jerry;
6       free(Jerry);
7   }
8
9   Class MOUSE {
10
11      int cookies = "this is the wrong type";
12
13      Give_Cookie(int count_cookies)
14      {
15      cookies = cookies + count_cookies;
16      }
17
18  }
```

Listing 212: test/test_output/semantic/fail_class3.out

```
1   Fatal error: exception Exceptions.ObjectInstanceVariableInvalid("you may only assign instance variables
        that are defined within the class and that are of the correct type char *, expected int")
```

Listing 213: test/test_output/full_pipeline/fail_class3.out

```
1   Fatal error: exception Exceptions.ObjectInstanceVariableInvalid("you may only assign instance variables
        that are defined within the class and that are of the correct type char *, expected int")
```

Listing 214: test/test_programs/fail_class4.meow

```
HAI ITZ ME FUNC Main,

    MAEK Jerry NEW MOUSE.
    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 2.

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This variable 'apples' does not exist
        cookies IZ SUM OF apples AN count_cookies.
    KBYE

KBYE
```

Listing 215: test/test_output/ast/fail_class4.out

```


Main()
{
    class MOUSE Jerry;
    free(Jerry);
}

Class MOUSE {

    int cookies = 2;

    Give_Cookie(int count_cookies)
    {
    cookies = apples + count_cookies;
    }

}
```

Listing 216: test/test_output/semantic/fail_class4.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier: apples")
```

Listing 217: test/test_output/full_pipeline/fail_class4.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier: apples")
```

Listing 218: test/test_programs/fail_class5.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR jerrys_cookies.
    MAEK Jerry NEW MOUSE.

    PSST Cannot use keyword HERE outside of a class
    jerrys_cookies IZ PURR Count_Cookies IN HERE.

    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 2.

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE
```

Listing 219: test/test_output/ast/fail_class5.out

```


Main()
{
    int jerrys_cookies;
    class MOUSE Jerry;
    jerrys_cookies = this.Count_Cookies();
    free(Jerry);
}

Class MOUSE {

    int cookies = 2;

    Give_Cookie(int count_cookies)
    {
    cookies = cookies + count_cookies;
    }

}
```

Listing 220: test/test_output/semantic/fail_class5.out

```
Fatal error: exception Exceptions.InvalidMethodCall("use of HERE ('this') keyword can only be used inside
    of a class to refer to its own methods/variables, but found in func Main")
```

Listing 221: test/test_output/full_pipeline/fail_class5.out

```
Fatal error: exception Exceptions.InvalidMethodCall("use of HERE ('this') keyword can only be used inside
    of a class to refer to its own methods/variables, but found in func Main")
```

Listing 222: test/test_programs/fail_class6.meow

```
PSST cannot have duplicates in class names

HAI ITZ ME CLASS FELINES,
    PSST This is one class!
KBYE

HAI ITZ ME CLASS FELINES,
    PSST This is another class!
KBYE

HAI ITZ ME FUNC Main,
    PSST This won't do much
KBYE
```

Listing 223: test/test_output/ast/fail_class6.out

```


Main()
{
}

Class FELINES {




}

Class FELINES {



}
```

Listing 224: test/test_output/semantic/fail_class6.out

```
Fatal error: exception Exceptions.DuplicateIdentifier("duplicate class name: FELINES")
```

Listing 225: test/test_output/full_pipeline/fail_class6.out

```
Fatal error: exception Exceptions.DuplicateIdentifier("duplicate class name: FELINES")
```

Listing 226: test/test_programs/fail_class7.meow

```
HAI ITZ ME CLASS NATURE,
    PSST duplicates in methods within a class
    HAI ITZ ME FUNC Wilderness,
    KBYE

    HAI ITZ ME FUNC Wilderness,
    KBYE

KBYE
```

Listing 227: test/test_output/ast/fail_class7.out

```


Class NATURE {


    Wilderness()
    {
    }
    Wilderness()
    {
    }

}
```

Listing 228: test/test_output/semantic/fail_class7.out

```
Fatal error: exception Exceptions.DuplicateIdentifier("duplicate class method name: Wilderness")
```

Listing 229: test/test_output/full_pipeline/fail_class7.out

```
Fatal error: exception Exceptions.DuplicateIdentifier("duplicate class method name: Wilderness")
```

Listing 230: test/test_programs/fail_class_access1.meow

```
HAI ITZ ME FUNC Main,
    PSST Instance variable doesn't exists within the class
    MAEK Jerry NEW MOUSE.
    bagels IN Jerry.

KBYE
```

```

HAI ITZ ME CLASS MOUSE,
    ITZ ME NUMBR cookies IZ 0.
KBYE
```

Listing 231: test/test_output/ast/fail_class_access1.out

```


Main()
{
    class MOUSE Jerry;
    Jerry.bagels;
}

Class MOUSE {

    int cookies = 0;


}
```

Listing 232: test/test_output/semantic/fail_class_access1.out

```
Fatal error: exception Exceptions.InstanceVariableNotFound("Jerry, instance of class MOUSE, has no member
    bagels")
```

Listing 233: test/test_output/full_pipeline/fail_class_access1.out

```
Fatal error: exception Exceptions.InstanceVariableNotFound("Jerry, instance of class MOUSE, has no member
    bagels")
```

Listing 234: test/test_programs/fail_class_access2.meow

```
HAI ITZ ME FUNC Main,
    PSST accessing instance variables in items that are not type Obtype
    ITZ ME NUMBR Jerry IZ 1.
    cookies IN Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,
    ITZ ME NUMBR cookies IZ 0.
KBYE
```

Listing 235: test/test_output/ast/fail_class_access2.out

```

```

```

Main()
{
    int Jerry = 1;
    Jerry.cookies;
}

Class MOUSE {

    int cookies = 0;


}
```

Listing 236: test/test_output/semantic/fail_class_access2.out

```
Fatal error: exception Exceptions.InstanceVariableAccessInvalid("instance variables only exist in classes, found: Jerry.cookies")
```

Listing 237: test/test_output/full_pipeline/fail_class_access2.out

```
Fatal error: exception Exceptions.InstanceVariableAccessInvalid("instance variables only exist in classes, found: Jerry.cookies")
```

Listing 238: test/test_programs/fail_class_default_vars.meow

```
HAI ITZ ME FUNC Main,

    MAEK Jerry NEW MOUSE.
    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 2.
    ITZ ME NUMBR candies IZ 5.

    PSST defaults build from literals or other class variables
    ITZ ME NUMBR treats IZ SUM OF idontexist AN candies.

KBYE
```

Listing 239: test/test_output/ast/fail_class_default_vars.out

```


Main()
```

```
{
    class MOUSE Jerry;
    free(Jerry);
}

Class MOUSE {

    int treats = idontexist + candies;
    int candies = 5;
    int cookies = 2;


}
```

Listing 240: test/test_output/semantic/fail_class_default_vars.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier: idontexist")
```

Listing 241: test/test_output/full_pipeline/fail_class_default_vars.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier: idontexist")
```

Listing 242: test/test_programs/fail_concat_bool.meow

```
HAI ITZ ME FUNC Main,

        ITZ ME YARN string IZ "string.".
        ITZ ME BOO true IZ AYE.

        PSST concat with boolean not supported
        string IZ CAT string AN true.

        BLEEP string.
KBYE
```

Listing 243: test/test_output/ast/fail_concat_bool.out

```


Main()
{
    char * string = "string.";
    bool true = true;
    string = string + true;
    free(string);
}
```

Listing 244: test/test_output/semantic/fail_concat_bool.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (char * and bool): string
     + true")
```

Listing 245: test/test_output/full_pipeline/fail_concat_bool.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (char * and bool): string
     + true")
```

Listing 246: test/test_programs/fail_concat_floats.meow

```
HAI ITZ ME FUNC Main,

        ITZ ME NUMBAR one IZ 1.0.
        ITZ ME NUMBER two IZ 2.0.
        ITZ ME YARN string.

        PSST concat with two floats is not supported
        string IZ CAT one AN two.

        BLEEP string.
KBYE
```

Listing 247: test/test_output/ast/fail_concat_floats.out

```


Main()
{
    float one = 1.0;
    class NUMBER two = 2.0;
    char * string;
    string = one + two;
    free(string);
}
```

Listing 248: test/test_output/semantic/fail_concat_floats.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
     expected type. expected class NUMBER, got float here: two = 2.0")
```

Listing 249: test/test_output/full_pipeline/fail_concat_floats.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
     expected type. expected class NUMBER, got float here: two = 2.0")
```

Listing 250: test/test_programs/fail_concat_int_float.meow

```
HAI ITZ ME FUNC Main,

        ITZ ME NUMBR one IZ 1.
        ITZ ME NUMBER two IZ 2.0.
        ITZ ME YARN string.

        PSST concat with two floats is not supported
        string IZ CAT one AN two.

        BLEEP string.
KBYE
```

Listing 251: test/test_output/ast/fail_concat_int_float.out

```


Main()
{
    int one = 1;
    class NUMBER two = 2.0;
    char * string;
    string = one + two;
    free(string);
}
```

Listing 252: test/test_output/semantic/fail_concat_int_float.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
    expected type. expected class NUMBER, got float here: two = 2.0")
```

Listing 253: test/test_output/full_pipeline/fail_concat_int_float.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
    expected type. expected class NUMBER, got float here: two = 2.0")
```

Listing 254: test/test_programs/fail_concat_ints.meow

```
HAI ITZ ME FUNC Main,

        ITZ ME NUMBR one IZ 1.
        ITZ ME NUMBR two IZ 2.
        ITZ ME YARN string.

        PSST concat with two ints is not supported
        string IZ CAT one AN two.

        BLEEP string.
KBYE
```

Listing 255: test/test_output/ast/fail_concat_ints.out

```
Main()
{
    int one = 1;
    int two = 2;
    char * string;
    string = one + two;
    free(string);
}
```

Listing 256: test/test_output/semantic/fail_concat_ints.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (int and int): one + two"
    )
```

Listing 257: test/test_output/full_pipeline/fail_concat_ints.out

```
Fatal error: exception Exceptions.IllegalBinaryOp("unexpected types in binary op (int and int): one + two"
    )
```

Listing 258: test/test_programs/fail_create_instance1.meow

```
HAI ITZ ME FUNC Main,
    PSST You can only create an "instance" of something that is type Objtype
    MAEK count NEW NUMBR.
KBYE
```

Listing 259: test/test_output/ast/fail_create_instance1.out

```
Main()
{
    int count;
}
```

Listing 260: test/test_output/semantic/fail_create_instance1.out

```
Fatal error: exception Exceptions.ObjectCreationInvalid("you can only create objects from classes, found:
    int count")
```

Listing 261: test/test_output/full_pipeline/fail_create_instance1.out

```
Fatal error: exception Exceptions.ObjectCreationInvalid("you can only create objects from classes, found:
    int count")
```

Listing 262: test/test_programs/fail_create_instance2.meow

```
HAI ITZ ME FUNC Main,

    PSST the class of new instance does not exist
    MAEK cookies NEW MOUSE.

KBYE
```

Listing 263: test/test_output/ast/fail_create_instance2.out

```


Main()
{
    class MOUSE cookies;
}
```

Listing 264: test/test_output/semantic/fail_create_instance2.out

```
Fatal error: exception Exceptions.ClassNotFound("undeclared identifier: MOUSE")
```

Listing 265: test/test_output/full_pipeline/fail_create_instance2.out

```
Fatal error: exception Exceptions.ClassNotFound("undeclared identifier: MOUSE")
```

Listing 266: test/test_programs/fail_create_instance3.meow

```
HAI ITZ ME FUNC Main,
    PSST If assigning instance variables they must be assignments
    MAEK jerry NEW MOUSE,
        WIT SUM OF 2 AN 1.
    BLEEP jerry.
KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 0.

    HAI ITZ ME NUMBR FUNC Count_Cookies,
        GIVE cookies.
    KBYE

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This uses the SUM prefix operator
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE
```

Listing 267: test/test_output/ast/fail_create_instance3.out

```

Main()
{
    class MOUSE jerry(2 + 1, );
    free(jerry);
}

Class MOUSE {

    int cookies = 0;

    Give_Cookie(int count_cookies)
    {
    cookies = cookies + count_cookies;
    }
    int Count_Cookies()
    {
    return cookies;
    }

}
```

Listing 268: test/test_output/semantic/fail_create_instance3.out

```
Fatal error: exception Exceptions.ObjectConstructorInvalid("to assign instance variables on object
    creation, you must use assignment expressions, found: 2 + 1 in allocation of new MOUSE")
```

Listing 269: test/test_output/full_pipeline/fail_create_instance3.out

```
Fatal error: exception Exceptions.ObjectConstructorInvalid("to assign instance variables on object
    creation, you must use assignment expressions, found: 2 + 1 in allocation of new MOUSE")
```

Listing 270: test/test_programs/fail_create_instance4.meow

```

HAI ITZ ME FUNC Main,
    PSST If assigning instance variables they must be of the expected type
    MAEK jerry NEW MOUSE,
        WIT cookies IZ "this is a string".
    BLEEP jerry.
KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 0.

```

```
    HAI ITZ ME NUMBR FUNC Count_Cookies,
        GIVE cookies.
    KBYE

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This uses the SUM prefix operator
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE
```

Listing 271: test/test_output/ast/fail_create_instance4.out

```


Main()
{
    class MOUSE jerry(cookies = "this is a string", );
    free(jerry);
}

Class MOUSE {

    int cookies = 0;

    Give_Cookie(int count_cookies)
    {
    cookies = cookies + count_cookies;
    }
    int Count_Cookies()
    {
    return cookies;
    }

}
```

Listing 272: test/test_output/semantic/fail_create_instance4.out

```
Fatal error: exception Exceptions.ObjectConstructorInvalid("you may only assign instance variables that
    are defined within the class and that are of the correct type, found: cookies = "this is a string" in
    allocation of new MOUSE")
```

Listing 273: test/test_output/full_pipeline/fail_create_instance4.out

```
Fatal error: exception Exceptions.ObjectConstructorInvalid("you may only assign instance variables that
    are defined within the class and that are of the correct type, found: cookies = "this is a string" in
    allocation of new MOUSE")
```

Listing 274: test/test_programs/fail_create_instance5.meow

```
HAI ITZ ME FUNC Main,
    PSST If assigning instance variables, they must exist in class definition
    MAEK jerry NEW MOUSE,
        WIT test IZ "this is not a valid inst var".
    BLEEP jerry.
KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 0.

KBYE
```

Listing 275: test/test_output/ast/fail_create_instance5.out

```


Main()
{
    class MOUSE jerry(test = "this is not a valid inst var", );
    free(jerry);
}

Class MOUSE {

    int cookies = 0;


}
```

Listing 276: test/test_output/semantic/fail_create_instance5.out

```
Fatal error: exception Exceptions.ObjectConstructorInvalid("you may only assign instance variables that
    are defined within the class and that are of the correct type, found: test = "this is not a valid inst
     var" in allocation of new MOUSE")
```

Listing 277: test/test_output/full_pipeline/fail_create_instance5.out

```
Fatal error: exception Exceptions.ObjectConstructorInvalid("you may only assign instance variables that
    are defined within the class and that are of the correct type, found: test = "this is not a valid inst
     var" in allocation of new MOUSE")
```

Listing 278: test/test_programs/fail_duplicate_import.meow

```
GIMME IMPORT_COLORS?
GIMME IMPORT_COLORS?

```

```
HAI ITZ ME FUNC Main,
    ITZ ME YARN green.
    ITZ ME YARN red.
KBYE
```

Listing 279: test/test_output/ast/fail_duplicate_import.out

```
Fatal error: exception Exceptions.DuplicateImport("Duplicate import in .meow file")
```

Listing 280: test/test_output/semantic/fail_duplicate_import.out

```
Fatal error: exception Exceptions.DuplicateImport("Duplicate import in .meow file")
```

Listing 281: test/test_output/full_pipeline/fail_duplicate_import.out

```
Fatal error: exception Exceptions.DuplicateImport("Duplicate import in .meow file")
```

Listing 282: test/test_programs/fail_for1.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR index IZ 10.

    PSST undeclared identifier count
    IM IN YR LOOP count NERFIN AN BIGGR count THAN 10 HAI
      index IZ PRODUKT OF 5 AN 1.
    KBYE

KBYE
```

Listing 283: test/test_output/ast/fail_for1.out

```


Main()
{
    int index = 10;
    for ( count-- count > 10) {
            {
    index = 5 * 1;
    }
    }
}
```

Listing 284: test/test_output/semantic/fail_for1.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier:  count")
```

Listing 285: test/test_output/full_pipeline/fail_for1.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier:  count")
```

Listing 286: test/test_programs/fail_for2.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR index IZ 10.

    PSST undeclared identifier count
    IM IN YR LOOP index NERFIN index IZ SUM OF 10 AN count AN BIGGR index THAN 10 HAI
      index IZ PRODUKT OF 5 AN 1.
    KBYE

KBYE
```

Listing 287: test/test_output/ast/fail_for2.out

```


Main()
{
    int index = 10;
    for (index = 10 + count index-- index > 10) {
            {
    index = 5 * 1;
    }
    }
}
```

Listing 288: test/test_output/semantic/fail_for2.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier:  count")
```

Listing 289: test/test_output/full_pipeline/fail_for2.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier:  count")
```

Listing 290: test/test_programs/fail_for3.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR index IZ 10.
    ITZ ME NUMBR count IZ 20.
    ITZ ME YARN condition.

    PSST Infinite loop OK
    IM IN YR LOOP index NERFIN count IZ 10 AN BIGGR index THAN 10 HAI
```

```
      index IZ PRODUKT OF 10 AN 1.
    KBYE

    PSST expected integer but got char * condition
    IM IN YR LOOP condition UPPIN AN SMALLR index THAN 10 HAI
      index IZ PRODUKT OF 10 AN 1.
    KBYE

KBYE
```

Listing 291: test/test_output/ast/fail_for3.out

```


Main()
{
    int index = 10;
    int count = 20;
    char * condition;
    for (count = 10 index-- index > 10) {
            {
    index = 10 * 1;
    }
    }
    for ( condition++ index < 10) {
            {
    index = 10 * 1;
    }
    }
}
```

Listing 292: test/test_output/semantic/fail_for3.out

```
Fatal error: exception Exceptions.ControlFlowIllegalArgument("operation type mismatch: expected Integer
    but got type char * condition")
```

Listing 293: test/test_output/full_pipeline/fail_for3.out

```
Fatal error: exception Exceptions.ControlFlowIllegalArgument("operation type mismatch: expected Integer
    but got type char * condition")
```

Listing 294: test/test_programs/fail_for4.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR index IZ 15.
    ITZ ME NUMBR count IZ 0.

```

```
    PSST loop termination is not a binary operation of <, <=, =>, or >
    IM IN YR LOOP index UPPIN AN PRODUKT OF index AN 2 HAI
      count IZ SUM OF 10 AN 1.
    KBYE

KBYE
```

Listing 295: test/test_output/ast/fail_for4.out

```


Main()
{
    int index = 15;
    int count = 0;
    for ( index++ index * 2) {
            {
    count = 10 + 1;
    }
    }
}
```

Listing 296: test/test_output/semantic/fail_for4.out

```
Fatal error: exception Exceptions.ControlFlowIllegalArgument("operation type mismatch: expected <, >, =,
    != as loop termination condition: index * 2")
```

Listing 297: test/test_output/full_pipeline/fail_for4.out

```
Fatal error: exception Exceptions.ControlFlowIllegalArgument("operation type mismatch: expected <, >, =,
    != as loop termination condition: index * 2")
```

Listing 298: test/test_programs/fail_for5.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR index IZ 15.
    ITZ ME NUMBR count IZ 0.

    PSST loop termination is not a binary operation of <, <=, =>, or >
    IM IN YR LOOP index UPPIN AN DIFF OF index AN 2 HAI
      count IZ SUM OF 10 AN 1.
    KBYE

KBYE
```

Listing 299: test/test_output/ast/fail_for5.out

```
1
2
3  Main()
4  {
5      int index = 15;
6      int count = 0;
7      for ( index++ index - 2) {
8              {
9      count = 10 + 1;
10     }
11     }
12 }
```

Listing 300: test/test_output/semantic/fail_for5.out

```
1  Fatal error: exception Exceptions.ControlFlowIllegalArgument("operation type mismatch: expected <, >, =,
       != as loop termination condition: index - 2")
```

Listing 301: test/test_output/full_pipeline/fail_for5.out

```
1  Fatal error: exception Exceptions.ControlFlowIllegalArgument("operation type mismatch: expected <, >, =,
       != as loop termination condition: index - 2")
```

Listing 302: test/test_programs/fail_func1.meow

```
1  HAI ITZ ME FUNC Main,
2      PSST Cannot have duplicate function names
3  KBYE
4
5  HAI ITZ ME FUNC Main,
6      PURR Main.
7  KBYE
```

Listing 303: test/test_output/ast/fail_func1.out

```
1
2
3  Main()
4  {
5      Main();
6  }
7
8  Main()
9  {
10 }
```

Listing 304: test/test_output/semantic/fail_func1.out

```
Fatal error: exception Exceptions.DuplicateIdentifier("duplicate function name, or conflict with built-in: Main")
```

Listing 305: test/test_output/full_pipeline/fail_func1.out

```
Fatal error: exception Exceptions.DuplicateIdentifier("duplicate function name, or conflict with built-in: Main")
```

Listing 306: test/test_programs/fail_func2.meow

```

PSST Void function attempts return
HAI ITZ ME FUNC Main,
    GIVE 0.
KBYE
```

Listing 307: test/test_output/ast/fail_func2.out

```


Main()
{
    return 0;
}
```

Listing 308: test/test_output/semantic/fail_func2.out

```
Fatal error: exception Exceptions.ReturnFromVoidFunction("attempting to return from a void function; see function Main")
```

Listing 309: test/test_output/full_pipeline/fail_func2.out

```
Fatal error: exception Exceptions.ReturnFromVoidFunction("attempting to return from a void function; see function Main")
```

Listing 310: test/test_programs/fail_func3.meow

```

PSST function attempts to return wrong type
HAI ITZ ME YARN FUNC Main,
    GIVE SUM OF 3 AN 4.
KBYE
```

Listing 311: test/test_output/ast/fail_func3.out

```


char * Main()
```

```
{
    return 3 + 4;
}
```

Listing 312: test/test_output/semantic/fail_func3.out

```
Fatal error: exception Exceptions.ReturnTypeInvalid("attempting to return value that doesn't match
    function return type; expected: char *, got: int in function Main")
```

Listing 313: test/test_output/full_pipeline/fail_func3.out

```
Fatal error: exception Exceptions.ReturnTypeInvalid("attempting to return value that doesn't match
    function return type; expected: char *, got: int in function Main")
```

Listing 314: test/test_programs/fail_func4_syntax.meow

```
GIMME hello?

PSST This function is missing comma after declaration
HAI ITZ ME YARN FUNC create_message WIT YARN name

        ITZ ME YARN msg.

        msg IZ "hello, ".
        msg IZ CAT CAT "*" AN msg AN name.

        GIVE msg.
KBYE
```

Listing 315: test/test_output/ast/fail_func4_syntax.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 316: test/test_output/semantic/fail_func4_syntax.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 317: test/test_output/full_pipeline/fail_func4_syntax.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 318: test/test_programs/fail_func5_syntax.meow

```
GIMME hello?

PSST This function is missing HAI KBYE
ITZ ME YARN FUNC create_message WIT YARN name,

```

```
        ITZ ME YARN msg.

        msg IZ "hello, ".
        msg IZ CAT CAT "*" AN msg AN name.

        GIVE msg.
```

Listing 319: test/test_output/ast/fail_func5_syntax.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 320: test/test_output/semantic/fail_func5_syntax.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 321: test/test_output/full_pipeline/fail_func5_syntax.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 322: test/test_programs/fail_func6_syntax.meow

```
GIMME hello?

HAI ITZ ME YARN FUNC create_message WIT YARN name,

    ITZ ME NUMBR cookies IZ 0.

    PSST Its not possible to define a function within a function
    PSST This is only possible for classes
    HAI ITZ ME NUMBR FUNC Count_Cookies,
        GIVE cookies.
    KBYE
KBYE
```

Listing 323: test/test_output/ast/fail_func6_syntax.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 324: test/test_output/semantic/fail_func6_syntax.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 325: test/test_output/full_pipeline/fail_func6_syntax.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 326: test/test_programs/fail_if1.meow

```
HAI ITZ ME FUNC Main,
    PSST termination expr is not a Boolean

    ITZ ME YARN condition.

    "Pancake"
    O RLY?
    YA RLY HAI
        condition IZ "Waffles".
    KBYE
KBYE
```

Listing 327: test/test_output/ast/fail_if1.out

```


Main()
{
    char * condition;
    if ("Pancake")
    {
    condition = "Waffles";
    }
}
```

Listing 328: test/test_output/semantic/fail_if1.out

```
Fatal error: exception Exceptions.ControlFlowIllegalArgument("operation type mismatch: expected Boolean
    but got type char * "Pancake"")
```

Listing 329: test/test_output/full_pipeline/fail_if1.out

```
Fatal error: exception Exceptions.ControlFlowIllegalArgument("operation type mismatch: expected Boolean
    but got type char * "Pancake"")
```

Listing 330: test/test_programs/fail_if2.meow

```
HAI ITZ ME FUNC Main,
    PSST code block indicators missing - parsing error

    ITZ ME YARN condition.

    SAEM 3 AN 3
    O RLY?
    YA RLY
        condition IZ "Waffles".
        condition IZ CAT condition AN " and Pancakes".
```

```
    NO WAI
        condition IZ "Eggs".
        condition IZ CAT condition AN " and Bacon".
KBYE
```

Listing 331: test/test_output/ast/fail_if2.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 332: test/test_output/semantic/fail_if2.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 333: test/test_output/full_pipeline/fail_if2.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 334: test/test_programs/fail_if3.meow

```
HAI ITZ ME FUNC Main,
    PSST if statement within if statement missing conditional

    ITZ ME YARN condition.

    SAEM 3 AN 3
    O RLY?
    YA RLY HAI
        "Walnuts"
        O RLY?
        YA RLY HAI
            condition IZ "Pancakes".
        KBYE
    KBYE
    NO WAI HAI
        condition IZ "Eggs".
    KBYE
KBYE
```

Listing 335: test/test_output/ast/fail_if3.out

```


Main()
{
    char * condition;
    if (3 == 3) {
    if ("Walnuts")
    {
```

```
    condition = "Pancakes";
    }
    }
    else        {
    condition = "Eggs";
    }
}
```

Listing 336: test/test_output/semantic/fail_if3.out

```
Fatal error: exception Exceptions.ControlFlowIllegalArgument("operation type mismatch: expected Boolean
    but got type char * "Walnuts"")
```

Listing 337: test/test_output/full_pipeline/fail_if3.out

```
Fatal error: exception Exceptions.ControlFlowIllegalArgument("operation type mismatch: expected Boolean
    but got type char * "Walnuts"")
```

Listing 338: test/test_programs/fail_if4.meow

```
HAI ITZ ME FUNC Main,
    PSST else comes before if - parsing error

    ITZ ME YARN condition.

    SAEM 3 AN 3
    O RLY?
    NO WAI HAI
        condition IZ "Waffles".
    KBYE
    YA RLY HAI
        condition IZ "Eggs".
    KBYE
KBYE
```

Listing 339: test/test_output/ast/fail_if4.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 340: test/test_output/semantic/fail_if4.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 341: test/test_output/full_pipeline/fail_if4.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 342: test/test_programs/fail_import_not_found.meow

```
GIMME DOES_NOT_EXIST?

HAI ITZ ME FUNC Main,
    ITZ ME YARN green.
    ITZ ME YARN red.
KBYE
```

Listing 343: test/test_output/ast/fail_import_not_found.out

```
Fatal error: exception Exceptions.ImportNotFound("Could not find import")
```

Listing 344: test/test_output/semantic/fail_import_not_found.out

```
Fatal error: exception Exceptions.ImportNotFound("Could not find import")
```

Listing 345: test/test_output/full_pipeline/fail_import_not_found.out

```
Fatal error: exception Exceptions.ImportNotFound("Could not find import")
```

Listing 346: test/test_programs/fail_method1.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR jerrys_cookies.

    PSST You cannot call a class method without an instance
    PURR Give_Cookie IN Undeclared WIT 2.

    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 0.

    HAI ITZ ME NUMBR FUNC Count_Cookies,
        GIVE cookies.
    KBYE

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This uses the SUM prefix operator
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE
```

Listing 347: test/test_output/ast/fail_method1.out

```

Main()
{
    int jerrys_cookies;
    Undeclared.Give_Cookie(2);
    free(Jerry);
}

Class MOUSE {

    int cookies = 0;

    Give_Cookie(int count_cookies)
    {
    cookies = cookies + count_cookies;
    }
    int Count_Cookies()
    {
    return cookies;
    }

}
```

Listing 348: test/test_output/semantic/fail_method1.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier: Undeclared")
```

Listing 349: test/test_output/full_pipeline/fail_method1.out

```
Fatal error: exception Exceptions.VariableNotFound("undeclared identifier: Undeclared")
```

Listing 350: test/test_programs/fail_method2.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR jerrys_cookies.
    MAEK Jerry NEW MOUSE.

    PSST You cannot pass incorrect types to methods!
    PURR Give_Cookie IN Jerry WIT "string!".

    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,
```

```
14
15      ITZ ME NUMBR cookies IZ 0.
16
17      HAI ITZ ME NUMBR FUNC Count_Cookies,
18          GIVE cookies.
19      KBYE
20
21      HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
22          PSST This uses the SUM prefix operator
23          cookies IZ SUM OF cookies AN count_cookies.
24      KBYE
25
26  KBYE
```

Listing 351: test/test_output/ast/fail_method2.out

```
1
2
3   Main()
4   {
5       int jerrys_cookies;
6       class MOUSE Jerry;
7       Jerry.Give_Cookie("string!");
8       free(Jerry);
9   }
10
11  Class MOUSE {
12
13      int cookies = 0;
14
15      Give_Cookie(int count_cookies)
16      {
17      cookies = cookies + count_cookies;
18      }
19      int Count_Cookies()
20      {
21      return cookies;
22      }
23
24  }
```

Listing 352: test/test_output/semantic/fail_method2.out

```
1   Fatal error: exception Exceptions.ArgumentTypeMismatch("method Give_Cookie received arg of unexpected type
        : "string!", expected typ int, but got char *")
```

Listing 353: test/test_output/full_pipeline/fail_method2.out

```
Fatal error: exception Exceptions.ArgumentTypeMismatch("method Give_Cookie received arg of unexpected type
    : "string!", expected typ int, but got char *")
```

Listing 354: test/test_programs/fail_method3.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR jerrys_cookies IZ 3.
    MAEK Jerry NEW MOUSE.

    PSST You cannot pass an unexpected number of args to a method (TOO MANY)!
    PURR Give_Cookie IN Jerry WIT jerrys_cookies AN "string!".

    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 0.

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This uses the SUM prefix operator
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE
```

Listing 355: test/test_output/ast/fail_method3.out

```


Main()
{
    int jerrys_cookies = 3;
    class MOUSE Jerry;
    Jerry.Give_Cookie(jerrys_cookies, "string!");
    free(Jerry);
}

Class MOUSE {

    int cookies = 0;

    Give_Cookie(int count_cookies)
    {
    cookies = cookies + count_cookies;
    }
```

```

}
```

Listing 356: test/test_output/semantic/fail_method3.out

```
Fatal error: exception Exceptions.MethodArgumentLengthMismatch("expected different number of arguments for
      method: Give_Cookie (got 2, expected 1)")
```

Listing 357: test/test_output/full_pipeline/fail_method3.out

```
Fatal error: exception Exceptions.MethodArgumentLengthMismatch("expected different number of arguments for
      method: Give_Cookie (got 2, expected 1)")
```

Listing 358: test/test_programs/fail_method4.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR jerrys_cookies IZ 3.
    MAEK Jerry NEW MOUSE.

    PSST You cannot pass an unexpected number of args to a method (TOO FEW)!
    PURR Give_Cookie IN Jerry.

    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 0.

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This uses the SUM prefix operator
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE
```

Listing 359: test/test_output/ast/fail_method4.out

```


Main()
{
    int jerrys_cookies = 3;
    class MOUSE Jerry;
    Jerry.Give_Cookie();
    free(Jerry);
```

```
}

Class MOUSE {

    int cookies = 0;

    Give_Cookie(int count_cookies)
    {
    cookies = cookies + count_cookies;
    }

}
```

Listing 360: test/test_output/semantic/fail_method4.out

```
Fatal error: exception Exceptions.MethodArgumentLengthMismatch("expected different number of arguments for
      method: Give_Cookie (got 0, expected 1)")
```

Listing 361: test/test_output/full_pipeline/fail_method4.out

```
Fatal error: exception Exceptions.MethodArgumentLengthMismatch("expected different number of arguments for
      method: Give_Cookie (got 0, expected 1)")
```

Listing 362: test/test_programs/fail_method5.meow

```


HAI ITZ ME NUMBR FUNC Main,

    ITZ ME NUMBR size IZ 2.
    ITZ ME YARN cat_name.

    MAEK animals NEW BUCKET OF YARN HOLDS size,
        WIT "Cats"
        AN "Dogs".

    PSST integers have no methods
    cat_name IZ PURR Get_Name IN animals[0].

    BLEEP animals.
    GIVE 0.

KBYE
```

Listing 363: test/test_output/ast/fail_method5.out

```

```

```
int Main()
{
    int size = 2;
    char * cat_name;
    char * [size] animals = [ "Cats", "Dogs" ];
    cat_name = animals[0].Get_Name();
    free(animals);
    return 0;
}
```

Listing 364: test/test_output/semantic/fail_method5.out

```
Fatal error: exception Exceptions.InvalidMethodCall("methods can only be called on objects: found char *
    instead of Objtype in animals[0].Get_Name()")
```

Listing 365: test/test_output/full_pipeline/fail_method5.out

```
Fatal error: exception Exceptions.InvalidMethodCall("methods can only be called on objects: found char *
    instead of Objtype in animals[0].Get_Name()")
```

Listing 366: test/test_programs/fail_method6.meow

```

HAI ITZ ME NUMBR FUNC Main,

    ITZ ME NUMBR size IZ 2.
    ITZ ME YARN cat_name.

    MAEK animals NEW BUCKET OF YARN HOLDS size,
        WIT "Cats"
        AN "Dogs".

    PSST integers have no methods
    cat_name IZ PURR Get_Name IN size.

    BLEEP animals.
    GIVE 0.

KBYE
```

Listing 367: test/test_output/ast/fail_method6.out

```


int Main()
{
    int size = 2;
    char * cat_name;
```

```
    char * [size] animals = [ "Cats", "Dogs" ];
    cat_name = size.Get_Name();
    free(animals);
    return 0;
}
```

Listing 368: test/test_output/semantic/fail_method6.out

```
Fatal error: exception Exceptions.InvalidMethodCall("methods can only be called on objects: found int
    instead of Objtype in size.Get_Name()")
```

Listing 369: test/test_output/full_pipeline/fail_method6.out

```
Fatal error: exception Exceptions.InvalidMethodCall("methods can only be called on objects: found int
    instead of Objtype in size.Get_Name()")
```

Listing 370: test/test_programs/fail_method7.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR jerrys_cookies.
    MAEK Jerry NEW MOUSE.

    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 0.

    PSST this method returns the wrong type
    HAI ITZ ME YARN FUNC Count_Cookies,
        GIVE cookies.
    KBYE

KBYE
```

Listing 371: test/test_output/ast/fail_method7.out

```


Main()
{
    int jerrys_cookies;
    class MOUSE Jerry;
    free(Jerry);
}
```

```

Class MOUSE {

    int cookies = 0;

    char * Count_Cookies()
    {
    return cookies;
    }

}
```

Listing 372: test/test_output/semantic/fail_method7.out

```
Fatal error: exception Exceptions.ReturnTypeInvalid("attempting to return value that doesn't match
    function return type; expected: char *, got: int in function Count_Cookies")
```

Listing 373: test/test_output/full_pipeline/fail_method7.out

```
Fatal error: exception Exceptions.ReturnTypeInvalid("attempting to return value that doesn't match
    function return type; expected: char *, got: int in function Count_Cookies")
```

Listing 374: test/test_programs/fail_pass_wrong_obj_as_param.meow

```
HAI ITZ ME NUMBR FUNC Main,

    ITZ ME NUMBR jerrys_cookies IZ 0.

    MAEK Jerry NEW KITTY.

    PSST Call class method to set cookies
    PURR Set_Num_Cookies IN Jerry WIT 10.

    PURR Print_Cookies WIT Jerry.
    BLEEP Jerry.
    GIVE 0.
KBYE

HAI ITZ ME FUNC Print_Cookies WIT MOUSE mouse_guy,

    ITZ ME NUMBR cookies.

    cookies IZ PURR Get_Num_Cookies IN mouse_guy.

    PURR Meow WIT "The mouse has: ".
    PURR Meow WIT cookies.

KBYE
```

```
25
26 HAI ITZ ME CLASS KITTY,
27
28     ITZ ME NUMBR claws.
29
30     HAI ITZ ME NUMBR FUNC Mice_Eaten,
31         GIVE 10.
32     KBYE
33
34 KBYE
35
36 HAI ITZ ME CLASS MOUSE,
37
38     ITZ ME NUMBR cookies.
39
40     HAI ITZ ME FUNC Set_Num_Cookies WIT NUMBR cookies_given,
41         cookies IZ cookies_given.
42     KBYE
43
44 KBYE
```

Listing 375: test/test_output/ast/fail_pass_wrong_obj_as_param.out

```
1
2
3  Print_Cookies(class MOUSE mouse_guy)
4  {
5      int cookies;
6      cookies = mouse_guy.Get_Num_Cookies();
7      printf("%X\n", "The mouse has: ");
8      printf("%X\n", cookies);
9  }
10
11 int Main()
12 {
13     int jerrys_cookies = 0;
14     class KITTY Jerry;
15     Jerry.Set_Num_Cookies(10);
16     Print_Cookies(Jerry);
17     free(Jerry);
18     return 0;
19 }
20
21 Class MOUSE {
22
23     int cookies;
24
25     Set_Num_Cookies(int cookies_given)
```

```
    {
    cookies = cookies_given;
    }

}

Class KITTY {

    int claws;

    int Mice_Eaten()
    {
    return 10;
    }

}
```

Listing 376: test/test_output/semantic/fail_pass_wrong_obj_as_param.out

```
Fatal error: exception Exceptions.ClassMethodNotFound("method does not exist for this class: MOUSE.
    Get_Num_Cookies")
```

Listing 377: test/test_output/full_pipeline/fail_pass_wrong_obj_as_param.out

```
Fatal error: exception Exceptions.ClassMethodNotFound("method does not exist for this class: MOUSE.
    Get_Num_Cookies")
```

Listing 378: test/test_programs/fail_scan_too_many_args.meow

```

HAI ITZ ME FUNC Main,

        PSST Creating a string to hold read content
        ITZ ME YARN user_message.
        ITZ ME YARN another_message.

        PSST Reading in content
        PURR Meow WIT "Please tell me your favorite color: ".
        PURR Scan WIT user_message AN another_message.

        PURR Meow WIT "Your favorite color is: ".
        PURR Meow WIT user_message.

KBYE
```

Listing 379: test/test_output/ast/fail_scan_too_many_args.out

```

```

```

Main()
{
    char * user_message;
    char * another_message;
    printf("%X\n", "Please tell me your favorite color: ");
    Scan(user_message, another_message);
    printf("%X\n", "Your favorite color is: ");
    printf("%X\n", user_message);
}
```

Listing 380: test/test_output/semantic/fail_scan_too_many_args.out

```
Fatal error: exception Exceptions.FunctionArgumentLengthMismatch("expected different number of arguments
    for function: Scan")
```

Listing 381: test/test_output/full_pipeline/fail_scan_too_many_args.out

```
Fatal error: exception Exceptions.FunctionArgumentLengthMismatch("expected different number of arguments
    for function: Scan")
```

Listing 382: test/test_programs/fail_scan_wrong_type.meow

```

HAI ITZ ME FUNC Main,

        PSST Creating a string to hold read content
        ITZ ME NUMBR user_message.

        PSST Reading in content
        PURR Meow WIT "Please tell me your favorite color: ".
        PURR Scan WIT user_message.

        PURR Meow WIT "Your favorite color is: ".
        PURR Meow WIT user_message.

KBYE
```

Listing 383: test/test_output/ast/fail_scan_wrong_type.out

```


Main()
{
    int user_message;
    printf("%X\n", "Please tell me your favorite color: ");
    Scan(user_message);
    printf("%X\n", "Your favorite color is: ");
```

```
    printf("%X\n", user_message);
}
```

Listing 384: test/test_output/semantic/fail_scan_wrong_type.out

```
Fatal error: exception Exceptions.ArgumentTypeMismatch("built in scan function takes only the id of a
    string as an argument")
```

Listing 385: test/test_output/full_pipeline/fail_scan_wrong_type.out

```
Fatal error: exception Exceptions.ArgumentTypeMismatch("built in scan function takes only the id of a
    string as an argument")
```

Listing 386: test/test_programs/fail_syntax_comment.meow

```

HAI ITZ ME YARN FUNC Main,

        (*  This is not how you write a comment in meowlang *)

KBYE
```

Listing 387: test/test_output/ast/fail_syntax_comment.out

```
Fatal error: exception Scanner.SyntaxError("Illegal character: '('")
```

Listing 388: test/test_output/semantic/fail_syntax_comment.out

```
Fatal error: exception Scanner.SyntaxError("Illegal character: '('")
```

Listing 389: test/test_output/full_pipeline/fail_syntax_comment.out

```
Fatal error: exception Scanner.SyntaxError("Illegal character: '('")
```

Listing 390: test/test_programs/fail_syntax_variables.meow

```

HAI ITZ ME YARN FUNC Main,

        PSST Initialize a variable incorrectly
        ITZ NUMBR count.
KBYE
```

Listing 391: test/test_output/ast/fail_syntax_variables.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 392: test/test_output/semantic/fail_syntax_variables.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 393: test/test_output/full_pipeline/fail_syntax_variables.out

```
Fatal error: exception Stdlib.Parsing.Parse_error
```

Listing 394: test/test_programs/fail_unop.meow

```
HAI ITZ ME FUNC Main,

    PSST Needs to be boolean value
    ITZ ME NUMBR num IZ 2.
    NOT num.

KBYE
```

Listing 395: test/test_output/ast/fail_unop.out

```


Main()
{
    int num = 2;
    !num;
}
```

Listing 396: test/test_output/semantic/fail_unop.out

```
Fatal error: exception Exceptions.IllegalUnaryOp("!num")
```

Listing 397: test/test_output/full_pipeline/fail_unop.out

```
Fatal error: exception Exceptions.IllegalUnaryOp("!num")
```

Listing 398: test/test_programs/fail_variables_int_to_str.meow

```

HAI ITZ ME YARN FUNC Main,

        PSST Creating some local variables
        ITZ ME NUMBR count.
        ITZ ME NUMBAR random.
        ITZ ME BOO say_hello.
        ITZ ME YARN msg.

        PSST Assigning local varibles values
        msg IZ 2.                       PSST ** This should be a string **
```

```
        say_hello IZ AYE.
        count IZ 2.
        random IZ 2.34.

        GIVE msg.  PSST Returning Hello to user
KBYE
```

Listing 399: test/test_output/ast/fail_variables_int_to_str.out

```


char * Main()
{
    int count;
    float random;
    bool say_hello;
    char * msg;
    msg = 2;
    say_hello = true;
    count = 2;
    random = 2.34;
    return msg;
}
```

Listing 400: test/test_output/semantic/fail_variables_int_to_str.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
    expected type. expected char *, got int here: msg = 2")
```

Listing 401: test/test_output/full_pipeline/fail_variables_int_to_str.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
    expected type. expected char *, got int here: msg = 2")
```

Listing 402: test/test_programs/fail_variables_str_to_bool.meow

```

HAI ITZ ME YARN FUNC Main,

        PSST Creating some local variables
        ITZ ME NUMBR count.
        ITZ ME NUMBAR random.
        ITZ ME BOO say_hello.
        ITZ ME YARN msg.

        PSST Assigning local varibles values
        msg IZ "test message".
        say_hello IZ "I shouldn't be here".  PSST This should be a bool value
```

```
        count IZ 2.
        random IZ 2.34.

        GIVE msg.  PSST Returning Hello to user
KBYE
```

Listing 403: test/test_output/ast/fail_variables_str_to_bool.out

```


char * Main()
{
    int count;
    float random;
    bool say_hello;
    char * msg;
    msg = "test message";
    say_hello = "I shouldn't be here";
    count = 2;
    random = 2.34;
    return msg;
}
```

Listing 404: test/test_output/semantic/fail_variables_str_to_bool.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
    expected type. expected bool, got char * here: say_hello = "I shouldn't be here"")
```

Listing 405: test/test_output/full_pipeline/fail_variables_str_to_bool.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
    expected type. expected bool, got char * here: say_hello = "I shouldn't be here"")
```

Listing 406: test/test_programs/fail_variables_str_to_int.meow

```

HAI ITZ ME YARN FUNC Main,

        PSST Creating some local variables
        ITZ ME NUMBR count.
        ITZ ME NUMBAR random.
        ITZ ME BOO say_hello.
        ITZ ME YARN msg.

        PSST Assigning local varibles values
        msg IZ "hello world".
        say_hello IZ AYE.
        count IZ "hi".                  PSST ** This should be an integer **
```

```
        random IZ 2.34.

        GIVE msg.  PSST Returning Hello to user
KBYE
```

Listing 407: test/test_output/ast/fail_variables_str_to_int.out

```


char * Main()
{
    int count;
    float random;
    bool say_hello;
    char * msg;
    msg = "hello world";
    say_hello = true;
    count = "hi";
    random = 2.34;
    return msg;
}
```

Listing 408: test/test_output/semantic/fail_variables_str_to_int.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
    expected type. expected int, got char * here: count = "hi"")
```

Listing 409: test/test_output/full_pipeline/fail_variables_str_to_int.out

```
Fatal error: exception Exceptions.VariableAssignmentError("variables can only be assigned to items of the
    expected type. expected int, got char * here: count = "hi"")
```

Listing 410: test/test_programs/fstphrase.meow

```
HAI ITZ ME FUNC Fstphrase,
    PURR Meow WIT "My favorite color is blue ".
KBYE
```

Listing 411: test/test_programs/import_bridge.meow

```
GIMME FSTPHRASE?
GIMME SNDPHRASE?
```

Listing 412: test/test_programs/import_colors.meow

```

HAI ITZ ME FUNC Colors,
    ITZ ME YARN green IZ "green".
    ITZ ME YARN red IZ "red".
KBYE
```

Listing 413: test/test_programs/import_example.meow

```
GIMME INNER_IMPORT_EXAMPLE?

HAI ITZ ME FUNC Say_Hello,

        ITZ ME YARN username.
        ITZ ME YARN message.

        username IZ PURR Get_User_Name.
        message IZ CAT "hello, " AN username.
        PURR Meow WIT message.

        PSST CAT and SCAN allocate strings on heap
        BLEEP message.
        BLEEP username.
KBYE
```

Listing 414: test/test_programs/inner_import_example.meow

```
HAI ITZ ME YARN FUNC Get_User_Name,

        ITZ ME YARN username.

        PURR Meow WIT "What is your name? ".
        PURR Scan WIT username.

        GIVE username.
KBYE
```

Listing 415: test/test_programs/multiple_imports.meow

```
GIMME IMPORT_EXAMPLE?
GIMME IMPORT_COLORS?
GIMME IMPORT_BRIDGE?

HAI ITZ ME NUMBR FUNC Main,
        PURR Say_Hello.
        PURR Sndphrase.
        PURR Fstphrase.
        GIVE 0.
KBYE
```

Listing 416: test/test_programs/sndphrase.meow

```
HAI ITZ ME FUNC Sndphrase,
    PURR Meow WIT "My least favorite color is yellow".
    PURR Get_User_Name.
KBYE
```

Listing 417: test/test_programs/test_array_access1.meow

```
1
2  HAI ITZ ME FUNC Main,
3
4      ITZ ME NUMBR idx IZ 0.
5
6      MAEK cool_array NEW BUCKET OF NUMBR HOLDS 3,
7              WIT 1 AN 2 AN 3.
8
9      PSST set new value in array
10     cool_array[idx] IZ 2.
11
12     PSST Print value at index 1
13     PURR Meow WIT cool_array[idx].
14 KBYE
```

Listing 418: test/test_output/ast/test_array_access1.out

```
1
2
3  Main()
4  {
5      int idx = 0;
6      int [3] cool_array = [ 1, 2, 3 ];
7      cool_array[idx] = 2;
8      printf("%X\n", cool_array[idx]);
9  }
```

Listing 419: test/test_output/semantic/test_array_access1.out

```
1  Semantic check succeded!
```

Listing 420: test/test_output/full_pipeline/test_array_access1.out

```
1  2
```

Listing 421: test/test_programs/test_array_access2.meow

```
1
2  HAI ITZ ME FUNC Main,
3
4      ITZ ME YARN str1 IZ "whoooopie!".
5      ITZ ME YARN str2 IZ "yiiippeee!".
6      ITZ ME YARN recovered_str.
7
8      PSST Create simple array with elements that are variables
9      MAEK exclamations NEW BUCKET OF YARN HOLDS 3,
10         WIT str1
```

```
        AN str2.

    recovered_str IZ exclamations[1].
    PURR Meow WIT recovered_str.
    BLEEP exclamations.

KBYE
```

Listing 422: test/test_output/ast/test_array_access2.out

```


Main()
{
    char * str1 = "whoooopie!";
    char * str2 = "yiiippeee!";
    char * recovered_str;
    char * [3] exclamations = [ str1, str2 ];
    recovered_str = exclamations[1];
    printf("%X\n", recovered_str);
    free(exclamations);
}
```

Listing 423: test/test_output/semantic/test_array_access2.out

```
Semantic check succeded!
```

Listing 424: test/test_output/full_pipeline/test_array_access2.out

```
yiiippeee!
```

Listing 425: test/test_programs/test_array_access3.meow

```

HAI ITZ ME FUNC Main,

    ITZ ME YARN str1 IZ "whoooopie!".
    ITZ ME YARN str2 IZ "yiiippeee!".
    ITZ ME YARN recovered_str.
    ITZ ME NUMBR index IZ 0.

    PSST Create simple array with elements that are variables
    MAEK exclamations NEW BUCKET OF YARN HOLDS 3,
        WIT str1
        AN str2.

    recovered_str IZ exclamations[index].
    PURR Meow WIT recovered_str.
```

```
    BLEEP exclamations.

KBYE
```

Listing 426: test/test_output/ast/test_array_access3.out

```


Main()
{
    char * str1 = "whoooopie!";
    char * str2 = "yiiippeee!";
    char * recovered_str;
    int index = 0;
    char * [3] exclamations = [ str1, str2 ];
    recovered_str = exclamations[index];
    printf("%X\n", recovered_str);
    free(exclamations);
}
```

Listing 427: test/test_output/semantic/test_array_access3.out

```
Semantic check succeded!
```

Listing 428: test/test_output/full_pipeline/test_array_access3.out

```
whoooopie!
```

Listing 429: test/test_programs/test_array_access4.meow

```

HAI ITZ ME CLASS PET,

    ITZ ME YARN name.
    ITZ ME YARN type.
    ITZ ME NUMBR age.

    HAI ITZ ME YARN FUNC Get_Name,
        GIVE name.
    KBYE

KBYE


HAI ITZ ME NUMBR FUNC Main,

    ITZ ME NUMBR size IZ 2.
    ITZ ME NUMBR idx IZ 1.
```

```
    ITZ ME PET Cat.
    ITZ ME YARN cat_name.

    PSST Make some pets
    MAEK Silvester NEW PET,
        WIT name IZ "Silvester"
        AN type IZ "cat"
        AN age IZ 4.

    MAEK Tank NEW PET,
        WIT name IZ "Tank"
        AN type IZ "dog"
        AN age IZ 2.

    PSST Make an array of objects
    MAEK my_pets NEW BUCKET OF PET HOLDS size,
        WIT Silvester
        AN Tank.

    PSST Make a printer instance an print element at index 'idx'
    cat_name IZ PURR Get_Name IN my_pets[0].
    PURR Meow WIT cat_name.

    BLEEP my_pets.
    BLEEP Silvester.
    BLEEP Tank.
    GIVE 0.

KBYE
```

Listing 430: test/test_output/ast/test_array_access4.out

```


int Main()
{
    int size = 2;
    int idx = 1;
    class PET Cat;
    char * cat_name;
    class PET Silvester(age = 4, type = "cat", name = "Silvester", );
    class PET Tank(age = 2, type = "dog", name = "Tank", );
    class PET [size] my_pets = [ Silvester, Tank ];
    cat_name = my_pets[0].Get_Name();
    printf("%X\n", cat_name);
    free(my_pets);
    free(Silvester);
    free(Tank);
```

```
    return 0;
}

Class PET {

    int age;
    char * type;
    char * name;

    char * Get_Name()
    {
    return name;
    }

}
```

Listing 431: test/test_output/semantic/test_array_access4.out

```
Semantic check succeded!
```

Listing 432: test/test_output/full_pipeline/test_array_access4.out

```
Silvester
```

Listing 433: test/test_programs/test_array_access5.meow

```

HAI ITZ ME CLASS PET,

    ITZ ME YARN name.
    ITZ ME YARN type.
    ITZ ME NUMBR age.

    HAI ITZ ME YARN FUNC Get_Name,
        GIVE name.
    KBYE

KBYE


HAI ITZ ME NUMBR FUNC Main,

    ITZ ME NUMBR size IZ 2.
    ITZ ME NUMBR idx IZ 1.
    ITZ ME PET Cat.
    ITZ ME YARN cat_name.

    PSST Make some pets
```

```
    MAEK Silvester NEW PET,
        WIT name IZ "Silvester"
        AN type IZ "cat"
        AN age IZ 4.

    MAEK Tank NEW PET,
        WIT name IZ "Tank"
        AN type IZ "dog"
        AN age IZ 2.

    PSST Make an array of objects
    MAEK my_pets NEW BUCKET OF PET HOLDS size.

    my_pets[0] IZ Silvester.

    PSST Make a printer instance an print element at index 'idx'
    cat_name IZ PURR Get_Name IN my_pets[0].
    PURR Meow WIT cat_name.

    BLEEP my_pets.
    BLEEP Silvester.
    BLEEP Tank.
    GIVE 0.

KBYE
```

Listing 434: test/test_output/ast/test_array_access5.out

```


int Main()
{
    int size = 2;
    int idx = 1;
    class PET Cat;
    char * cat_name;
    class PET Silvester(age = 4, type = "cat", name = "Silvester", );
    class PET Tank(age = 2, type = "dog", name = "Tank", );
    class PET [size] my_pets = [  ];
    my_pets[0] = Silvester;
    cat_name = my_pets[0].Get_Name();
    printf("%X\n", cat_name);
    free(my_pets);
    free(Silvester);
    free(Tank);
    return 0;
}

```

```
Class PET {

    int age;
    char * type;
    char * name;

    char * Get_Name()
    {
    return name;
    }

}
```

Listing 435: test/test_output/semantic/test_array_access5.out

```
Semantic check succeded!
```

Listing 436: test/test_output/full_pipeline/test_array_access5.out

```
Silvester
```

Listing 437: test/test_programs/test_array_assignment1.meow

```

HAI ITZ ME FUNC Main,

    ITZ ME YARN str1 IZ "whoooopie!".
    ITZ ME YARN str2 IZ "yiiippeee!".
    ITZ ME YARN recovered_str.
    ITZ ME NUMBR index IZ 0.

    PSST Create simple array with elements that are variables
    MAEK exclamations NEW BUCKET OF YARN HOLDS 3,
        WIT str1
        AN str2.

    exclamations[index] IZ "new value!".

    PURR Meow WIT exclamations[index].
    BLEEP exclamations.

KBYE
```

Listing 438: test/test_output/ast/test_array_assignment1.out

```


Main()
```

```
{
    char * str1 = "whoooopie!";
    char * str2 = "yiiippeee!";
    char * recovered_str;
    int index = 0;
    char * [3] exclamations = [ str1, str2 ];
    exclamations[index] = "new value!";
    printf("%X\n", exclamations[index]);
    free(exclamations);
}
```

Listing 439: test/test_output/semantic/test_array_assignment1.out

```
Semantic check succeded!
```

Listing 440: test/test_output/full_pipeline/test_array_assignment1.out

```
new value!
```

Listing 441: test/test_programs/test_array_assignment2.meow

```

HAI ITZ ME FUNC Main,

    ITZ ME NUMBR value1 IZ 10.
    ITZ ME NUMBR value2 IZ 11.
    ITZ ME NUMBR value3 IZ 12.
    ITZ ME NUMBR index IZ 2.

    PSST Create simple array with elements that are variables
    MAEK values NEW BUCKET OF NUMBR HOLDS 3,
        WIT value1
        AN value2.

    values[index] IZ value3.

    PURR Meow WIT values[index].
    BLEEP values.

KBYE
```

Listing 442: test/test_output/ast/test_array_assignment2.out

```


Main()
{
    int value1 = 10;
```

```
    int value2 = 11;
    int value3 = 12;
    int index = 2;
    int [3] values = [ value1, value2 ];
    values[index] = value3;
    printf("%X\n", values[index]);
    free(values);
}
```

Listing 443: test/test_output/semantic/test_array_assignment2.out

```
Semantic check succeded!
```

Listing 444: test/test_output/full_pipeline/test_array_assignment2.out

```
12
```

Listing 445: test/test_programs/test_array_assignment3.meow

```

HAI ITZ ME FUNC Main,

    ITZ ME YARN str1 IZ "whoooopie!".
    ITZ ME YARN str2 IZ "yiiippeee!".
    ITZ ME YARN str3 IZ "yayy".
    ITZ ME NUMBR index IZ 1.

    PSST Create simple array with elements that are variables
    MAEK exclamations NEW BUCKET OF YARN HOLDS 3,
        WIT str1
        AN str2.

    exclamations[index] IZ str3.

    PURR Meow WIT exclamations[index].
    BLEEP exclamations.
KBYE
```

Listing 446: test/test_output/ast/test_array_assignment3.out

```


Main()
{
    char * str1 = "whoooopie!";
    char * str2 = "yiiippeee!";
    char * str3 = "yayy";
    int index = 1;
```

```
    char * [3] exclamations = [ str1, str2 ];
    exclamations[index] = str3;
    printf("%X\n", exclamations[index]);
    free(exclamations);
}
```

Listing 447: test/test_output/semantic/test_array_assignment3.out

```
Semantic check succeded!
```

Listing 448: test/test_output/full_pipeline/test_array_assignment3.out

```
yayy
```

Listing 449: test/test_programs/test_array_assignment4.meow

```
HAI ITZ ME BUCKET OF YARN FUNC Create_Array WIT YARN item1 AN YARN item2,

   ITZ ME NUMBR count IZ 3.

    MAEK string_array NEW BUCKET OF YARN HOLDS count,
        WIT item1
        AN item2.

   GIVE string_array.
KBYE

HAI ITZ ME FUNC Main,

   ITZ ME BUCKET OF YARN fruits.
   ITZ ME YARN apple IZ "apples".
   ITZ ME YARN pear IZ "pears".
   ITZ ME YARN recovered_str.

   PSST you cannot do an index assignment on something other than arrays
   fruits IZ PURR Create_Array WIT apple AN pear.

   recovered_str IZ fruits[1].
   PURR Meow WIT recovered_str.
   BLEEP fruits.

KBYE
```

Listing 450: test/test_output/ast/test_array_assignment4.out

```


Main()
```

```
{
    char * [] fruits;
    char * apple = "apples";
    char * pear = "pears";
    char * recovered_str;
    fruits = Create_Array(apple, pear);
    recovered_str = fruits[1];
    printf("%X\n", recovered_str);
    free(fruits);
}

char * [] Create_Array(char * item1, char * item2)
{
    int count = 3;
    char * [count] string_array = [ item1, item2 ];
    return string_array;
}
```

Listing 451: test/test_output/semantic/test_array_assignment4.out

```
Semantic check succeded!
```

Listing 452: test/test_output/full_pipeline/test_array_assignment4.out

```
pears
```

Listing 453: test/test_programs/test_cast_float_to_int.meow

```

HAI ITZ ME NUMBR FUNC Main,

        ITZ ME NUMBR count.
        ITZ ME NUMBAR float_count IZ 2.34534534.

        count IZ NUMBR float_count.
        PURR Meow WIT count.

        GIVE 0.
KBYE
```

Listing 454: test/test_output/ast/test_cast_float_to_int.out

```


int Main()
{
    int count;
    float float_count = 2.34534534;
```

```
    count = (int) float_count;
    printf("%X\n", count);
    return 0;
}
```

Listing 455: test/test_output/semantic/test_cast_float_to_int.out

```
Semantic check succeded!
```

Listing 456: test/test_output/full_pipeline/test_cast_float_to_int.out

```
2
```

Listing 457: test/test_programs/test_cast_float_to_str.meow

```

HAI ITZ ME NUMBR FUNC Main,

        ITZ ME NUMBAR value IZ 2.234.
        ITZ ME YARN str.

        str IZ YARN value.
        PURR Meow WIT str.  PSST 2.234
        BLEEP str.          PSST need to free memory

        GIVE 0.
KBYE
```

Listing 458: test/test_output/ast/test_cast_float_to_str.out

```


int Main()
{
    float value = 2.234;
    char * str;
    str = (char *) value;
    printf("%X\n", str);
    free(str);
    return 0;
}
```

Listing 459: test/test_output/semantic/test_cast_float_to_str.out

```
Semantic check succeded!
```

Listing 460: test/test_output/full_pipeline/test_cast_float_to_str.out

```
2.234000
```

Listing 461: test/test_programs/test_cast_int_to_float.meow

```
1
2  HAI ITZ ME NUMBR FUNC Main,
3
4          ITZ ME NUMBR count IZ 2.
5          ITZ ME NUMBAR float_count.
6
7          float_count IZ NUMBAR count.
8          PURR Meow WIT float_count.
9
10         GIVE 0.
11 KBYE
```

Listing 462: test/test_output/ast/test_cast_int_to_float.out

```
1
2
3  int Main()
4  {
5      int count = 2;
6      float float_count;
7      float_count = (float) count;
8      printf("%X\n", float_count);
9      return 0;
10 }
```

Listing 463: test/test_output/semantic/test_cast_int_to_float.out

```
1  Semantic check succeded!
```

Listing 464: test/test_output/full_pipeline/test_cast_int_to_float.out

```
1  2
```

Listing 465: test/test_programs/test_cast_int_to_str.meow

```
1
2  HAI ITZ ME NUMBR FUNC Main,
3
4          ITZ ME NUMBR count IZ 203423.
5          ITZ ME YARN value.
6
7          PURR Meow WIT count.
8          value IZ YARN count.  PSST Do Cast
9
10         PURR Meow WIT value.
11         BLEEP value.          PSST Memory allocated so need to free
12         GIVE 0.
13 KBYE
```

Listing 466: test/test_output/ast/test_cast_int_to_str.out

```

int Main()
{
    int count = 203423;
    char * value;
    printf("%X\n", count);
    value = (char *) count;
    printf("%X\n", value);
    free(value);
    return 0;
}
```

Listing 467: test/test_output/semantic/test_cast_int_to_str.out

```
Semantic check succeded!
```

Listing 468: test/test_output/full_pipeline/test_cast_int_to_str.out

```
203423
203423
```

Listing 469: test/test_programs/test_cast_str_to_float.meow

```

HAI ITZ ME NUMBR FUNC Main,

        ITZ ME NUMBAR value.
        ITZ ME YARN str IZ "2.234".

        value IZ NUMBAR str.
        PURR Meow WIT SUM OF value AN 2.  PSST 4.234

        GIVE 0.
KBYE
```

Listing 470: test/test_output/ast/test_cast_str_to_float.out

```

int Main()
{
    float value;
    char * str = "2.234";
    value = (float) str;
    printf("%X\n", value + 2);
    return 0;
}
```

Listing 471: test/test_output/semantic/test_cast_str_to_float.out

```
Semantic check succeded!
```

Listing 472: test/test_output/full_pipeline/test_cast_str_to_float.out

```
4.234
```

Listing 473: test/test_programs/test_cast_str_to_int.meow

```

HAI ITZ ME NUMBR FUNC Main,

        PSST Creating some local variables
        ITZ ME NUMBR count.
        ITZ ME YARN value IZ "2".

        PSST Assigning local varibles values
        count IZ NUMBR value.

        PSST Test printing an integer
        PURR Meow WIT SUM OF count AN 2.

        GIVE 0.
KBYE
```

Listing 474: test/test_output/ast/test_cast_str_to_int.out

```


int Main()
{
    int count;
    char * value = "2";
    count = (int) value;
    printf("%X\n", count + 2);
    return 0;
}
```

Listing 475: test/test_output/semantic/test_cast_str_to_int.out

```
Semantic check succeded!
```

Listing 476: test/test_output/full_pipeline/test_cast_str_to_int.out

```
4
```

Listing 477: test/test_programs/test_class_default_vars.meow

```
1
2  HAI ITZ ME FUNC Main,
3
4      ITZ ME NUMBR count_treats.
5      MAEK Jerry NEW MOUSE.
6      count_treats IZ PURR Get_Num_Treats IN Jerry.
7      PURR Meow WIT count_treats.
8      BLEEP Jerry.
9
10 KBYE
11
12 HAI ITZ ME CLASS MOUSE,
13
14     ITZ ME NUMBR cookies IZ 2.
15     ITZ ME NUMBR candies IZ 5.
16     ITZ ME NUMBR treats IZ SUM OF cookies AN candies.
17
18     HAI ITZ ME NUMBR FUNC Get_Num_Treats,
19         GIVE treats.
20     KBYE
21
22 KBYE
```

Listing 478: test/test_output/ast/test_class_default_vars.out

```
1
2
3  Main()
4  {
5      int count_treats;
6      class MOUSE Jerry;
7      count_treats = Jerry.Get_Num_Treats();
8      printf("%X\n", count_treats);
9      free(Jerry);
10 }
11
12 Class MOUSE {
13
14     int treats = cookies + candies;
15     int candies = 5;
16     int cookies = 2;
17
18     int Get_Num_Treats()
19     {
20     return treats;
21     }
22
```

```
}
```

Listing 479: test/test_output/semantic/test_class_default_vars.out

```
Semantic check succeded!
```

Listing 480: test/test_output/full_pipeline/test_class_default_vars.out

```
7
```

Listing 481: test/test_programs/test_class_inner_method_call.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR jerrys_cookies IZ 0.

    MAEK Jerry NEW MOUSE.

    PSST Call class method to set cookies
    PURR Meow WIT "Setting Jerry's cookies count to 10".
    PURR Set_Num_Cookies IN Jerry WIT 10.

    PSST Call class method to increment cookies by 1
    PURR Incr_Cookies IN Jerry.

    PSST Use class method to Get New Number of Cookies
    jerrys_cookies IZ PURR Get_Num_Cookies IN Jerry.
    BLEEP Jerry.

    PURR Meow WIT "Incremented cookies by 1, now Jerry has: ".
    PURR Meow WIT jerrys_cookies.

KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies.

    HAI ITZ ME FUNC Set_Num_Cookies WIT NUMBR cookies_given,
        cookies IZ cookies_given.
    KBYE

    HAI ITZ ME NUMBR FUNC Get_Num_Cookies,
        GIVE cookies.
    KBYE

    HAI ITZ ME FUNC Incr_Cookies,
        PSST Testing that this method call works correctly
```

```
        ITZ ME NUMBR existing_cookies IZ PURR Get_Num_Cookies IN HERE.
        cookies IZ SUM OF existing_cookies AN 1.
    KBYE

KBYE
```

Listing 482: test/test_output/ast/test_class_inner_method_call.out

```


Main()
{
    int jerrys_cookies = 0;
    class MOUSE Jerry;
    printf("%X\n", "Setting Jerry's cookies count to 10");
    Jerry.Set_Num_Cookies(10);
    Jerry.Incr_Cookies();
    jerrys_cookies = Jerry.Get_Num_Cookies();
    free(Jerry);
    printf("%X\n", "Incremented cookies by 1, now Jerry has: ");
    printf("%X\n", jerrys_cookies);
}

Class MOUSE {

    int cookies;

    Incr_Cookies()
    {
    int existing_cookies = this.Get_Num_Cookies();
    cookies = existing_cookies + 1;
    }
    int Get_Num_Cookies()
    {
    return cookies;
    }
    Set_Num_Cookies(int cookies_given)
    {
    cookies = cookies_given;
    }

}
```

Listing 483: test/test_output/semantic/test_class_inner_method_call.out

```
Semantic check succeded!
```

Listing 484: test/test_output/full_pipeline/test_class_inner_method_call.out

```
Setting Jerry's cookies count to 10
Incremented cookies by 1, now Jerry has:
11
```

Listing 485: test/test_programs/test_class_specify_constructor.meow

```

HAI ITZ ME FUNC Main,

    ITZ ME NUMBR count_cookies IZ 100.
    ITZ ME NUMBR count_candies IZ 49.
    ITZ ME NUMBR count_treats IZ SUM OF count_cookies AN count_candies.

    PSST dont use any of the default valuesre
    MAEK Jerry NEW MOUSE,
        WIT treats IZ count_treats
        AN cookies IZ count_cookies
        AN candies IZ count_candies.

    count_treats IZ PURR Get_Num_Treats IN Jerry.
    PURR Meow WIT count_treats.

    count_cookies IZ PURR Get_Num_Cookies IN Jerry.
    PURR Meow WIT count_cookies.

    count_candies IZ PURR Get_Num_Candies IN Jerry.
    PURR Meow WIT count_candies.

    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 2.
    ITZ ME NUMBR candies IZ 5.
    ITZ ME NUMBR treats IZ SUM OF cookies AN candies.

    HAI ITZ ME NUMBR FUNC Get_Num_Treats,
        GIVE treats.
    KBYE

   HAI ITZ ME NUMBR FUNC Get_Num_Cookies,
        GIVE cookies.
    KBYE

   HAI ITZ ME NUMBR FUNC Get_Num_Candies,
```

```
        GIVE candies.
    KBYE

KBYE
```

Listing 486: test/test_output/ast/test_class_specify_constructor.out

```


Main()
{
    int count_cookies = 100;
    int count_candies = 49;
    int count_treats = count_cookies + count_candies;
    class MOUSE Jerry(candies = count_candies, cookies = count_cookies, treats = count_treats, );
    count_treats = Jerry.Get_Num_Treats();
    printf("%X\n", count_treats);
    count_cookies = Jerry.Get_Num_Cookies();
    printf("%X\n", count_cookies);
    count_candies = Jerry.Get_Num_Candies();
    printf("%X\n", count_candies);
    free(Jerry);
}

Class MOUSE {

    int treats = cookies + candies;
    int candies = 5;
    int cookies = 2;

    int Get_Num_Candies()
    {
    return candies;
    }
    int Get_Num_Cookies()
    {
    return cookies;
    }
    int Get_Num_Treats()
    {
    return treats;
    }

}
```

Listing 487: test/test_output/semantic/test_class_specify_constructor.out

```
Semantic check succeded!
```

Listing 488: test/test_output/full_pipeline/test_class_specify_constructor.out

```
149
100
49
```

Listing 489: test/test_programs/test_class_specify_constructor_some.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR count_cookies IZ 100.
    ITZ ME NUMBR count_candies IZ 49.
    ITZ ME NUMBR count_treats IZ SUM OF count_cookies AN count_candies.

    PSST dont use any of the default valuesre
    MAEK Jerry NEW MOUSE,
        WIT treats IZ count_treats.

    count_treats IZ PURR Get_Num_Treats IN Jerry.
    PURR Meow WIT count_treats.

    count_cookies IZ PURR Get_Num_Cookies IN Jerry.
    PURR Meow WIT count_cookies.

    count_candies IZ PURR Get_Num_Candies IN Jerry.
    PURR Meow WIT count_candies.

    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 2.
    ITZ ME NUMBR candies IZ 5.
    ITZ ME NUMBR treats IZ SUM OF cookies AN candies.

    HAI ITZ ME NUMBR FUNC Get_Num_Treats,
        GIVE treats.
    KBYE

   HAI ITZ ME NUMBR FUNC Get_Num_Cookies,
        GIVE cookies.
    KBYE

   HAI ITZ ME NUMBR FUNC Get_Num_Candies,
        GIVE candies.
    KBYE
```

```

KBYE
```

Listing 490: test/test_output/ast/test_class_specify_constructor_some.out

```
Main()
{
    int count_cookies = 100;
    int count_candies = 49;
    int count_treats = count_cookies + count_candies;
    class MOUSE Jerry(treats = count_treats, );
    count_treats = Jerry.Get_Num_Treats();
    printf("%X\n", count_treats);
    count_cookies = Jerry.Get_Num_Cookies();
    printf("%X\n", count_cookies);
    count_candies = Jerry.Get_Num_Candies();
    printf("%X\n", count_candies);
    free(Jerry);
}

Class MOUSE {

    int treats = cookies + candies;
    int candies = 5;
    int cookies = 2;

    int Get_Num_Candies()
    {
    return candies;
    }
    int Get_Num_Cookies()
    {
    return cookies;
    }
    int Get_Num_Treats()
    {
    return treats;
    }

}
```

Listing 491: test/test_output/semantic/test_class_specify_constructor_some.out

```
Semantic check succeded!
```

Listing 492: test/test_output/full_pipeline/test_class_specify_constructor_some.out

```
149
2
5
```

Listing 493: test/test_programs/test_classes_demo.meow

```
HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies.

    HAI ITZ ME FUNC Set_Num_Cookies WIT NUMBR cookies_given,
        cookies IZ cookies_given.
    KBYE

    HAI ITZ ME NUMBR FUNC Get_Num_Cookies,
        GIVE cookies.
    KBYE

    HAI ITZ ME FUNC Incr_Cookies,
        ITZ ME NUMBR existing_cookies IZ PURR Get_Num_Cookies IN HERE.
        cookies IZ SUM OF existing_cookies AN 1.
    KBYE

KBYE

HAI ITZ ME FUNC Main,

    ITZ ME NUMBR jerrys_cookies.
    ITZ ME YARN message.
    ITZ ME YARN str_cookies.
    MAEK Jerry NEW MOUSE.

    PSST Set the cookie count
    PURR Meow WIT "Setting Jerry's cookies count to 10!".
    PURR Set_Num_Cookies IN Jerry WIT 10.

    PSST Call class method to increment cookies by 1
    PURR Meow WIT "Incrementing Jerry's cookie count...".
    PURR Incr_Cookies IN Jerry.

    PSST Use class method to get new number of cookies
    jerrys_cookies IZ PURR Get_Num_Cookies IN Jerry.
    str_cookies IZ YARN jerrys_cookies.

    message IZ CAT "Jerry now has " AN CAT str_cookies AN " cookies!".
    PURR Meow WIT message.
```

```
42
43      BLEEP Jerry.
44      BLEEP str_cookies.
45  KBYE
```

Listing 494: test/test_output/ast/test_classes_demo.out

```
1
2
3   Main()
4   {
5       int jerrys_cookies;
6       char * message;
7       char * str_cookies;
8       class MOUSE Jerry;
9       printf("%X\n", "Setting Jerry's cookies count to 10!");
10      Jerry.Set_Num_Cookies(10);
11      printf("%X\n", "Incrementing Jerry's cookie count...");
12      Jerry.Incr_Cookies();
13      jerrys_cookies = Jerry.Get_Num_Cookies();
14      str_cookies = (char *) jerrys_cookies;
15      message = "Jerry now has " + str_cookies + " cookies!";
16      printf("%X\n", message);
17      free(Jerry);
18      free(str_cookies);
19  }
20
21  Class MOUSE {
22
23      int cookies;
24
25      Incr_Cookies()
26      {
27      int existing_cookies = this.Get_Num_Cookies();
28      cookies = existing_cookies + 1;
29      }
30      int Get_Num_Cookies()
31      {
32      return cookies;
33      }
34      Set_Num_Cookies(int cookies_given)
35      {
36      cookies = cookies_given;
37      }
38
39  }
```

Listing 495: test/test_output/semantic/test_classes_demo.out

```
Semantic check succeded!
```

Listing 496: test/test_output/full_pipeline/test_classes_demo.out

```
Setting Jerry's cookies count to 10!
Incrementing Jerry's cookie count...
Jerry now has 11 cookies!
```

Listing 497: test/test_programs/test_comparison1.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME BOO yes IZ AYE.
    ITZ ME BOO yah_huh IZ AYE.
    ITZ ME BOO no IZ NAY.

    PURR Meow WIT DIFFRINT yes AN no.       PSST 1
    PURR Meow WIT DIFFRINT yes AN yah_huh. PSST 0
    PURR Meow WIT SAEM yes AN yah_huh.      PSST 1
    PURR Meow WIT SAEM yes AN no.           PSST 0

KBYE
```

Listing 498: test/test_output/ast/test_comparison1.out

```


Main()
{
    bool yes = true;
    bool yah_huh = true;
    bool no = false;
    printf("%X\n", yes != no);
    printf("%X\n", yes != yah_huh);
    printf("%X\n", yes == yah_huh);
    printf("%X\n", yes == no);
}
```

Listing 499: test/test_output/semantic/test_comparison1.out

```
Semantic check succeded!
```

Listing 500: test/test_output/full_pipeline/test_comparison1.out

```
1
0
1
0
```

Listing 501: test/test_programs/test_comparison2.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR ten IZ 10.
    ITZ ME NUMBR eleven IZ 11.
    ITZ ME NUMBR ten_again IZ 10.

    PURR Meow WIT DIFFRINT ten AN eleven.       PSST 1
    PURR Meow WIT DIFFRINT ten AN ten_again.    PSST 0
    PURR Meow WIT SAEM ten AN ten_again.        PSST 1
    PURR Meow WIT SAEM ten AN eleven.           PSST 0

KBYE
```

Listing 502: test/test_output/ast/test_comparison2.out

```


Main()
{
    int ten = 10;
    int eleven = 11;
    int ten_again = 10;
    printf("%X\n", ten != eleven);
    printf("%X\n", ten != ten_again);
    printf("%X\n", ten == ten_again);
    printf("%X\n", ten == eleven);
}
```

Listing 503: test/test_output/semantic/test_comparison2.out

```
Semantic check succeded!
```

Listing 504: test/test_output/full_pipeline/test_comparison2.out

```
1
0
1
0
```

Listing 505: test/test_programs/test_comparison3.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBAR val IZ 10.2.
    ITZ ME NUMBAR different IZ 20.1.
    ITZ ME NUMBAR val_again IZ 10.2.

```

```
    PURR Meow WIT DIFFRINT val AN different.     PSST 1
    PURR Meow WIT DIFFRINT val AN val_again.     PSST 0
    PURR Meow WIT SAEM val AN val_again.         PSST 1
    PURR Meow WIT SAEM val AN different.         PSST 0

KBYE
```

Listing 506: test/test_output/ast/test_comparison3.out

```


Main()
{
    float val = 10.2;
    float different = 20.1;
    float val_again = 10.2;
    printf("%X\n", val != different);
    printf("%X\n", val != val_again);
    printf("%X\n", val == val_again);
    printf("%X\n", val == different);
}
```

Listing 507: test/test_output/semantic/test_comparison3.out

```
Semantic check succeded!
```

Listing 508: test/test_output/full_pipeline/test_comparison3.out

```
1
0
1
0
```

Listing 509: test/test_programs/test_comparison4.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR ten IZ 10.
    ITZ ME NUMBR eleven IZ 11.
    ITZ ME NUMBAR ten_again IZ 10.0.

    PURR Meow WIT DIFFRINT ten AN eleven.        PSST 1
    PURR Meow WIT DIFFRINT ten AN ten_again.     PSST 0
    PURR Meow WIT SAEM ten AN ten_again.         PSST 1
    PURR Meow WIT SAEM ten AN eleven.            PSST 0

KBYE
```

Listing 510: test/test_output/ast/test_comparison4.out

```
1
2
3  Main()
4  {
5      int ten = 10;
6      int eleven = 11;
7      float ten_again = 10.0;
8      printf("%X\n", ten != eleven);
9      printf("%X\n", ten != ten_again);
10     printf("%X\n", ten == ten_again);
11     printf("%X\n", ten == eleven);
12 }
```

Listing 511: test/test_output/semantic/test_comparison4.out

```
1  Semantic check succeded!
```

Listing 512: test/test_output/full_pipeline/test_comparison4.out

```
1  1
2  0
3  1
4  0
```

Listing 513: test/test_programs/test_comparison5.meow

```
1  HAI ITZ ME FUNC Main,
2
3      ITZ ME NUMBR ten IZ 10.
4      ITZ ME NUMBR fifteen IZ 15.
5
6      PURR Meow WIT SMALLR ten THAN fifteen.     PSST 1
7      PURR Meow WIT SMALLR fifteen THAN ten.     PSST 0
8      PURR Meow WIT BIGGR fifteen THAN ten.      PSST 1
9      PURR Meow WIT BIGGR ten THAN fifteen.      PSST 0
10
11 KBYE
```

Listing 514: test/test_output/ast/test_comparison5.out

```
1
2
3  Main()
4  {
5      int ten = 10;
6      int fifteen = 15;
7      printf("%X\n", ten < fifteen);
```

```
    printf("%X\n", fifteen < ten);
    printf("%X\n", fifteen > ten);
    printf("%X\n", ten > fifteen);
}
```

Listing 515: test/test_output/semantic/test_comparison5.out

```
Semantic check succeded!
```

Listing 516: test/test_output/full_pipeline/test_comparison5.out

```
1
0
1
0
```

Listing 517: test/test_programs/test_comparison6.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBAR ten IZ 10.2.
    ITZ ME NUMBAR fifteen IZ 15.4.

    PURR Meow WIT SMALLR ten THAN fifteen.    PSST 1
    PURR Meow WIT SMALLR fifteen THAN ten.    PSST 0
    PURR Meow WIT BIGGR fifteen THAN ten.     PSST 1
    PURR Meow WIT BIGGR ten THAN fifteen.     PSST 0

KBYE
```

Listing 518: test/test_output/ast/test_comparison6.out

```


Main()
{
    float ten = 10.2;
    float fifteen = 15.4;
    printf("%X\n", ten < fifteen);
    printf("%X\n", fifteen < ten);
    printf("%X\n", fifteen > ten);
    printf("%X\n", ten > fifteen);
}
```

Listing 519: test/test_output/semantic/test_comparison6.out

```
Semantic check succeded!
```

Listing 520: test/test_output/full_pipeline/test_comparison6.out

```
1
0
1
0
```

Listing 521: test/test_programs/test_comparison7.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR ten IZ 10.
    ITZ ME NUMBAR fifteen IZ 15.4.

    PURR Meow WIT SMALLR ten THAN fifteen.    PSST 1
    PURR Meow WIT SMALLR fifteen THAN ten.    PSST 0
    PURR Meow WIT BIGGR fifteen THAN ten.     PSST 1
    PURR Meow WIT BIGGR ten THAN fifteen.     PSST 0

KBYE
```

Listing 522: test/test_output/ast/test_comparison7.out

```


Main()
{
    int ten = 10;
    float fifteen = 15.4;
    printf("%X\n", ten < fifteen);
    printf("%X\n", fifteen < ten);
    printf("%X\n", fifteen > ten);
    printf("%X\n", ten > fifteen);
}
```

Listing 523: test/test_output/semantic/test_comparison7.out

```
Semantic check succeded!
```

Listing 524: test/test_output/full_pipeline/test_comparison7.out

```
1
0
1
0
```

Listing 525: test/test_programs/test_comparison8.meow

```
HAI ITZ ME FUNC Main,
```

```
2
3       ITZ ME NUMBR ten IZ 10.
4       ITZ ME NUMBAR fifteen IZ 15.4.
5
6       PURR Meow WIT BOTH OF
7           SMALLR ten THAN fifteen AN BIGGR fifteen THAN ten.    PSST 1
8
9       PURR Meow WIT BOTH OF
10          SMALLR fifteen THAN ten AN BIGGR fifteen THAN ten.    PSST 0
11
12      PURR Meow WIT EITHER OF
13          SMALLR fifteen THAN ten AN BIGGR fifteen THAN ten.    PSST 1
14
15      PURR Meow WIT EITHER OF
16          SMALLR fifteen THAN ten AN BIGGR ten THAN fifteen.    PSST 0
17
18  KBYE
```

Listing 526: test/test_output/ast/test_comparison8.out

```
1
2
3   Main()
4   {
5       int ten = 10;
6       float fifteen = 15.4;
7       printf("%X\n", ten < fifteen && fifteen > ten);
8       printf("%X\n", fifteen < ten && fifteen > ten);
9       printf("%X\n", fifteen < ten || fifteen > ten);
10      printf("%X\n", fifteen < ten || ten > fifteen);
11  }
```

Listing 527: test/test_output/semantic/test_comparison8.out

```
1   Semantic check succeded!
```

Listing 528: test/test_output/full_pipeline/test_comparison8.out

```
1   1
2   0
3   1
4   0
```

Listing 529: test/test_programs/test_comparison9.meow

```
1   HAI ITZ ME FUNC Main,
2
3       ITZ ME YARN val IZ "this is a string".
```

```
    ITZ ME YARN dup_val IZ "this is a string".
    ITZ ME YARN not_dup IZ "is something different".

    PURR Meow WIT SAEM val AN dup_val.   PSST 1
    PURR Meow WIT SAEM val AN not_dup.   PSST 0

KBYE
```

Listing 530: test/test_output/ast/test_comparison9.out

```


Main()
{
    char * val = "this is a string";
    char * dup_val = "this is a string";
    char * not_dup = "is something different";
    printf("%X\n", val == dup_val);
    printf("%X\n", val == not_dup);
}
```

Listing 531: test/test_output/semantic/test_comparison9.out

```
Semantic check succeded!
```

Listing 532: test/test_output/full_pipeline/test_comparison9.out

```
1
0
```

Listing 533: test/test_programs/test_concat.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME YARN val IZ "string1. ".
    ITZ ME YARN dup_val IZ "string2.".
    ITZ ME YARN concatenated_str.

    concatenated_str IZ CAT val AN dup_val.
    PURR Meow WIT concatenated_str.

    BLEEP concatenated_str.

KBYE
```

Listing 534: test/test_output/ast/test_concat.out

```

```

```

Main()
{
    char * val = "string1. ";
    char * dup_val = "string2.";
    char * concatenated_str;
    concatenated_str = val + dup_val;
    printf("%X\n", concatenated_str);
    free(concatenated_str);
}
```

Listing 535: test/test_output/semantic/test_concat.out

```
Semantic check succeded!
```

Listing 536: test/test_output/full_pipeline/test_concat.out

```
string1. string2.
```

Listing 537: test/test_programs/test_concat_float_str.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME YARN str IZ "this is a string. the following is a float: ".
    ITZ ME NUMBAR flt IZ 2.0.
    ITZ ME YARN concatenated_str.

    concatenated_str IZ CAT str AN flt.
    PURR Meow WIT concatenated_str.

    BLEEP concatenated_str.

KBYE
```

Listing 538: test/test_output/ast/test_concat_float_str.out

```


Main()
{
    char * str = "this is a string. the following is a float: ";
    float flt = 2.0;
    char * concatenated_str;
    concatenated_str = str + flt;
    printf("%X\n", concatenated_str);
    free(concatenated_str);
}
```

Listing 539: test/test_output/semantic/test_concat_float_str.out

```
Semantic check succeded!
```

Listing 540: test/test_output/full_pipeline/test_concat_float_str.out

```
this is a string. the following is a float: 2.000000
```

Listing 541: test/test_programs/test_concat_int_str.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR int IZ 2.
    ITZ ME YARN str IZ " <- that was an int. this is a string.".
    ITZ ME YARN concatenated_str.

    concatenated_str IZ CAT int AN str.
    PURR Meow WIT concatenated_str.

    BLEEP concatenated_str.

KBYE
```

Listing 542: test/test_output/ast/test_concat_int_str.out

```


Main()
{
    int int = 2;
    char * str = " <- that was an int. this is a string.";
    char * concatenated_str;
    concatenated_str = int + str;
    printf("%X\n", concatenated_str);
    free(concatenated_str);
}
```

Listing 543: test/test_output/semantic/test_concat_int_str.out

```
Semantic check succeded!
```

Listing 544: test/test_output/full_pipeline/test_concat_int_str.out

```
2 <- that was an int. this is a string.
```

Listing 545: test/test_programs/test_concat_str_float.meow

```
HAI ITZ ME FUNC Main,

```

```
    ITZ ME NUMBAR flt IZ 2.0.
    ITZ ME YARN str IZ " <- that was a float. this is a string.".
    ITZ ME YARN concatenated_str.

    concatenated_str IZ CAT flt AN str.
    PURR Meow WIT concatenated_str.

    BLEEP concatenated_str.

KBYE
```

Listing 546: test/test_output/ast/test_concat_str_float.out

```


Main()
{
    float flt = 2.0;
    char * str = " <- that was a float. this is a string.";
    char * concatenated_str;
    concatenated_str = flt + str;
    printf("%X\n", concatenated_str);
    free(concatenated_str);
}
```

Listing 547: test/test_output/semantic/test_concat_str_float.out

```
Semantic check succeded!
```

Listing 548: test/test_output/full_pipeline/test_concat_str_float.out

```
2.000000 <- that was a float. this is a string.
```

Listing 549: test/test_programs/test_concat_str_int.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME YARN str IZ "this is a string. the following is an int: ".
    ITZ ME NUMBR int IZ 2.
    ITZ ME YARN concatenated_str.

    concatenated_str IZ CAT str AN int.
    PURR Meow WIT concatenated_str.

    BLEEP concatenated_str.

KBYE
```

Listing 550: test/test_output/ast/test_concat_str_int.out

```

Main()
{
    char * str = "this is a string. the following is an int: ";
    int int = 2;
    char * concatenated_str;
    concatenated_str = str + int;
    printf("%X\n", concatenated_str);
    free(concatenated_str);
}
```

Listing 551: test/test_output/semantic/test_concat_str_int.out

```
Semantic check succeded!
```

Listing 552: test/test_output/full_pipeline/test_concat_str_int.out

```
this is a string. the following is an int: 2
```

Listing 553: test/test_programs/test_create_array1.meow

```

HAI ITZ ME NUMBR FUNC Main,

    PSST Create simple array
    MAEK simple_array NEW BUCKET OF YARN HOLDS 3.
    BLEEP simple_array.
    GIVE 0.

KBYE
```

Listing 554: test/test_output/ast/test_create_array1.out

```


int Main()
{
    char * [3] simple_array = [  ];
    free(simple_array);
    return 0;
}
```

Listing 555: test/test_output/semantic/test_create_array1.out

```
Semantic check succeded!
```

Listing 556: test/test_output/full_pipeline/test_create_array1.out

```
```

Listing 557: test/test_programs/test_create_array2.meow

```
1
2  HAI ITZ ME NUMBR FUNC Main,
3
4      PSST Create simple array with all elements initialized
5      MAEK animals NEW BUCKET OF YARN HOLDS 3,
6          WIT "Cats"
7          AN "Dogs"
8          AN "More Dogs".
9
10     PURR Meow WIT animals[0].
11     PURR Meow WIT animals[1].
12     PURR Meow WIT animals[2].
13
14     BLEEP animals.
15     GIVE 0.
16 KBYE
```

Listing 558: test/test_output/ast/test_create_array2.out

```
1
2
3  int Main()
4  {
5      char * [3] animals = [ "Cats", "Dogs", "More Dogs" ];
6      printf("%X\n", animals[0]);
7      printf("%X\n", animals[1]);
8      printf("%X\n", animals[2]);
9      free(animals);
10     return 0;
11 }
```

Listing 559: test/test_output/semantic/test_create_array2.out

```
1  Semantic check succeded!
```

Listing 560: test/test_output/full_pipeline/test_create_array2.out

```
1  Cats
2  Dogs
3  More Dogs
```

Listing 561: test/test_programs/test_create_array3.meow

```
1
```

```
HAI ITZ ME FUNC Main,

    PSST Create simple array with all integers initialized
    MAEK numbers NEW BUCKET OF NUMBR HOLDS 3,
        WIT 1
        AN 2
        AN 3.

    PURR Meow WIT numbers[2].
    BLEEP numbers.

KBYE
```

Listing 562: test/test_output/ast/test_create_array3.out

```


Main()
{
    int [3] numbers = [ 1, 2, 3 ];
    printf("%X\n", numbers[2]);
    free(numbers);
}
```

Listing 563: test/test_output/semantic/test_create_array3.out

```
Semantic check succeded!
```

Listing 564: test/test_output/full_pipeline/test_create_array3.out

```
3
```

Listing 565: test/test_programs/test_create_array4.meow

```

HAI ITZ ME FUNC Main,

    PSST Create array using an expression as a size
    ITZ ME NUMBR size IZ SUM OF 3 AN 5.

    MAEK animals NEW BUCKET OF YARN HOLDS size,
        WIT "Cats"
        AN "Dogs".

    PURR Meow WIT animals[0].
    PURR Meow WIT animals[1].
    BLEEP animals.

KBYE
```

Listing 566: test/test_output/ast/test_create_array4.out

```

Main()
{
    int size = 3 + 5;
    char * [size] animals = [ "Cats", "Dogs" ];
    printf("%X\n", animals[0]);
    printf("%X\n", animals[1]);
    free(animals);
}
```

Listing 567: test/test_output/semantic/test_create_array4.out

```
Semantic check succeded!
```

Listing 568: test/test_output/full_pipeline/test_create_array4.out

```
Cats
Dogs
```

Listing 569: test/test_programs/test_create_array5.meow

```

HAI ITZ ME FUNC Main,

    PSST Create simple array with some elements initialized
    MAEK animals NEW BUCKET OF YARN HOLDS 3,
        WIT "Cats"
        AN "Dogs".

    BLEEP animals.

KBYE
```

Listing 570: test/test_output/ast/test_create_array5.out

```

Main()
{
    char * [3] animals = [ "Cats", "Dogs" ];
    free(animals);
}
```

Listing 571: test/test_output/semantic/test_create_array5.out

```
Semantic check succeded!
```

Listing 572: test/test_output/full_pipeline/test_create_array5.out

```
```

Listing 573: test/test_programs/test_create_array6.meow

```
1
2  HAI ITZ ME FUNC Main,
3
4      ITZ ME YARN str1 IZ "whoooopie!".
5      ITZ ME YARN str2 IZ "yiiippeee!".
6
7      PSST Create simple array with elements that are variables
8      MAEK exclamations NEW BUCKET OF YARN HOLDS 3,
9          WIT str1
10         AN str2.
11
12     PURR Meow WIT exclamations[0].
13     BLEEP exclamations.
14
15 KBYE
```

Listing 574: test/test_output/ast/test_create_array6.out

```
1
2
3  Main()
4  {
5      char * str1 = "whoooopie!";
6      char * str2 = "yiiippeee!";
7      char * [3] exclamations = [ str1, str2 ];
8      printf("%X\n", exclamations[0]);
9      free(exclamations);
10 }
```

Listing 575: test/test_output/semantic/test_create_array6.out

```
1  Semantic check succeded!
```

Listing 576: test/test_output/full_pipeline/test_create_array6.out

```
1  whoooopie!
```

Listing 577: test/test_programs/test_create_array7.meow

```
1
2  HAI ITZ ME FUNC Main,
3
4          PSST Bucket size specified, not contents
5          MAEK empty_array NEW BUCKET OF BOO HOLDS 10.
```

```
        BLEEP empty_array.

KBYE
```

Listing 578: test/test_output/ast/test_create_array7.out

```

Main()
{
    bool [10] empty_array = [  ];
    free(empty_array);
}
```

Listing 579: test/test_output/semantic/test_create_array7.out

```
Semantic check succeded!
```

Listing 580: test/test_output/full_pipeline/test_create_array7.out

```
```

Listing 581: test/test_programs/test_create_array8.meow

```

HAI ITZ ME CLASS PET,

    ITZ ME YARN name.
    ITZ ME YARN type.
    ITZ ME NUMBR age.

    HAI ITZ ME YARN FUNC Get_Name,
        GIVE name.
    KBYE

KBYE


HAI ITZ ME NUMBR FUNC Main,

    ITZ ME NUMBR size IZ 2.
    ITZ ME NUMBR idx IZ 1.
    ITZ ME PET Cat.
    ITZ ME YARN cat_name.

    PSST Make some pets
    MAEK Silvester NEW PET,
        WIT name IZ "Silvester"
        AN type IZ "cat"
```

```
        AN age IZ 4.

    MAEK Tank NEW PET,
        WIT name IZ "Tank"
        AN type IZ "dog"
        AN age IZ 2.

    PSST Make an array of objects
    MAEK my_pets NEW BUCKET OF PET HOLDS size,
        WIT Silvester
        AN Tank.

    BLEEP my_pets.
    BLEEP Silvester.
    BLEEP Tank.
    GIVE 0.

KBYE
```

Listing 582: test/test_output/ast/test_create_array8.out

```


int Main()
{
    int size = 2;
    int idx = 1;
    class PET Cat;
    char * cat_name;
    class PET Silvester(age = 4, type = "cat", name = "Silvester", );
    class PET Tank(age = 2, type = "dog", name = "Tank", );
    class PET [size] my_pets = [ Silvester, Tank ];
    free(my_pets);
    free(Silvester);
    free(Tank);
    return 0;
}

Class PET {

    int age;
    char * type;
    char * name;

    char * Get_Name()
    {
    return name;
    }
```

```

}
```

Listing 583: test/test_output/semantic/test_create_array8.out

```
Semantic check succeded!
```

Listing 584: test/test_output/full_pipeline/test_create_array8.out

```
```

Listing 585: test/test_programs/test_create_instance.meow

```
HAI ITZ ME FUNC Main,
    PSST If assigning instance variables they must be of the expected type
    MAEK jerry NEW MOUSE,
        WIT cookies IZ 2.
    BLEEP jerry.
KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 0.

    HAI ITZ ME NUMBR FUNC Count_Cookies,
        GIVE cookies.
    KBYE

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This uses the SUM prefix operator
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE
```

Listing 586: test/test_output/ast/test_create_instance.out

```


Main()
{
    class MOUSE jerry(cookies = 2, );
    free(jerry);
}

Class MOUSE {

    int cookies = 0;

```

```
    Give_Cookie(int count_cookies)
    {
    cookies = cookies + count_cookies;
    }
    int Count_Cookies()
    {
    return cookies;
    }

}
```

Listing 587: test/test_output/semantic/test_create_instance.out

```
Semantic check succeded!
```

Listing 588: test/test_output/full_pipeline/test_create_instance.out

```
```

Listing 589: test/test_programs/test_for.meow

```
HAI ITZ ME FUNC Main,

        ITZ ME NUMBR count.
        ITZ ME YARN count_str.
        ITZ ME NUMBR index.
        ITZ ME YARN condition.
        count IZ 0.

        PSST Incrementing with initialized index
        IM IN YR LOOP count UPPIN AN SMALLR count THAN 10 HAI
            condition IZ "count is still not 10".
        KBYE

        PSST Decrementing with initialized index
        count IZ 20.
        IM IN YR LOOP count NERFIN AN BIGGR count THAN 10 HAI
            condition IZ "count is still more than 10. ".
            count_str IZ YARN count.
            condition IZ CAT condition AN "count is: ".
            condition IZ CAT condition AN count_str.
            PURR Meow WIT condition.
        KBYE

        BLEEP condition.


        PSST Initialize index and then increment
        IM IN YR LOOP index NERFIN index IZ 15 AN BIGGR index THAN 10 HAI
```

```
            condition IZ "index is still more than 10".
        KBYE

KBYE
```

Listing 590: test/test_output/ast/test_for.out

```


Main()
{
    int count;
    char * count_str;
    int index;
    char * condition;
    count = 0;
    for ( count++ count < 10) {
            {
    condition = "count is still not 10";
    }
    }
    count = 20;
    for ( count-- count > 10) {
            {
    condition = "count is still more than 10. ";
    count_str = (char *) count;
    condition = condition + "count is: ";
    condition = condition + count_str;
    printf("%X\n", condition);
    }
    }
    free(condition);
    for (index = 15 index-- index > 10) {
            {
    condition = "index is still more than 10";
    }
    }
}
```

Listing 591: test/test_output/semantic/test_for.out

```
Semantic check succeded!
```

Listing 592: test/test_output/full_pipeline/test_for.out

```
count is still more than 10. count is: 20
count is still more than 10. count is: 19
count is still more than 10. count is: 18
```

```
count is still more than 10. count is: 17
count is still more than 10. count is: 16
count is still more than 10. count is: 15
count is still more than 10. count is: 14
count is still more than 10. count is: 13
count is still more than 10. count is: 12
count is still more than 10. count is: 11
```

Listing 593: test/test_programs/test_hello_world.meow

```
HAI ITZ ME NUMBR FUNC Main,

        PURR Meow WIT "hello, world".
        GIVE 0.

KBYE
```

Listing 594: test/test_output/ast/test_hello_world.out

```


int Main()
{
    printf("%X\n", "hello, world");
    return 0;
}
```

Listing 595: test/test_output/semantic/test_hello_world.out

```
Semantic check succeded!
```

Listing 596: test/test_output/full_pipeline/test_hello_world.out

```
hello, world
```

Listing 597: test/test_programs/test_hello_world_complex.meow

```
HAI ITZ ME CLASS PERSON,

    ITZ ME YARN name.
    ITZ ME NUMBR height_inches IZ 70.

    HAI ITZ ME YARN FUNC Get_Name,
        PURR Meow WIT "What is your name?".
        PURR Scan WIT name.
        GIVE name.
```

```
    KBYE

KBYE

HAI ITZ ME NUMBR FUNC Main,
        PSST Test printing an integer
        PURR Meow WIT "hello world\n".
        GIVE 0.
KBYE
```

Listing 598: test/test_output/ast/test_hello_world_complex.out

```


int Main()
{
    printf("%X\n", "hello world\n");
    return 0;
}

Class PERSON {

    int height_inches = 70;
    char * name = "Alexandra";

    char * Get_Name()
    {
    return name;
    }

}
```

Listing 599: test/test_output/semantic/test_hello_world_complex.out

```
Semantic check succeded!
```

Listing 600: test/test_output/full_pipeline/test_hello_world_complex.out

```
hello world\n
```

Listing 601: test/test_programs/test_if1.meow

```
HAI ITZ ME FUNC Main,

        PSST Test if-then-else

        ITZ ME YARN condition.

```

```
        PSST IF THEN
        SMALLR 4 THAN 10
        O RLY?
        YA RLY HAI
            condition IZ "math is the only Truth".
        KBYE

        PSST IF THEN ELSE
        SAEM 4 AN 4
        O RLY?
        YA RLY HAI
            condition IZ "same same".
        KBYE
        NO WAI HAI
            condition IZ "not same same".
        KBYE

        PSST IF THEN ELSE with code blocks
        SAEM 10 AN 10
        O RLY?
        YA RLY HAI
            condition IZ "ten out of ten.".
        KBYE
        NO WAI HAI
            condition IZ "my ten is better than your ten.".
        KBYE

KBYE
```

Listing 602: test/test_output/ast/test_if1.out

```


Main()
{
    char * condition;
    if (4 < 10)
    {
    condition = "math is the only Truth";
    }
    if (4 == 4) {
    condition = "same same";
    }
    else        {
    condition = "not same same";
    }
    if (10 == 10)   {
    condition = "ten out of ten.";
```

```
    }
    else        {
    condition = "my ten is better than your ten.";
    }
}
```

Listing 603: test/test_output/semantic/test_if1.out

```
Semantic check succeded!
```

Listing 604: test/test_output/full_pipeline/test_if1.out

```
```

Listing 605: test/test_programs/test_if2.meow

```
HAI ITZ ME FUNC Main,
    PSST if statement inside if statement

    ITZ ME YARN condition.

    SAEM 3 AN 3
    O RLY?
    YA RLY HAI
        SMALLR 1 THAN 4
        O RLY?
        YA RLY HAI
            condition IZ "Pancakes".
        KBYE
    KBYE
    NO WAI HAI
        condition IZ "Eggs".
    KBYE

KBYE
```

Listing 606: test/test_output/ast/test_if2.out

```


Main()
{
    char * condition;
    if (3 == 3) {
    if (1 < 4)
    {
    condition = "Pancakes";
    }
    }
```

```
    else        {
    condition = "Eggs";
    }
}
```

Listing 607: `test/test_output/semantic/test_if2.out`

```
Semantic check succeded!
```

Listing 608: `test/test_output/full_pipeline/test_if2.out`

```
```

Listing 609: `test/test_programs/test_imports.meow`

```
GIMME IMPORT_EXAMPLE?

HAI ITZ ME NUMBR FUNC Main,
        PURR Say_Hello.
        GIVE 0.
KBYE
```

Listing 610: `test/test_output/ast/test_imports.out`

```


char * Get_User_Name()
{
    char * username;
    printf("%X\n", "What is your name? ");
    Scan(username);
    return username;
}

Say_Hello()
{
    char * username;
    char * message;
    username = Get_User_Name();
    message = "hello, " + username;
    printf("%X\n", message);
    free(message);
    free(username);
}

int Main()
{
    Say_Hello();
    return 0;
}
```

Listing 611: test/test_output/semantic/test_imports.out

```
Semantic check succeded!
```

Listing 612: test/test_output/full_pipeline/test_imports.out

```
What is your name?
hello, tester
```

Listing 613: test/test_programs/test_math_auto_cast.meow

```

HAI ITZ ME NUMBR FUNC Main,

        PSST Creating some local variables
        ITZ ME NUMBR count IZ 2.
        ITZ ME NUMBAR count2 IZ 2.3.

        PSST Test printing an integer
        PURR Meow WIT PRODUKT OF count AN count2.  PSST 4.6

        GIVE 0.
KBYE
```

Listing 614: test/test_output/ast/test_math_auto_cast.out

```


int Main()
{
    int count = 2;
    float count2 = 2.3;
    printf("%X\n", count * count2);
    return 0;
}
```

Listing 615: test/test_output/semantic/test_math_auto_cast.out

```
Semantic check succeded!
```

Listing 616: test/test_output/full_pipeline/test_math_auto_cast.out

```
4.6
```

Listing 617: test/test_programs/test_meow_bool.meow

```

HAI ITZ ME NUMBR FUNC Main,

```

```
        PSST Creating a boolean
        ITZ ME BOO is_true IZ AYE.
        ITZ ME BOO is_false IZ NAY.

        PSST Test printing booleans
        PURR Meow WIT is_true.
        PURR Meow WIT is_false.

        GIVE 0.
KBYE
```

Listing 618: test/test_output/ast/test_meow_bool.out

```


int Main()
{
    bool is_true = true;
    bool is_false = false;
    printf("%X\n", is_true);
    printf("%X\n", is_false);
    return 0;
}
```

Listing 619: test/test_output/semantic/test_meow_bool.out

```
Semantic check succeded!
```

Listing 620: test/test_output/full_pipeline/test_meow_bool.out

```
1
0
```

Listing 621: test/test_programs/test_meow_float.meow

```

HAI ITZ ME NUMBR FUNC Main,

        PSST Creating a float
        ITZ ME NUMBAR pi IZ 3.14159.

        PSST Test printing float
        PURR Meow WIT pi.

        GIVE 0.
KBYE
```

Listing 622: test/test_output/ast/test_meow_float.out

```

int Main()
{
    float pi = 3.14159;
    printf("%X\n", pi);
    return 0;
}
```

Listing 623: test/test_output/semantic/test_meow_float.out

```
Semantic check succeded!
```

Listing 624: test/test_output/full_pipeline/test_meow_float.out

```
3.14159
```

Listing 625: test/test_programs/test_meow_int.meow

```

HAI ITZ ME NUMBR FUNC Main,

        PSST Creating some local variables
        ITZ ME NUMBR count.

        PSST Assigning local varibles values
        count IZ 2.

        PSST Test printing an integer
        PURR Meow WIT count.

        GIVE 0.
KBYE
```

Listing 626: test/test_output/ast/test_meow_int.out

```

int Main()
{
    int count;
    count = 2;
    printf("%X\n", count);
    return 0;
}
```

Listing 627: test/test_output/semantic/test_meow_int.out

```
Semantic check succeded!
```

Listing 628: test/test_output/full_pipeline/test_meow_int.out

```
2
```

Listing 629: test/test_programs/test_meow_string.meow

```

HAI ITZ ME NUMBR FUNC Main,

        PSST Creating some local variables
        ITZ ME YARN msg.

        PSST Assigning local varibles values
        msg IZ "nice to meet you!".

        PSST Test printing an integer
        PURR Meow WIT msg.

        GIVE 0.
KBYE
```

Listing 630: test/test_output/ast/test_meow_string.out

```


int Main()
{
    char * msg;
    msg = "nice to meet you!";
    printf("%X\n", msg);
    return 0;
}
```

Listing 631: test/test_output/semantic/test_meow_string.out

```
Semantic check succeded!
```

Listing 632: test/test_output/full_pipeline/test_meow_string.out

```
nice to meet you!
```

Listing 633: test/test_programs/test_mouse_class.meow

```
HAI ITZ ME FUNC Main,

```

```
    ITZ ME NUMBR jerrys_cookies.
    MAEK Jerry NEW MOUSE.    PSST no specification of cookies

    jerrys_cookies IZ cookies IN Jerry.
    cookies IN Jerry IZ 2.

    PURR Give_Cookie IN Jerry WIT 2.
    jerrys_cookies IZ PURR Count_Cookies IN Jerry.

    BLEEP Jerry.

KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 0.

    HAI ITZ ME NUMBR FUNC Count_Cookies,
        GIVE cookies.
    KBYE

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This uses the SUM prefix operator
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE
```

Listing 634: `test/test_output/ast/test_mouse_class.out`

```


Main()
{
    int jerrys_cookies;
    class MOUSE Jerry;
    jerrys_cookies = Jerry.cookies;
    Jerry.cookies = 2;
    Jerry.Give_Cookie(2);
    jerrys_cookies = Jerry.Count_Cookies();
    free(Jerry);
}

Class MOUSE {

    int cookies = 0;

    Give_Cookie(int count_cookies)
```

```
    {
    cookies = cookies + count_cookies;
    }
    int Count_Cookies()
    {
    return cookies;
    }

}
```

Listing 635: test/test_output/semantic/test_mouse_class.out

```
Semantic check succeded!
```

Listing 636: test/test_output/full_pipeline/test_mouse_class.out

```
```

Listing 637: test/test_programs/test_mouse_class_two.meow

```
PSST Just providing that functions and classes can be defined in any order
HAI ITZ ME NUMBR FUNC Rando,
    GIVE 0.
KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies IZ 0.

    HAI ITZ ME NUMBR FUNC Count_Cookies,
        GIVE cookies.
    KBYE

    HAI ITZ ME FUNC Give_Cookie WIT NUMBR count_cookies,
        PSST This uses the SUM prefix operator
        cookies IZ SUM OF cookies AN count_cookies.
    KBYE

KBYE

HAI ITZ ME NUMBR FUNC Main,

    ITZ ME NUMBR jerrys_cookies.
    MAEK Jerry NEW MOUSE.   PSST no specification of cookies

    jerrys_cookies IZ cookies IN Jerry.
    cookies IN Jerry IZ 2.

```

```
    PURR Give_Cookie IN Jerry WIT 2.
    jerrys_cookies IZ PURR Count_Cookies IN Jerry.

    BLEEP Jerry.
    GIVE jerrys_cookies.

KBYE
```

Listing 638: test/test_output/ast/test_mouse_class_two.out

```


int Main()
{
    int jerrys_cookies;
    class MOUSE Jerry;
    jerrys_cookies = Jerry.cookies;
    Jerry.cookies = 2;
    Jerry.Give_Cookie(2);
    jerrys_cookies = Jerry.Count_Cookies();
    free(Jerry);
    return jerrys_cookies;
}

int Rando()
{
    return 0;
}

Class MOUSE {

    int cookies = 0;

    Give_Cookie(int count_cookies)
    {
    cookies = cookies + count_cookies;
    }
    int Count_Cookies()
    {
    return cookies;
    }

}
```

Listing 639: test/test_output/semantic/test_mouse_class_two.out

```
Semantic check succeded!
```

Listing 640: test/test_output/full_pipeline/test_mouse_class_two.out

```

```

Listing 641: test/test_programs/test_pass_array_to_function.meow

```
1
2  HAI ITZ ME FUNC Print_First_In_Array WIT BUCKET OF NUMBR numbers,
3
4      PURR Meow WIT "This is the first number in the array: ".
5      PURR Meow WIT numbers[0].
6      PURR Meow WIT "This is the second number in the array: ".
7      PURR Meow WIT numbers[1].
8
9  KBYE
10
11 HAI ITZ ME FUNC Main,
12
13     MAEK numbers NEW BUCKET OF NUMBR HOLDS 3,
14         WIT 1
15         AN 2
16         AN 3.
17
18     PURR Print_First_In_Array WIT numbers.
19     BLEEP numbers.
20
21 KBYE
```

Listing 642: test/test_output/ast/test_pass_array_to_function.out

```
1
2
3  Main()
4  {
5      int [3] numbers = [ 1, 2, 3 ];
6      Print_First_In_Array(numbers);
7      free(numbers);
8  }
9
10 Print_First_In_Array(int [] numbers)
11 {
12     printf("%X\n", "This is the first number in the array: ");
13     printf("%X\n", numbers[0]);
14     printf("%X\n", "This is the second number in the array: ");
15     printf("%X\n", numbers[1]);
16 }
```

Listing 643: test/test_output/semantic/test_pass_array_to_function.out

```
1  Semantic check succeded!
```

Listing 644: test/test_output/full_pipeline/test_pass_array_to_function.out

```
This is the first number in the array:
1
This is the second number in the array:
2
```

Listing 645: test/test_programs/test_pass_array_to_method.meow

```
HAI ITZ ME CLASS ARRAY_PRINTER,

    HAI ITZ ME FUNC Print_Str_Array_Element
        WIT BUCKET OF YARN expressions
        AN NUMBR index,

        PURR Meow WIT "The best type of pets are: ".
        PURR Meow WIT expressions[index].
    KBYE

KBYE

HAI ITZ ME NUMBR FUNC Main,

    ITZ ME NUMBR size IZ 2.
    ITZ ME NUMBR idx IZ 1.

    PSST Make an array of strings
    MAEK animals NEW BUCKET OF YARN HOLDS size,
        WIT "Cats"
        AN "Dogs".

    PSST Make a printer instance an print element at index 'idx'
    MAEK printer NEW ARRAY_PRINTER.
    PURR Print_Str_Array_Element IN printer WIT animals AN idx.

    BLEEP animals.
    GIVE 0.

KBYE
```

Listing 646: test/test_output/ast/test_pass_array_to_method.out

```


int Main()
{
    int size = 2;
    int idx = 1;
```

```
    char * [size] animals = [ "Cats", "Dogs" ];
    class ARRAY_PRINTER printer;
    printer.Print_Str_Array_Element(animals, idx);
    free(animals);
    return 0;
}

Class ARRAY_PRINTER {


    Print_Str_Array_Element(char * [] expressions, int index)
    {
    printf("%X\n", "The best type of pets are: ");
        printf("%X\n", expressions[index]);
    }

}
```

Listing 647: test/test_output/semantic/test_pass_array_to_method.out

```
Semantic check succeded!
```

Listing 648: test/test_output/full_pipeline/test_pass_array_to_method.out

```
The best type of pets are:
Dogs
```

Listing 649: test/test_programs/test_pass_obj_as_param.meow

```
HAI ITZ ME NUMBR FUNC Main,

    ITZ ME NUMBR jerrys_cookies IZ 0.

    MAEK Jerry NEW MOUSE.

    PSST Call class method to set cookies
    PURR Set_Num_Cookies IN Jerry WIT 10.

    PURR Print_Cookies WIT Jerry.
    BLEEP Jerry.
    GIVE 0.
KBYE

HAI ITZ ME FUNC Print_Cookies WIT MOUSE mouse_guy,

    ITZ ME NUMBR cookies.

    cookies IZ PURR Get_Num_Cookies IN mouse_guy.
```

```
20
21      PURR Meow WIT "The mouse has: ".
22      PURR Meow WIT cookies.
23
24  KBYE
25
26  HAI ITZ ME CLASS MOUSE,
27
28      ITZ ME NUMBR cookies.
29
30      HAI ITZ ME FUNC Set_Num_Cookies WIT NUMBR cookies_given,
31          cookies IZ cookies_given.
32      KBYE
33
34      HAI ITZ ME NUMBR FUNC Get_Num_Cookies,
35          GIVE cookies.
36      KBYE
37
38  KBYE
```

Listing 650: test/test_output/ast/test_pass_obj_as_param.out

```
1
2
3   Print_Cookies(class MOUSE mouse_guy)
4   {
5       int cookies;
6       cookies = mouse_guy.Get_Num_Cookies();
7       printf("%X\n", "The mouse has: ");
8       printf("%X\n", cookies);
9   }
10
11  int Main()
12  {
13      int jerrys_cookies = 0;
14      class MOUSE Jerry;
15      Jerry.Set_Num_Cookies(10);
16      Print_Cookies(Jerry);
17      free(Jerry);
18      return 0;
19  }
20
21  Class MOUSE {
22
23      int cookies;
24
25      int Get_Num_Cookies()
26      {
```

```
    return cookies;
    }
    Set_Num_Cookies(int cookies_given)
    {
    cookies = cookies_given;
    }

}
```

Listing 651: test/test_output/semantic/test_pass_obj_as_param.out

```
Semantic check succeded!
```

Listing 652: test/test_output/full_pipeline/test_pass_obj_as_param.out

```
The mouse has:
10
```

Listing 653: test/test_programs/test_pointless.meow

```

HAI ITZ ME FUNC Main,
        PSST Nothing happening here!
KBYE
```

Listing 654: test/test_output/ast/test_pointless.out

```


Main()
{
}
```

Listing 655: test/test_output/semantic/test_pointless.out

```
Semantic check succeded!
```

Listing 656: test/test_output/full_pipeline/test_pointless.out

```
```

Listing 657: test/test_programs/test_scan.meow

```

HAI ITZ ME FUNC Main,

        PSST Creating a string to hold read content
        ITZ ME YARN user_message.

```

```
        PSST Reading in content
        PURR Meow WIT "Please tell me your favorite color: ".
        PURR Scan WIT user_message.

        PURR Meow WIT "Your favorite color is: ".
        PURR Meow WIT user_message.

        BLEEP user_message.
KBYE
```

Listing 658: test/test_output/ast/test_scan.out

```


Main()
{
    char * user_message;
    printf("%X\n", "Please tell me your favorite color: ");
    Scan(user_message);
    printf("%X\n", "Your favorite color is: ");
    printf("%X\n", user_message);
    free(user_message);
}
```

Listing 659: test/test_output/semantic/test_scan.out

```
Semantic check succeded!
```

Listing 660: test/test_output/full_pipeline/test_scan.out

```
Please tell me your favorite color:
Your favorite color is:
some value
```

Listing 661: test/test_programs/test_scan2.meow

```

HAI ITZ ME FUNC Main,

        ITZ ME YARN favorite_color.

        PSST Reading in content
        PURR Meow WIT "Please tell me your favorite color: ".
        PURR Scan WIT favorite_color.

        PURR Meow WIT CAT "Your favorite color is: " AN favorite_color.

        BLEEP favorite_color.
KBYE
```

Listing 662: test/test_output/ast/test_scan2.out

```
Main()
{
    char * favorite_color;
    printf("%X\n", "Please tell me your favorite color: ");
    Scan(favorite_color);
    printf("%X\n", "Your favorite color is: " + favorite_color);
    free(favorite_color);
}
```

Listing 663: test/test_output/semantic/test_scan2.out

```
Semantic check succeded!
```

Listing 664: test/test_output/full_pipeline/test_scan2.out

```
Please tell me your favorite color:
Your favorite color is: some value
```

Listing 665: test/test_programs/test_simple_class.meow

```
HAI ITZ ME FUNC Main,

    ITZ ME NUMBR jerrys_cookies.

    MAEK Jerry NEW MOUSE.
    cookies IN Jerry IZ 15.

    PSST Make sure that we can call a method and get back cookies
    jerrys_cookies IZ PURR Count_Cookies IN Jerry.
    PURR Meow WIT "Jerry has this many cookies: ".
    PURR Meow WIT jerrys_cookies.

    PSST BLEEP Jerry.
KBYE

HAI ITZ ME CLASS MOUSE,

    ITZ ME NUMBR cookies.
    ITZ ME YARN name.

    HAI ITZ ME NUMBR FUNC Count_Cookies,
        GIVE cookies.
    KBYE
KBYE
```

Listing 666: test/test_output/ast/test_simple_class.out

```

Main()
{
    int jerrys_cookies;
    class MOUSE Jerry;
    Jerry.cookies = 15;
    jerrys_cookies = Jerry.Count_Cookies();
    printf("%X\n", "Jerry has this many cookies: ");
    printf("%X\n", jerrys_cookies);
}

Class MOUSE {

    char * name;
    int cookies;

    int Count_Cookies()
    {
    return cookies;
    }

}
```

Listing 667: test/test_output/semantic/test_simple_class.out

```
Semantic check succeded!
```

Listing 668: test/test_output/full_pipeline/test_simple_class.out

```
Jerry has this many cookies:
15
```

Listing 669: test/test_programs/test_simple_class_no_methods.meow

```
HAI ITZ ME FUNC Main,

    MAEK Jerry NEW MOUSE.
    cookies IN Jerry IZ 5.

    PSST Make sure that we can print the cookies in Jerry, which should be 5
    PURR Meow WIT cookies IN Jerry.

    PSST BLEEP Jerry.
KBYE

HAI ITZ ME CLASS MOUSE,
```

```
    ITZ ME NUMBR cookies.
    ITZ ME YARN name.
KBYE
```

Listing 670: test/test_output/ast/test_simple_class_no_methods.out

```


Main()
{
    class MOUSE Jerry;
    Jerry.cookies = 5;
    printf("%X\n", Jerry.cookies);
}

Class MOUSE {

    char * name;
    int cookies;


}
```

Listing 671: test/test_output/semantic/test_simple_class_no_methods.out

```
Semantic check succeded!
```

Listing 672: test/test_output/full_pipeline/test_simple_class_no_methods.out

```
5
```

Listing 673: test/test_programs/test_simple_functions.meow

```

HAI ITZ ME FUNC print_something,
        PURR Meow WIT "In the print_something function".
KBYE

HAI ITZ ME NUMBR FUNC Main,
        PSST Attempt to call a helper function
        PURR print_something.
        GIVE 0.
KBYE
```

Listing 674: test/test_output/ast/test_simple_functions.out

```

```

```
int Main()
{
    print_something();
    return 0;
}

print_something()
{
    printf("%X\n", "In the print_something function");
}
```

Listing 675: test/test_output/semantic/test_simple_functions.out

```
Semantic check succeded!
```

Listing 676: test/test_output/full_pipeline/test_simple_functions.out

```
In the print_something function
```

Listing 677: test/test_programs/test_simple_math.meow

```

HAI ITZ ME NUMBR FUNC Add_Three WIT NUMBR num1 AN NUMBR num2 AN NUMBR num3,
        PSST Trying to add three numbers

        ITZ ME NUMBR sum.
        sum IZ SUM OF num1 AN SUM OF num2 AN num3.
        GIVE sum.
KBYE

HAI ITZ ME NUMBR FUNC Subtract_Two WIT NUMBR num1 AN NUMBR num2,
        PSST Trying to add three numbers

        ITZ ME NUMBR diff.
        diff IZ DIFF OF num1 AN num2.
        GIVE diff.
KBYE

HAI ITZ ME NUMBR FUNC Multiply_Four WIT NUMBR num1 AN NUMBR num2
        AN NUMBR num3 AN NUMBR num4,

        PSST Trying to add multiply four numbers

        ITZ ME NUMBR product.
        product IZ PRODUKT OF num1 AN PRODUKT OF num2 AN PRODUKT OF num3 AN num4.
        GIVE product.
KBYE

```

```
HAI ITZ ME NUMBR FUNC Divide_Two WIT NUMBR num1 AN NUMBR num2,

        PSST Trying to divide two numbers

        ITZ ME NUMBR quotient.
        quotient IZ QUOSHUNT OF num1 AN num2.
        GIVE quotient.
KBYE


HAI ITZ ME NUMBR FUNC Main,

        ITZ ME NUMBR one IZ 1.
        ITZ ME NUMBR two IZ 2.
        ITZ ME NUMBR three IZ 3.
        ITZ ME NUMBR sum.
        ITZ ME NUMBR diff.
        ITZ ME NUMBR product.
        ITZ ME NUMBR quotient.

        PSST * ATTEMPT TO ADD *
        sum IZ PURR Add_Three WIT one AN two AN three.
        PURR Meow WIT sum.

        PSST * ATTEMPT TO SUBTRACT *
        diff IZ PURR Subtract_Two WIT three AN two.
        PURR Meow WIT diff.

        PSST * ATTEMPT TO MULTIPLY *
        product IZ PURR Multiply_Four WIT one AN two AN diff AN sum.
        PURR Meow WIT product.

        PSST * ATTEMPT TO DIVIDE *
        quotient IZ PURR Divide_Two WIT sum AN two.
        PURR Meow WIT quotient.

        GIVE 0.
KBYE
```

Listing 678: test/test_output/ast/test_simple_math.out

```
int Main()
{
    int one = 1;
    int two = 2;
    int three = 3;
```

```
    int sum;
    int diff;
    int product;
    int quotient;
    sum = Add_Three(one, two, three);
    printf("%X\n", sum);
    diff = Subtract_Two(three, two);
    printf("%X\n", diff);
    product = Multiply_Four(one, two, diff, sum);
    printf("%X\n", product);
    quotient = Divide_Two(sum, two);
    printf("%X\n", quotient);
    return 0;
}

int Divide_Two(int num1, int num2)
{
    int quotient;
    quotient = num1 / num2;
    return quotient;
}

int Multiply_Four(int num1, int num2, int num3, int num4)
{
    int product;
    product = num1 * num2 * num3 * num4;
    return product;
}

int Subtract_Two(int num1, int num2)
{
    int diff;
    diff = num1 - num2;
    return diff;
}

int Add_Three(int num1, int num2, int num3)
{
    int sum;
    sum = num1 + num2 + num3;
    return sum;
}
```

Listing 679: test/test_output/semantic/test_simple_math.out

```
Semantic check succeded!
```

Listing 680: test/test_output/full_pipeline/test_simple_math.out

```
6
1
12
3
```

Listing 681: test/test_programs/test_unop1.meow

```
HAI ITZ ME FUNC Main,

    PSST Needs to be boolean value
    ITZ ME BOO yes IZ AYE.
    ITZ ME BOO no IZ NAY.

    PURR Meow WIT NOT yes.
    PURR Meow WIT NOT no.

KBYE
```

Listing 682: test/test_output/ast/test_unop1.out

```


Main()
{
    bool yes = true;
    bool no = false;
    printf("%X\n", !yes);
    printf("%X\n", !no);
}
```

Listing 683: test/test_output/semantic/test_unop1.out

```
Semantic check succeded!
```

Listing 684: test/test_output/full_pipeline/test_unop1.out

```
0
1
```

Listing 685: test/test_programs/test_variables.meow

```

HAI ITZ ME NUMBR FUNC Main,

        PSST Creating some local variables
        ITZ ME NUMBR count IZ 2.
        ITZ ME NUMBAR random.
```

```
        ITZ ME BOO say_hello.
        ITZ ME YARN msg.

        PSST Assigning local varibles values
        msg IZ "hello world".
        say_hello IZ AYE.
        count IZ 2.
        random IZ 2.34.

        GIVE 0.
KBYE
```

Listing 686: `test/test_output/ast/test_variables.out`

```


int Main()
{
    int count = 2;
    float random;
    bool say_hello;
    char * msg;
    msg = "hello world";
    say_hello = true;
    count = 2;
    random = 2.34;
    return 0;
}
```

Listing 687: `test/test_output/semantic/test_variables.out`

```
Semantic check succeded!
```

Listing 688: `test/test_output/full_pipeline/test_variables.out`

8.3 Test Output

This appendix section includes the test output achieved when all tests pass successfully:

```
************************************************************
                  STARTING ALL CHECKS!
************************************************************
****************************************
 RUNNING CHECKS ON ABSTRACT SYNTAX TREE
****************************************
```

```
*** Running fail_variables_str_to_int (ast) ***
TEST PASSED

*** Running fail_variables_int_to_str (ast) ***
TEST PASSED

*** Running test_scan (ast) ***
TEST PASSED

*** Running fail_class6 (ast) ***
TEST PASSED

*** Running test_create_array5 (ast) ***
TEST PASSED

*** Running fail_binop6 (ast) ***
TEST PASSED

*** Running fail_concat_ints (ast) ***
TEST PASSED

*** Running test_comparison1 (ast) ***
TEST PASSED

Skipping accessory file ./cat_adventure_import.meow...

*** Running test_math_auto_cast (ast) ***
TEST PASSED

*** Running fail_method5 (ast) ***
TEST PASSED

*** Running test_array_access5 (ast) ***
TEST PASSED

*** Running fail_if1 (ast) ***
TEST PASSED

*** Running fail_func2 (ast) ***
TEST PASSED
```

```
*** Running fail_array_syntax4 (ast) ***
TEST PASSED

*** Running fail_array_access2 (ast) ***
TEST PASSED

*** Running test_meow_float (ast) ***
TEST PASSED

*** Running test_meow_bool (ast) ***
TEST PASSED

*** Running fail_array_too_many_elements (ast) ***
TEST PASSED

*** Running fail_create_instance4 (ast) ***
TEST PASSED

*** Running fail_concat_floats (ast) ***
TEST PASSED

Skipping accessory file ./multiple_imports.meow...

*** Running fail_array_holds_none (ast) ***
TEST PASSED

*** Running fail_create_instance5 (ast) ***
TEST PASSED

*** Running test_hello_world_complex (ast) ***
TEST PASSED

*** Running test_imports (ast) ***
TEST PASSED

Skipping accessory file ./import_colors.meow...

*** Running test_create_array8 (ast) ***
TEST PASSED
```

```
*** Running test_create_instance (ast) ***
TEST PASSED

*** Running fail_array_access3 (ast) ***
TEST PASSED

Skipping accessory file ./demo_concat.meow...

*** Running test_concat_int_str (ast) ***
TEST PASSED

*** Running fail_func3 (ast) ***
TEST PASSED

*** Running test_array_access4 (ast) ***
TEST PASSED

*** Running test_if2 (ast) ***
TEST PASSED

*** Running fail_scan_too_many_args (ast) ***
TEST PASSED

*** Running fail_method4 (ast) ***
TEST PASSED

*** Running test_concat (ast) ***
TEST PASSED

*** Running fail_binop7 (ast) ***
TEST PASSED

*** Running test_create_array4 (ast) ***
TEST PASSED

*** Running fail_class7 (ast) ***
TEST PASSED

*** Running fail_for1 (ast) ***
```

```
TEST PASSED

*** Running test_class_inner_method_call (ast) ***
TEST PASSED

*** Running test_hello_world (ast) ***
TEST PASSED

*** Running test_cast_str_to_float (ast) ***
TEST PASSED

*** Running fail_array_access4 (ast) ***
TEST PASSED

*** Running fail_concat_int_float (ast) ***
TEST PASSED

*** Running fail_class_access1 (ast) ***
TEST PASSED

*** Running test_simple_class_no_methods (ast) ***
TEST PASSED

*** Running fail_create_instance2 (ast) ***
TEST PASSED

*** Running test_mouse_class (ast) ***
TEST PASSED

*** Running fail_array_wrong_types1 (ast) ***
TEST PASSED

Skipping accessory file ./pet_store.meow...

*** Running fail_func4_syntax (ast) ***
TEST PASSED

Skipping accessory file ./demo_for.meow...

Skipping accessory file ./demo_array.meow...
```

```
*** Running test_create_array3 (ast) ***
TEST PASSED

*** Running fail_assign (ast) ***
TEST PASSED

*** Running test_comparison7 (ast) ***
TEST PASSED

*** Running test_cast_int_to_float (ast) ***
TEST PASSED

*** Running fail_method3 (ast) ***
TEST PASSED

*** Running fail_cast_object_to_string (ast) ***
TEST PASSED

*** Running test_array_access3 (ast) ***
TEST PASSED

*** Running fail_array_syntax2 (ast) ***
TEST PASSED

*** Running fail_array_syntax3 (ast) ***
TEST PASSED

*** Running test_array_access2 (ast) ***
TEST PASSED

*** Running test_unop1 (ast) ***
TEST PASSED

Skipping accessory file ./cat_adventure.meow...

*** Running fail_method2 (ast) ***
TEST PASSED

*** Running fail_syntax_comment (ast) ***
```

```
TEST PASSED

*** Running test_comparison6 (ast) ***
TEST PASSED

*** Running fail_binop1 (ast) ***
TEST PASSED

*** Running test_create_array2 (ast) ***
TEST PASSED

*** Running fail_class1 (ast) ***
TEST PASSED

*** Running fail_bad_import_names (ast) ***
TEST PASSED

*** Running fail_create_instance3 (ast) ***
TEST PASSED

*** Running fail_array_assignment1 (ast) ***
TEST PASSED

*** Running test_scan2 (ast) ***
TEST PASSED

*** Running test_array_assignment1 (ast) ***
TEST PASSED

*** Running fail_array_access5 (ast) ***
TEST PASSED

*** Running test_concat_str_int (ast) ***
TEST PASSED

*** Running fail_cast_string_to_object (ast) ***
TEST PASSED

*** Running fail_array_assignment2 (ast) ***
TEST PASSED
```

```
*** Running test_comparison9 (ast) ***
TEST PASSED

*** Running fail_func6_syntax (ast) ***
TEST PASSED

*** Running test_array_assignment2 (ast) ***
TEST PASSED

*** Running fail_array_access6 (ast) ***
TEST PASSED

*** Running test_pass_obj_as_param (ast) ***
TEST PASSED

*** Running fail_array_size_wrong_type (ast) ***
TEST PASSED

*** Running fail_array_mixed_types (ast) ***
TEST PASSED

Skipping accessory file ./inner_import_example.meow...

*** Running test_array_access1 (ast) ***
TEST PASSED

Skipping accessory file ./fstphrase.meow...

*** Running test_concat_str_float (ast) ***
TEST PASSED

*** Running fail_method1 (ast) ***
TEST PASSED

*** Running test_meow_string (ast) ***
TEST PASSED

*** Running test_comparison5 (ast) ***
TEST PASSED
```

```
*** Running fail_binop2 (ast) ***
TEST PASSED

*** Running test_create_array1 (ast) ***
TEST PASSED

*** Running test_cast_float_to_int (ast) ***
TEST PASSED

*** Running fail_class2 (ast) ***
TEST PASSED

*** Running test_class_specify_constructor_some (ast) ***
TEST PASSED

*** Running fail_for4 (ast) ***
TEST PASSED

*** Running fail_for5 (ast) ***
TEST PASSED

*** Running fail_concat_bool (ast) ***
TEST PASSED

*** Running test_cast_float_to_str (ast) ***
TEST PASSED

*** Running fail_class3 (ast) ***
TEST PASSED

*** Running fail_binop3 (ast) ***
TEST PASSED

*** Running test_comparison4 (ast) ***
TEST PASSED

*** Running test_mouse_class_two (ast) ***
TEST PASSED
```

```
*** Running fail_cast_redundant (ast) ***
TEST PASSED

*** Running test_class_default_vars (ast) ***
TEST PASSED

*** Running fail_if4 (ast) ***
TEST PASSED

*** Running fail_array_syntax1 (ast) ***
TEST PASSED

*** Running fail_import_not_found (ast) ***
TEST PASSED

*** Running test_simple_functions (ast) ***
TEST PASSED

*** Running fail_syntax_variables (ast) ***
TEST PASSED

*** Running fail_array_element_not_defined (ast) ***
TEST PASSED

*** Running test_array_assignment3 (ast) ***
TEST PASSED

*** Running fail_class_access2 (ast) ***
TEST PASSED

*** Running test_comparison8 (ast) ***
TEST PASSED

*** Running test_pointless (ast) ***
TEST PASSED

*** Running fail_create_instance1 (ast) ***
TEST PASSED

*** Running fail_array_assignment3 (ast) ***
```

```
TEST PASSED

*** Running fail_array_wrong_types2 (ast) ***
TEST PASSED

Skipping accessory file ./sndphrase.meow...

*** Running fail_if3 (ast) ***
TEST PASSED

*** Running test_if1 (ast) ***
TEST PASSED

*** Running fail_array_size_not_defined (ast) ***
TEST PASSED

*** Running fail_method7 (ast) ***
TEST PASSED

*** Running fail_unop (ast) ***
TEST PASSED

*** Running fail_binop10 (ast) ***
TEST PASSED

*** Running test_pass_array_to_method (ast) ***
TEST PASSED

Skipping accessory file ./import_bridge.meow...

*** Running test_concat_float_str (ast) ***
TEST PASSED

*** Running test_cast_int_to_str (ast) ***
TEST PASSED

*** Running test_cast_str_to_int (ast) ***
TEST PASSED

*** Running test_comparison3 (ast) ***
```

```
TEST PASSED

*** Running test_pass_array_to_function (ast) ***
TEST PASSED

*** Running fail_binop4 (ast) ***
TEST PASSED

*** Running test_create_array7 (ast) ***
TEST PASSED

*** Running fail_class4 (ast) ***
TEST PASSED

*** Running fail_for2 (ast) ***
TEST PASSED

*** Running fail_array_assignment4 (ast) ***
TEST PASSED

*** Running fail_class_default_vars (ast) ***
TEST PASSED

*** Running test_classes_demo (ast) ***
TEST PASSED

*** Running fail_duplicate_import (ast) ***
TEST PASSED

*** Running test_array_assignment4 (ast) ***
TEST PASSED

*** Running test_for (ast) ***
TEST PASSED

*** Running test_variables (ast) ***
TEST PASSED

*** Running fail_binop8 (ast) ***
TEST PASSED
```

```
*** Running test_simple_math (ast) ***
TEST PASSED

*** Running fail_array_access1 (ast) ***
TEST PASSED

*** Running fail_binop9 (ast) ***
TEST PASSED

Skipping accessory file ./demo_cast.meow...

*** Running test_meow_int (ast) ***
TEST PASSED

Skipping accessory file ./pets.meow...

Skipping accessory file ./import_example.meow...

*** Running fail_pass_wrong_obj_as_param (ast) ***
TEST PASSED

*** Running fail_scan_wrong_type (ast) ***
TEST PASSED

*** Running fail_for3 (ast) ***
TEST PASSED

*** Running fail_class5 (ast) ***
TEST PASSED

*** Running test_create_array6 (ast) ***
TEST PASSED

*** Running fail_func5_syntax (ast) ***
TEST PASSED

*** Running fail_binop5 (ast) ***
TEST PASSED
```

```
*** Running test_comparison2 (ast) ***
TEST PASSED

*** Running test_simple_class (ast) ***
TEST PASSED

*** Running test_class_specify_constructor (ast) ***
TEST PASSED

*** Running fail_binop11 (ast) ***
TEST PASSED

*** Running fail_method6 (ast) ***
TEST PASSED

*** Running fail_variables_str_to_bool (ast) ***
TEST PASSED

*** Running fail_if2 (ast) ***
TEST PASSED

*** Running fail_func1 (ast) ***
TEST PASSED

*** 158 successful ast tests completed! Good to go! ***
****************************************
     RUNNING CHECKS ON SEMANTICS
****************************************

*** Running fail_variables_str_to_int (semantic) ***
TEST PASSED

*** Running fail_variables_int_to_str (semantic) ***
TEST PASSED

*** Running test_scan (semantic) ***
TEST PASSED

*** Running fail_class6 (semantic) ***
TEST PASSED
```

```
*** Running test_create_array5 (semantic) ***
TEST PASSED

*** Running fail_binop6 (semantic) ***
TEST PASSED

*** Running fail_concat_ints (semantic) ***
TEST PASSED

*** Running test_comparison1 (semantic) ***
TEST PASSED

Skipping accessory file ./cat_adventure_import.meow...

*** Running test_math_auto_cast (semantic) ***
TEST PASSED

*** Running fail_method5 (semantic) ***
TEST PASSED

*** Running test_array_access5 (semantic) ***
TEST PASSED

*** Running fail_if1 (semantic) ***
TEST PASSED

*** Running fail_func2 (semantic) ***
TEST PASSED

*** Running fail_array_syntax4 (semantic) ***
TEST PASSED

*** Running fail_array_access2 (semantic) ***
TEST PASSED

*** Running test_meow_float (semantic) ***
TEST PASSED

*** Running test_meow_bool (semantic) ***
```

```
TEST PASSED

*** Running fail_array_too_many_elements (semantic) ***
TEST PASSED

*** Running fail_create_instance4 (semantic) ***
TEST PASSED

*** Running fail_concat_floats (semantic) ***
TEST PASSED

Skipping accessory file ./multiple_imports.meow...

*** Running fail_array_holds_none (semantic) ***
TEST PASSED

*** Running fail_create_instance5 (semantic) ***
TEST PASSED

*** Running test_hello_world_complex (semantic) ***
TEST PASSED

*** Running test_imports (semantic) ***
TEST PASSED

Skipping accessory file ./import_colors.meow...

*** Running test_create_array8 (semantic) ***
TEST PASSED

*** Running test_create_instance (semantic) ***
TEST PASSED

*** Running fail_array_access3 (semantic) ***
TEST PASSED

Skipping accessory file ./demo_concat.meow...

*** Running test_concat_int_str (semantic) ***
TEST PASSED
```

```
*** Running fail_func3 (semantic) ***
TEST PASSED

*** Running test_array_access4 (semantic) ***
TEST PASSED

*** Running test_if2 (semantic) ***
TEST PASSED

*** Running fail_scan_too_many_args (semantic) ***
TEST PASSED

*** Running fail_method4 (semantic) ***
TEST PASSED

*** Running test_concat (semantic) ***
TEST PASSED

*** Running fail_binop7 (semantic) ***
TEST PASSED

*** Running test_create_array4 (semantic) ***
TEST PASSED

*** Running fail_class7 (semantic) ***
TEST PASSED

*** Running fail_for1 (semantic) ***
TEST PASSED

*** Running test_class_inner_method_call (semantic) ***
TEST PASSED

*** Running test_hello_world (semantic) ***
TEST PASSED

*** Running test_cast_str_to_float (semantic) ***
TEST PASSED
```

```
*** Running fail_array_access4 (semantic) ***
TEST PASSED

*** Running fail_concat_int_float (semantic) ***
TEST PASSED

*** Running fail_class_access1 (semantic) ***
TEST PASSED

*** Running test_simple_class_no_methods (semantic) ***
TEST PASSED

*** Running fail_create_instance2 (semantic) ***
TEST PASSED

*** Running test_mouse_class (semantic) ***
TEST PASSED

*** Running fail_array_wrong_types1 (semantic) ***
TEST PASSED

Skipping accessory file ./pet_store.meow...

*** Running fail_func4_syntax (semantic) ***
TEST PASSED

Skipping accessory file ./demo_for.meow...

Skipping accessory file ./demo_array.meow...

*** Running test_create_array3 (semantic) ***
TEST PASSED

*** Running fail_assign (semantic) ***
TEST PASSED

*** Running test_comparison7 (semantic) ***
TEST PASSED

*** Running test_cast_int_to_float (semantic) ***
```

```
TEST PASSED

*** Running fail_method3 (semantic) ***
TEST PASSED

*** Running fail_cast_object_to_string (semantic) ***
TEST PASSED

*** Running test_array_access3 (semantic) ***
TEST PASSED

*** Running fail_array_syntax2 (semantic) ***
TEST PASSED

*** Running fail_array_syntax3 (semantic) ***
TEST PASSED

*** Running test_array_access2 (semantic) ***
TEST PASSED

*** Running test_unop1 (semantic) ***
TEST PASSED

Skipping accessory file ./cat_adventure.meow...

*** Running fail_method2 (semantic) ***
TEST PASSED

*** Running fail_syntax_comment (semantic) ***
TEST PASSED

*** Running test_comparison6 (semantic) ***
TEST PASSED

*** Running fail_binop1 (semantic) ***
TEST PASSED

*** Running test_create_array2 (semantic) ***
TEST PASSED
```

```
*** Running fail_class1 (semantic) ***
TEST PASSED

*** Running fail_bad_import_names (semantic) ***
TEST PASSED

*** Running fail_create_instance3 (semantic) ***
TEST PASSED

*** Running fail_array_assignment1 (semantic) ***
TEST PASSED

*** Running test_scan2 (semantic) ***
TEST PASSED

*** Running test_array_assignment1 (semantic) ***
TEST PASSED

*** Running fail_array_access5 (semantic) ***
TEST PASSED

*** Running test_concat_str_int (semantic) ***
TEST PASSED

*** Running fail_cast_string_to_object (semantic) ***
TEST PASSED

*** Running fail_array_assignment2 (semantic) ***
TEST PASSED

*** Running test_comparison9 (semantic) ***
TEST PASSED

*** Running fail_func6_syntax (semantic) ***
TEST PASSED

*** Running test_array_assignment2 (semantic) ***
TEST PASSED

*** Running fail_array_access6 (semantic) ***
```

```
TEST PASSED

*** Running test_pass_obj_as_param (semantic) ***
TEST PASSED

*** Running fail_array_size_wrong_type (semantic) ***
TEST PASSED

*** Running fail_array_mixed_types (semantic) ***
TEST PASSED

Skipping accessory file ./inner_import_example.meow...

*** Running test_array_access1 (semantic) ***
TEST PASSED

Skipping accessory file ./fstphrase.meow...

*** Running test_concat_str_float (semantic) ***
TEST PASSED

*** Running fail_method1 (semantic) ***
TEST PASSED

*** Running test_meow_string (semantic) ***
TEST PASSED

*** Running test_comparison5 (semantic) ***
TEST PASSED

*** Running fail_binop2 (semantic) ***
TEST PASSED

*** Running test_create_array1 (semantic) ***
TEST PASSED

*** Running test_cast_float_to_int (semantic) ***
TEST PASSED

*** Running fail_class2 (semantic) ***
```

```
TEST PASSED

*** Running test_class_specify_constructor_some (semantic) ***
TEST PASSED

*** Running fail_for4 (semantic) ***
TEST PASSED

*** Running fail_for5 (semantic) ***
TEST PASSED

*** Running fail_concat_bool (semantic) ***
TEST PASSED

*** Running test_cast_float_to_str (semantic) ***
TEST PASSED

*** Running fail_class3 (semantic) ***
TEST PASSED

*** Running fail_binop3 (semantic) ***
TEST PASSED

*** Running test_comparison4 (semantic) ***
TEST PASSED

*** Running test_mouse_class_two (semantic) ***
TEST PASSED

*** Running fail_cast_redundant (semantic) ***
TEST PASSED

*** Running test_class_default_vars (semantic) ***
TEST PASSED

*** Running fail_if4 (semantic) ***
TEST PASSED

*** Running fail_array_syntax1 (semantic) ***
TEST PASSED
```

```
*** Running fail_import_not_found (semantic) ***
TEST PASSED

*** Running test_simple_functions (semantic) ***
TEST PASSED

*** Running fail_syntax_variables (semantic) ***
TEST PASSED

*** Running fail_array_element_not_defined (semantic) ***
TEST PASSED

*** Running test_array_assignment3 (semantic) ***
TEST PASSED

*** Running fail_class_access2 (semantic) ***
TEST PASSED

*** Running test_comparison8 (semantic) ***
TEST PASSED

*** Running test_pointless (semantic) ***
TEST PASSED

*** Running fail_create_instance1 (semantic) ***
TEST PASSED

*** Running fail_array_assignment3 (semantic) ***
TEST PASSED

*** Running fail_array_wrong_types2 (semantic) ***
TEST PASSED

Skipping accessory file ./sndphrase.meow...

*** Running fail_if3 (semantic) ***
TEST PASSED

*** Running test_if1 (semantic) ***
```

```
TEST PASSED

*** Running fail_array_size_not_defined (semantic) ***
TEST PASSED

*** Running fail_method7 (semantic) ***
TEST PASSED

*** Running fail_unop (semantic) ***
TEST PASSED

*** Running fail_binop10 (semantic) ***
TEST PASSED

*** Running test_pass_array_to_method (semantic) ***
TEST PASSED

Skipping accessory file ./import_bridge.meow...

*** Running test_concat_float_str (semantic) ***
TEST PASSED

*** Running test_cast_int_to_str (semantic) ***
TEST PASSED

*** Running test_cast_str_to_int (semantic) ***
TEST PASSED

*** Running test_comparison3 (semantic) ***
TEST PASSED

*** Running test_pass_array_to_function (semantic) ***
TEST PASSED

*** Running fail_binop4 (semantic) ***
TEST PASSED

*** Running test_create_array7 (semantic) ***
TEST PASSED
```

```
*** Running fail_class4 (semantic) ***
TEST PASSED

*** Running fail_for2 (semantic) ***
TEST PASSED

*** Running fail_array_assignment4 (semantic) ***
TEST PASSED

*** Running fail_class_default_vars (semantic) ***
TEST PASSED

*** Running test_classes_demo (semantic) ***
TEST PASSED

*** Running fail_duplicate_import (semantic) ***
TEST PASSED

*** Running test_array_assignment4 (semantic) ***
TEST PASSED

*** Running test_for (semantic) ***
TEST PASSED

*** Running test_variables (semantic) ***
TEST PASSED

*** Running fail_binop8 (semantic) ***
TEST PASSED

*** Running test_simple_math (semantic) ***
TEST PASSED

*** Running fail_array_access1 (semantic) ***
TEST PASSED

*** Running fail_binop9 (semantic) ***
TEST PASSED

Skipping accessory file ./demo_cast.meow...
```

```
*** Running test_meow_int (semantic) ***
TEST PASSED

Skipping accessory file ./pets.meow...

Skipping accessory file ./import_example.meow...

*** Running fail_pass_wrong_obj_as_param (semantic) ***
TEST PASSED

*** Running fail_scan_wrong_type (semantic) ***
TEST PASSED

*** Running fail_for3 (semantic) ***
TEST PASSED

*** Running fail_class5 (semantic) ***
TEST PASSED

*** Running test_create_array6 (semantic) ***
TEST PASSED

*** Running fail_func5_syntax (semantic) ***
TEST PASSED

*** Running fail_binop5 (semantic) ***
TEST PASSED

*** Running test_comparison2 (semantic) ***
TEST PASSED

*** Running test_simple_class (semantic) ***
TEST PASSED

*** Running test_class_specify_constructor (semantic) ***
TEST PASSED

*** Running fail_binop11 (semantic) ***
TEST PASSED
```

```
*** Running fail_method6 (semantic) ***
TEST PASSED

*** Running fail_variables_str_to_bool (semantic) ***
TEST PASSED

*** Running fail_if2 (semantic) ***
TEST PASSED

*** Running fail_func1 (semantic) ***
TEST PASSED

*** 158 successful semantic tests completed! Good to go! ***
****************************************
    RUNNING CHECKS ON FULL PIPELINE
****************************************

*** Running fail_variables_str_to_int (full_pipeline) ***
TEST PASSED

*** Running fail_variables_int_to_str (full_pipeline) ***
TEST PASSED

*** Running test_scan (full_pipeline) ***
TEST PASSED

*** Running fail_class6 (full_pipeline) ***
TEST PASSED

*** Running test_create_array5 (full_pipeline) ***
TEST PASSED

*** Running fail_binop6 (full_pipeline) ***
TEST PASSED

*** Running fail_concat_ints (full_pipeline) ***
TEST PASSED

*** Running test_comparison1 (full_pipeline) ***
```

```
TEST PASSED

Skipping accessory file ./cat_adventure_import.meow...

*** Running test_math_auto_cast (full_pipeline) ***
TEST PASSED

*** Running fail_method5 (full_pipeline) ***
TEST PASSED

*** Running test_array_access5 (full_pipeline) ***
TEST PASSED

*** Running fail_if1 (full_pipeline) ***
TEST PASSED

*** Running fail_func2 (full_pipeline) ***
TEST PASSED

*** Running fail_array_syntax4 (full_pipeline) ***
TEST PASSED

*** Running fail_array_access2 (full_pipeline) ***
TEST PASSED

*** Running test_meow_float (full_pipeline) ***
TEST PASSED

*** Running test_meow_bool (full_pipeline) ***
TEST PASSED

*** Running fail_array_too_many_elements (full_pipeline) ***
TEST PASSED

*** Running fail_create_instance4 (full_pipeline) ***
TEST PASSED

*** Running fail_concat_floats (full_pipeline) ***
TEST PASSED
```

```
Skipping accessory file ./multiple_imports.meow...

*** Running fail_array_holds_none (full_pipeline) ***
TEST PASSED

*** Running fail_create_instance5 (full_pipeline) ***
TEST PASSED

*** Running test_hello_world_complex (full_pipeline) ***
TEST PASSED

*** Running test_imports (full_pipeline) ***
TEST PASSED

Skipping accessory file ./import_colors.meow...

*** Running test_create_array8 (full_pipeline) ***
TEST PASSED

*** Running test_create_instance (full_pipeline) ***
TEST PASSED

*** Running fail_array_access3 (full_pipeline) ***
TEST PASSED

Skipping accessory file ./demo_concat.meow...

*** Running test_concat_int_str (full_pipeline) ***
TEST PASSED

*** Running fail_func3 (full_pipeline) ***
TEST PASSED

*** Running test_array_access4 (full_pipeline) ***
TEST PASSED

*** Running test_if2 (full_pipeline) ***
TEST PASSED

*** Running fail_scan_too_many_args (full_pipeline) ***
```

```
TEST PASSED

*** Running fail_method4 (full_pipeline) ***
TEST PASSED

*** Running test_concat (full_pipeline) ***
TEST PASSED

*** Running fail_binop7 (full_pipeline) ***
TEST PASSED

*** Running test_create_array4 (full_pipeline) ***
TEST PASSED

*** Running fail_class7 (full_pipeline) ***
TEST PASSED

*** Running fail_for1 (full_pipeline) ***
TEST PASSED

*** Running test_class_inner_method_call (full_pipeline) ***
TEST PASSED

*** Running test_hello_world (full_pipeline) ***
TEST PASSED

*** Running test_cast_str_to_float (full_pipeline) ***
TEST PASSED

*** Running fail_array_access4 (full_pipeline) ***
TEST PASSED

*** Running fail_concat_int_float (full_pipeline) ***
TEST PASSED

*** Running fail_class_access1 (full_pipeline) ***
TEST PASSED

*** Running test_simple_class_no_methods (full_pipeline) ***
TEST PASSED
```

```
*** Running fail_create_instance2 (full_pipeline) ***
TEST PASSED

*** Running test_mouse_class (full_pipeline) ***
TEST PASSED

*** Running fail_array_wrong_types1 (full_pipeline) ***
TEST PASSED

Skipping accessory file ./pet_store.meow...

*** Running fail_func4_syntax (full_pipeline) ***
TEST PASSED

Skipping accessory file ./demo_for.meow...

Skipping accessory file ./demo_array.meow...

*** Running test_create_array3 (full_pipeline) ***
TEST PASSED

*** Running fail_assign (full_pipeline) ***
TEST PASSED

*** Running test_comparison7 (full_pipeline) ***
TEST PASSED

*** Running test_cast_int_to_float (full_pipeline) ***
TEST PASSED

*** Running fail_method3 (full_pipeline) ***
TEST PASSED

*** Running fail_cast_object_to_string (full_pipeline) ***
TEST PASSED

*** Running test_array_access3 (full_pipeline) ***
TEST PASSED
```

```
*** Running fail_array_syntax2 (full_pipeline) ***
TEST PASSED

*** Running fail_array_syntax3 (full_pipeline) ***
TEST PASSED

*** Running test_array_access2 (full_pipeline) ***
TEST PASSED

*** Running test_unop1 (full_pipeline) ***
TEST PASSED

Skipping accessory file ./cat_adventure.meow...

*** Running fail_method2 (full_pipeline) ***
TEST PASSED

*** Running fail_syntax_comment (full_pipeline) ***
TEST PASSED

*** Running test_comparison6 (full_pipeline) ***
TEST PASSED

*** Running fail_binop1 (full_pipeline) ***
TEST PASSED

*** Running test_create_array2 (full_pipeline) ***
TEST PASSED

*** Running fail_class1 (full_pipeline) ***
TEST PASSED

*** Running fail_bad_import_names (full_pipeline) ***
TEST PASSED

*** Running fail_create_instance3 (full_pipeline) ***
TEST PASSED

*** Running fail_array_assignment1 (full_pipeline) ***
TEST PASSED
```

```
*** Running test_scan2 (full_pipeline) ***
TEST PASSED

*** Running test_array_assignment1 (full_pipeline) ***
TEST PASSED

*** Running fail_array_access5 (full_pipeline) ***
TEST PASSED

*** Running test_concat_str_int (full_pipeline) ***
TEST PASSED

*** Running fail_cast_string_to_object (full_pipeline) ***
TEST PASSED

*** Running fail_array_assignment2 (full_pipeline) ***
TEST PASSED

*** Running test_comparison9 (full_pipeline) ***
TEST PASSED

*** Running fail_func6_syntax (full_pipeline) ***
TEST PASSED

*** Running test_array_assignment2 (full_pipeline) ***
TEST PASSED

*** Running fail_array_access6 (full_pipeline) ***
TEST PASSED

*** Running test_pass_obj_as_param (full_pipeline) ***
TEST PASSED

*** Running fail_array_size_wrong_type (full_pipeline) ***
TEST PASSED

*** Running fail_array_mixed_types (full_pipeline) ***
TEST PASSED
```

```
Skipping accessory file ./inner_import_example.meow...

*** Running test_array_access1 (full_pipeline) ***
TEST PASSED

Skipping accessory file ./fstphrase.meow...

*** Running test_concat_str_float (full_pipeline) ***
TEST PASSED

*** Running fail_method1 (full_pipeline) ***
TEST PASSED

*** Running test_meow_string (full_pipeline) ***
TEST PASSED

*** Running test_comparison5 (full_pipeline) ***
TEST PASSED

*** Running fail_binop2 (full_pipeline) ***
TEST PASSED

*** Running test_create_array1 (full_pipeline) ***
TEST PASSED

*** Running test_cast_float_to_int (full_pipeline) ***
TEST PASSED

*** Running fail_class2 (full_pipeline) ***
TEST PASSED

*** Running test_class_specify_constructor_some (full_pipeline) ***
TEST PASSED

*** Running fail_for4 (full_pipeline) ***
TEST PASSED

*** Running fail_for5 (full_pipeline) ***
TEST PASSED
```

```
*** Running fail_concat_bool (full_pipeline) ***
TEST PASSED

*** Running test_cast_float_to_str (full_pipeline) ***
TEST PASSED

*** Running fail_class3 (full_pipeline) ***
TEST PASSED

*** Running fail_binop3 (full_pipeline) ***
TEST PASSED

*** Running test_comparison4 (full_pipeline) ***
TEST PASSED

*** Running test_mouse_class_two (full_pipeline) ***
TEST PASSED

*** Running fail_cast_redundant (full_pipeline) ***
TEST PASSED

*** Running test_class_default_vars (full_pipeline) ***
TEST PASSED

*** Running fail_if4 (full_pipeline) ***
TEST PASSED

*** Running fail_array_syntax1 (full_pipeline) ***
TEST PASSED

*** Running fail_import_not_found (full_pipeline) ***
TEST PASSED

*** Running test_simple_functions (full_pipeline) ***
TEST PASSED

*** Running fail_syntax_variables (full_pipeline) ***
TEST PASSED

*** Running fail_array_element_not_defined (full_pipeline) ***
```

```
TEST PASSED

*** Running test_array_assignment3 (full_pipeline) ***
TEST PASSED

*** Running fail_class_access2 (full_pipeline) ***
TEST PASSED

*** Running test_comparison8 (full_pipeline) ***
TEST PASSED

*** Running test_pointless (full_pipeline) ***
TEST PASSED

*** Running fail_create_instance1 (full_pipeline) ***
TEST PASSED

*** Running fail_array_assignment3 (full_pipeline) ***
TEST PASSED

*** Running fail_array_wrong_types2 (full_pipeline) ***
TEST PASSED

Skipping accessory file ./sndphrase.meow...

*** Running fail_if3 (full_pipeline) ***
TEST PASSED

*** Running test_if1 (full_pipeline) ***
TEST PASSED

*** Running fail_array_size_not_defined (full_pipeline) ***
TEST PASSED

*** Running fail_method7 (full_pipeline) ***
TEST PASSED

*** Running fail_unop (full_pipeline) ***
TEST PASSED
```

```
*** Running fail_binop10 (full_pipeline) ***
TEST PASSED

*** Running test_pass_array_to_method (full_pipeline) ***
TEST PASSED

Skipping accessory file ./import_bridge.meow...

*** Running test_concat_float_str (full_pipeline) ***
TEST PASSED

*** Running test_cast_int_to_str (full_pipeline) ***
TEST PASSED

*** Running test_cast_str_to_int (full_pipeline) ***
TEST PASSED

*** Running test_comparison3 (full_pipeline) ***
TEST PASSED

*** Running test_pass_array_to_function (full_pipeline) ***
TEST PASSED

*** Running fail_binop4 (full_pipeline) ***
TEST PASSED

*** Running test_create_array7 (full_pipeline) ***
TEST PASSED

*** Running fail_class4 (full_pipeline) ***
TEST PASSED

*** Running fail_for2 (full_pipeline) ***
TEST PASSED

*** Running fail_array_assignment4 (full_pipeline) ***
TEST PASSED

*** Running fail_class_default_vars (full_pipeline) ***
TEST PASSED
```

```
*** Running test_classes_demo (full_pipeline) ***
TEST PASSED

*** Running fail_duplicate_import (full_pipeline) ***
TEST PASSED

*** Running test_array_assignment4 (full_pipeline) ***
TEST PASSED

*** Running test_for (full_pipeline) ***
TEST PASSED

*** Running test_variables (full_pipeline) ***
TEST PASSED

*** Running fail_binop8 (full_pipeline) ***
TEST PASSED

*** Running test_simple_math (full_pipeline) ***
TEST PASSED

*** Running fail_array_access1 (full_pipeline) ***
TEST PASSED

*** Running fail_binop9 (full_pipeline) ***
TEST PASSED

Skipping accessory file ./demo_cast.meow...

*** Running test_meow_int (full_pipeline) ***
TEST PASSED

Skipping accessory file ./pets.meow...

Skipping accessory file ./import_example.meow...

*** Running fail_pass_wrong_obj_as_param (full_pipeline) ***
TEST PASSED
```

```
*** Running fail_scan_wrong_type (full_pipeline) ***
TEST PASSED

*** Running fail_for3 (full_pipeline) ***
TEST PASSED

*** Running fail_class5 (full_pipeline) ***
TEST PASSED

*** Running test_create_array6 (full_pipeline) ***
TEST PASSED

*** Running fail_func5_syntax (full_pipeline) ***
TEST PASSED

*** Running fail_binop5 (full_pipeline) ***
TEST PASSED

*** Running test_comparison2 (full_pipeline) ***
TEST PASSED

*** Running test_simple_class (full_pipeline) ***
TEST PASSED

*** Running test_class_specify_constructor (full_pipeline) ***
TEST PASSED

*** Running fail_binop11 (full_pipeline) ***
TEST PASSED

*** Running fail_method6 (full_pipeline) ***
TEST PASSED

*** Running fail_variables_str_to_bool (full_pipeline) ***
TEST PASSED

*** Running fail_if2 (full_pipeline) ***
TEST PASSED

*** Running fail_func1 (full_pipeline) ***
```

```
TEST PASSED

*** 158 successful full_pipeline tests completed! Good to go! ***

**********************************************************
                   ALL CHECKS PASS!
**********************************************************
```

8.4 Git History

```
commit 98c11b5b709958597d115960395713f06d12ae4f
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Apr 25 16:59:35 2021 -0400

    final test updates

commit 49a26d5f7b0b97a1494fdd47f839458fe7488e44
Merge: cc1dff5 6c81c10
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Apr 25 16:51:34 2021 -0400

    Merge branch 'main' of github.com:mmfrenkel/meowlang into main

commit cc1dff5a8480ac0f358c4bdca3a971859409978a
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Apr 25 16:51:27 2021 -0400

    supporting demo programs

commit 6c81c1018c325b05ff4aaa15cef8b108a6536ffc
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 24 20:53:09 2021 -0400

    demo array is now a simpler example

commit 99d1b615bc584011e8cceabec055830cf90e778e
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 24 20:52:47 2021 -0400

    now passes check

commit 7a04df9a5a6cadf0fb3cb6435b9bb052930b3eb2
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 24 10:41:48 2021 -0400

    use 2.8 as an example to demonstrate truncation instead of rounding

commit 798fc59c05d7315d2940d95c80d1c5fb995bc4db
Author: Lauren Pham <lauren.pham@focus.org>
```

```
Date:   Sat Apr 24 10:09:17 2021 -0400

    demo program for arrays

commit 08cc9d07999642ff81f25f32a4bf81d25fdf7365
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 24 10:00:16 2021 -0400

    demo program for for loop

commit 73aa2bcfdd7cb78bb9ca380c4cd919e109e976de
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 24 09:57:50 2021 -0400

    demo program for concat

commit 9baca7f6641cfce6eb047129a316099fd622bde8
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 24 09:57:03 2021 -0400

    spacing

commit ed1b15d82b65614613e030943aa9be56aee45a42
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 24 09:51:37 2021 -0400

    demo program for casting

commit 03576ab40bded11012f3ca6293083ed8f1b74272
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sat Apr 24 09:49:45 2021 -0400

    update documentation

commit 84949a3be42c6d4a057b71332fac5f240dee602e
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sat Apr 24 09:45:41 2021 -0400

    new scan test
```

```
commit 3596d76270b81b383f9ee93d7d50441da81075c9
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sat Apr 24 09:41:05 2021 -0400

    update cat adventure

commit d70a5fe7e3703f2702756017b7767b7559952212
Merge: d19f65b 1c87b3d
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Fri Apr 23 13:32:06 2021 -0400

    update readme

commit d19f65b7f817a8492cc7741f3680e0fd926bb743
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Fri Apr 23 13:31:20 2021 -0400

    close to last update

commit 1c87b3d05e1122322a6dd843a0097f5dc2e43c2b
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Wed Apr 21 20:28:08 2021 -0400

    Update README.md

commit 4f3f35e628f3a6493e15feb29dbf109777c61303
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Wed Apr 21 20:24:18 2021 -0400

    update to create meowlang.sh script

commit a318d49aa89279e8fc5379fa69f6fd3debf4561c
Author: Michelle <ml4080@columbia.edu>
Date:   Tue Apr 20 22:11:36 2021 -0400

    Added imports to demo

commit 2895c019318b0987ba5297def645e42a05dc8bdd
Merge: 8365534 d355521
Author: Carolyn Chen <chen.cec@gmail.com>
```

```
Date:   Tue Apr 20 19:42:03 2021 -0400

    Merge branch 'imports-testing' into main

commit d3555210f927e670cd9f4b79f9bead64ee4ceb4a
Author: Carolyn Chen <chen.cec@gmail.com>
Date:   Tue Apr 20 19:41:22 2021 -0400

    added fail import test

    No ImportNotFound fail test due to unique file path

commit 83655343447aaf8fbcb87406513e3b4fa8e0b814
Author: Michelle <ml4080@columbia.edu>
Date:   Tue Apr 20 17:36:10 2021 -0400

    Fixed spacing for outputs

commit 14d273ad3e3868e9ad64cc5da659a352e2acb252
Author: Michelle <ml4080@columbia.edu>
Date:   Tue Apr 20 16:42:09 2021 -0400

    Added separate functions to demo

commit 7a130206c9264eeae287f745e116d5c6af6deff3
Author: Michelle <ml4080@columbia.edu>
Date:   Tue Apr 20 16:22:26 2021 -0400

    Added concat

commit 2fe03ebfb667a88e8cc3fdde07d0bca6eaa9dafc
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Tue Apr 20 15:11:09 2021 -0400

    update printing on regression test for accessory files

commit 6945e750956c024a5afdd9dee5cdc522b9a44f6d
Author: Carolyn Chen <chen.cec@gmail.com>
Date:   Tue Apr 20 14:50:18 2021 -0400
```

```
    Circular imports and duplicate imports exceptions working

commit 9c05e67346a24df00a9f1606289dbb3b28cebba3
Author: Carolyn Chen <chen.cec@gmail.com>
Date:   Mon Apr 19 16:05:47 2021 -0400

    Update _tags

    Remove package(str) from _tags

commit 7488bf0afb6720105c5476d794e59fe8daedddd6
Merge: d87f539 9b62090
Author: Carolyn Chen <chen.cec@gmail.com>
Date:   Mon Apr 19 14:00:32 2021 -0400

    Merge branch 'semant-imports' into main

commit 9b62090fa305e5a4b2c46366bcb20c7657f48dfe
Merge: a439519 e21fee5
Author: Carolyn Chen <chen.cec@gmail.com>
Date:   Mon Apr 19 13:43:13 2021 -0400

    Merge branch 'semant-imports' of https://github.com/mmfrenkel/meowlang into semant-imports

commit d87f5395ee50eaab643f8acf9087cb9a1d924148
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Mon Apr 19 13:07:16 2021 -0400

    comments

commit f3c3347294f414aa792d6e834716c7e59fad79bf
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Mon Apr 19 12:55:15 2021 -0400

    BLEEP concat objects

commit e21fee5aaa3936c031f7f1f9f34111d1ae9bd775
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Mon Apr 19 08:24:23 2021 -0400
```

```
    whoops, removing rogue file

commit f3c235d6a2b48cc46576836f1006f61c55040adc
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Mon Apr 19 08:22:08 2021 -0400

    test import on three levels

commit d0fb9a298903946efa9b1a7561a1ee9939f48926
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Mon Apr 19 00:50:15 2021 -0400

    fail test programs for concat - str and float, int and int, float and float, int and float

commit 6120182d4a327d7bf02c350c922b89c1a39a1c71
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Mon Apr 19 00:41:40 2021 -0400

    test programs and output for concat str tand int, str and float, int and str, float and str

commit 1cd1cec23affa3fd008ac640da8a1333a2c1d30c
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Mon Apr 19 00:41:13 2021 -0400

    support automatic casting when concatenating string with int or float

commit d1db7157631de5b819eaff82d9119310eff13193
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Mon Apr 19 00:20:18 2021 -0400

    failing case for concat - can't concat bool literal

commit dd6586534d427382ab429e7049afa112c95b32ad
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Mon Apr 19 00:11:29 2021 -0400

    add test outputs, now running all tests does not report any outputs missing for diff

commit 03452bbcd2523dfd768e765db1dfaeb90b7b2066
Author: Lauren Pham <lauren.pham@focus.org>
```

```
Date:   Mon Apr 19 00:10:50 2021 -0400

    use string concat now that it has been implemented

commit 5263bd376b8e0519b9aa9a4beb7ee9f6f11a9b18
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Apr 18 23:09:04 2021 -0400

    adding other random test

commit c6a42e45ee0a18effcbb81ec38223fa08c99c318
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Apr 18 23:07:54 2021 -0400

    imports working

commit a439519f645c7985925bda20d3fadf06db1271e2
Author: Carolyn Chen <chen.cec@gmail.com>
Date:   Sun Apr 18 16:16:23 2021 -0400

    Compiling imports with exceptions

    Modified a number of previous test files to exclude imports.

commit 99a8ff0306e1da332f570c4e91ce108309419b89
Merge: acb30ca 69eb1d0
Author: Carolyn Chen <chen.cec@gmail.com>
Date:   Sat Apr 17 14:58:26 2021 -0400

    Merge branch 'main' into semant-imports

commit 69eb1d023c83d696ba5bd9f4f7142f87e099d344
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 17 13:09:18 2021 -0400

    rm comments

commit a7890df30ef4ffd4f326cec6d341366b3a4affc9
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sat Apr 17 13:06:04 2021 -0400
```

```
    fix str concat

commit 8a7b23257a981d7813dadf2ec7e8d3684e30fa38
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 17 12:58:15 2021 -0400

    meg's proposed memcpy changes

commit 4d4a4731ed3243bc428bc93f5d32f5c077052ae4
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 17 12:51:04 2021 -0400

    main function works

commit 879f88609a61f88893fbaaae8e86fa13e6845ae3
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 17 12:35:44 2021 -0400

    buffer+strlen(lhs) for end of lhs

commit 2f570e761161b2060b2650a76583367f8b3f2e46
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 17 12:32:36 2021 -0400

    custom_strcat segfault

commit a9ede5a5f7d6cca9548de61e358ae9b1c8211a08
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 17 12:25:15 2021 -0400

    match SBinop String Concat String with build_call strcat_func

commit 74aa8a195827cfd81774bc2aab3a06718dd21468
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 17 12:03:07 2021 -0400

    syntax error

commit f716a11f8d4da3c16b5e3008aa60b4a8f22d7c3b
```

```
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 17 11:54:31 2021 -0400

    add custom_strcat

commit 2b2238e955f492363a5bf04249ff3fb08f42d5f3
Merge: 8639ca0 6466f3e
Author: Michelle <ml4080@columbia.edu>
Date:   Sat Apr 17 09:37:19 2021 -0400

    Merge branch 'demo' of https://github.com/mmfrenkel/meowlang into main

commit 8639ca0328c471830d57799e809ee6c32d95e820
Author: Michelle <ml4080@columbia.edu>
Date:   Sat Apr 17 09:31:23 2021 -0400

    Moved demo to demos folder

commit 6466f3e8a02619d96d6b952f85f1f2ae2f0ced59
Author: Michelle <ml4080@columbia.edu>
Date:   Sat Apr 17 09:00:58 2021 -0400

    Added arrays to demo

commit e298614967ca34187c328140586a851f9b8adeac
Author: Michelle <ml4080@columbia.edu>
Date:   Sat Apr 17 07:45:20 2021 -0400

    Finished first draft of demo

commit f775cb945f93fa08ad508049c5c1776b3a862521
Author: Michelle <ml4080@columbia.edu>
Date:   Sat Apr 17 01:26:41 2021 -0400

    Added demo.meow

commit 1d91a6c951f8b7e3048ba55d2640ada39bb15e38
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Fri Apr 16 13:45:30 2021 -0400
```

```
    whoops, removing extraneous file

commit 1fe5fde13616545bd17203c1465f020a3b3f63d5
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Wed Apr 14 11:05:59 2021 -0400

    format fix

commit 1b1adf160e5a3e91af3b371620a24d9bcbaf6585
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Wed Apr 14 10:59:58 2021 -0400

    adding float->str, str->float casting

commit 9f1df323f4cccc2741bc7ef7741ea8c3c1b064e8
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Wed Apr 14 10:11:00 2021 -0400

    move c files to another location

commit f0ee73d56fc4d7d2a843dc53f5223844b0b5fc2c
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Wed Apr 14 10:05:03 2021 -0400

    adding string comparison by value

commit 1f575d5f869da4e48a1fa17bfe725e0d4c231705
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Wed Apr 14 08:38:18 2021 -0400

    add casting from int to str, str to int

commit 211eae4b09162beccf5155f7471c1e29ee67ed1f
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Tue Apr 13 22:13:58 2021 -0400

    add type casting int to float, vice versa

commit 2609c6ec155b8327725e88bb2563344d515d0a64
Author: mmfrenkel <megan.frenkel@gmail.com>
```

```
Date:   Tue Apr 13 21:32:10 2021 -0400

    binary, ops semantics

commit 81af0f48590f21fc70cb434ffd7d47a1eb5295e8
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 10 13:46:24 2021 -0400

    program outputs for testing

commit 6b8a5ef88eb8a62b7be8572a4f5d840e1de30c8b
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 10 13:45:44 2021 -0400

    add outputs for testing

commit e7e45617e8e5da54826d3078efab0fa760ba6285
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 10 13:45:12 2021 -0400

    string comp not yet supported so use integer comp in test_if

commit 8825639f61eab5bcd1bbc1f92b3bc161bce42d22
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 10 13:27:13 2021 -0400

    test programs and output match new syntax rules for if statement

commit 11689f0a50b653018b2cef7b30ae29007d4fd14f
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 10 13:23:12 2021 -0400

    stmt -> LBRACE stmt_list RBRACE on code blocks in control flow

commit c32ff63fa5b17acccd45a3585d3f98fac3d72135
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 10 13:03:34 2021 -0400

    tests accomplished in test_programs/fail_for*.meow tests
```

```
commit a7e5bf578a0e93fa3bb5b92c410d03a36947e3be
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Fri Apr 9 23:07:41 2021 -0400

    need to SAssign the SBinop inc/decremented value to index, otherwise the
    ++/-- happens to the value stored in index but is not stored back to index.
    now test_loops program runs and terminates

commit fc545f6042daa987e7dd5644d5ae383fe849e6f8
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Apr 9 18:10:48 2021 -0400

    refactor arrays to remove shift-reduce conflict (#25)

commit acb30cafe0e61fc750c74c62beb8de055b7964fb
Merge: 3ba4e62 25029f4
Author: Carolyn Chen <chen.cec@gmail.com>
Date:   Fri Apr 9 17:29:06 2021 -0400

    Merge branch 'main' into semant-imports

commit 25029f4cbacc3c6517f0a5be2704d99f60c34425
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Thu Apr 8 21:04:19 2021 -0400

    style fixes

commit b1fc359376619ea447b48e2790feba823e11e650
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Thu Apr 8 15:30:01 2021 -0400

    add custom scanf

commit f2e63d5265187ed419e13a81895057e92ca6c0ed
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Thu Apr 8 13:59:56 2021 -0400

    whoops, forgot to add new files

commit 8105c67b896843c8adf508ced23dfd0fbe2a0de1
```

```
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Thu Apr 8 13:59:22 2021 -0400

    update semantics to check return type and void functions

commit decef67af7c4db85c4c12add851001c729b7c6f7
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Thu Apr 8 10:08:55 2021 -0400

    bug fixing and testing for arrays

commit cb7339e2a9c325974814e2035cfbc6a2748e28c9
Merge: 2999269 2175477
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Apr 6 17:47:54 2021 -0400

    Merge branch 'main' into control-flow

commit 299926913d34868cd2b8afeb96d8c79e6e0d0b44
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Apr 6 17:46:58 2021 -0400

    directly pipe output so format matches test

commit 2175477b053b6b06bed317a30d095e257e573a24
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Tue Apr 6 17:46:32 2021 -0400

    arrays working with codegen (#24)

commit fdd597f9a1ad7304e082cc19bfa12acd28cf2340
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Apr 6 17:45:21 2021 -0400

    fix type-directed disambiguation

commit a9c6e29a6414d1298e6e7a6d4b40902a40d62905
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Apr 6 17:43:37 2021 -0400
```

```
    exclude concat since not implemented in codegen yet

commit 74e95cccb9b1e44055218cdbacb88255e3e91d2b
Merge: 2b47437 a6f002b
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Apr 6 17:38:49 2021 -0400

    Merge branch 'main' into control-flow

commit 2b474375b087396f10bf3717648e6b151fbe84dd
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Apr 6 17:38:23 2021 -0400

    comments

commit 59a13806524c5eae04690f2e845433730de9a0d6
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Apr 6 17:31:03 2021 -0400

    turns out unimplemented expr is concat, not the for loop itself. add back multi-line call and comment out concat

commit 248b0cde7e236a287aeda9b3267324b5a58e9bde
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Apr 6 16:48:47 2021 -0400

    test_loops compiles to llvm without multi-statement code blocks inside of loop body.
    gets found expr or function not yet supported if multi-statement code block.

commit a6f002bd746881321000087cf5a468e91120373b
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Tue Apr 6 14:35:49 2021 -0400

    array_semantic checks (#23)

    * semantic checks for arrays

    * semantic check fixes for arrays

commit b4d7266a017a39e3fcc3650667534a3b137413d8
Merge: 01b5728 4c23a68
```

```
Author: Michelle Lin <63328989+Snowfleece@users.noreply.github.com>
Date:   Mon Apr 5 22:57:56 2021 -0400

    Merge branch 'main' into main

commit 4c23a68e9fab0079ae495a92f40319dcdcf277f7
Author: Michelle <ml4080@columbia.edu>
Date:   Mon Apr 5 21:35:19 2021 -0400

    More regression tests for conditionals

commit 01b57281b7a4801fc4428d78894eaeac3c4548fd
Merge: 7c4f14f 479ea6d
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 3 09:32:37 2021 -0400

    Merge branch 'control-flow' into main

commit 479ea6d360d355f4141eab8311656f20d27372b4
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 3 09:30:33 2021 -0400

    testing and output for for loops

commit 7c4f14f54dc8b36ba49810316f00eeee8240abe9
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sat Apr 3 09:27:27 2021 -0400

    update make commands so tests dont run every time

commit fc9d8610fc05c87d2a070e5ac1ea2dd6e8e0a642
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 3 09:14:25 2021 -0400

    all tests produce ast with test -a flag successfully

commit 36f5854d95d5f8535e54e24026fd925d9adc381d
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Apr 3 08:55:22 2021 -0400
```

```
    fail programs cover all cases in based on microc semant and meowlang semant

commit 7c0620b2ac6870d79c14941bc52e444cd9f296f0
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Apr 2 21:57:18 2021 -0400

    Update README.md

commit b43c8f52d3ff4ca2e28957a62b4317cf7560f064
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Apr 2 21:55:09 2021 -0400

    Update README.md

commit 2db8109e404a6ea9d7ed8a697c4bb098630d940c
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Apr 2 21:43:25 2021 -0400

    Ci cd (#21)

    * attempt travis ci/cd

    * Update .travis.yml

    * Update .travis.yml

    * Update .travis.yml

    * Update .travis.yml

    * Update .travis.yml

    * Update .travis.yml

    * Update .travis.yml

    * Update test_all.sh

    * Update run_regression_tests.sh
```

```
commit d81fe38291a0f628534ae6780396cdb65d8a2a73
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Apr 2 17:00:46 2021 -0400

    Update .travis.yml

commit b77ff6734027b24073f80fee5e443beb1a8a3b40
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Apr 2 16:58:02 2021 -0400

    Update .travis.yml

commit c1e77b0647932564496acfe1799c595f3d0ad71f
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Apr 2 16:55:04 2021 -0400

    Update .travis.yml

commit 38d97c59d88f1404beedeb29b36d9d1a86d5891b
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Apr 2 16:51:34 2021 -0400

    Update .travis.yml

commit 2f2093e358c69ed7b2e082fbddbc5173788783f9
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Apr 2 16:47:08 2021 -0400

    Update .travis.yml

commit 432c652ec7e3348ee9821aca94c9bdfcf29c4ae5
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Apr 2 13:43:33 2021 -0400

    attempt travis ci/cd (#20)

commit 068743a381de9fac549abed21fdde07a41c7057a
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Apr 2 11:50:29 2021 -0400
```

```
    Update README.md

commit ebaf6857481c4fe496f062e03f47dcc91ad98a91
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Fri Apr 2 11:47:29 2021 -0400

    fix issue that occurred on merge

commit 8020532be99f9dd4ea3e67070dce0d80b156276c
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Apr 2 11:37:47 2021 -0400

    Class constructor (#19)

    * default and custom constructor working

    * updating test suite

    * refactor/comment for clarity

    * update printing to use printf

    * add additional test

commit 30da81fb55368eb92f8aa1b2756a0cbda62945c8
Author: Michelle Lin <63328989+Snowfleece@users.noreply.github.com>
Date:   Fri Apr 2 11:36:42 2021 -0400

    If/else codegen and testing (#18)

    * Add a couple fail_binop tests

    * Added preliminary semant tests

    * Added preliminary tests

    * Finished creating test files

    * Added if/else to codegen and testing suite
```

```
commit f65ec1ec3e46a2481b73581887f24e7c43bba50a
Author: Michelle <ml4080@columbia.edu>
Date:   Thu Apr 1 16:52:44 2021 -0400

    Added if/else to codegen and testing suite

commit 43cf2fbd9c0870ccbad91a01eb601434973f0fe4
Merge: e96d22c d5f30ed
Author: Michelle Lin <63328989+Snowfleece@users.noreply.github.com>
Date:   Wed Mar 31 11:27:33 2021 -0400

    Merge pull request #8 from mmfrenkel/main

    New Update with LLVM working

commit 3ba4e62d36aad1e24446d11eddd0d044b09d5292
Merge: d482e61 d5f30ed
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Mar 30 18:40:44 2021 -0400

    Merge branch 'main' into semant-imports

commit d482e61bb76e40ae381fc6acd0b0fbd202e22f24
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Mar 30 18:20:40 2021 -0400

    interim

commit d5f30edaf6bffcc3bac83895d1d776d1dcc494f1
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sat Mar 27 18:24:05 2021 -0400

    more style related fixes

commit dd976afa2c3b799a5f5c70c216df7df6c0e29902
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sat Mar 27 18:19:50 2021 -0400

    replace tabs with spaces for consistency
```

```
commit cd9e9148448813a2616f494822c1e675520246b9
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sat Mar 27 18:15:01 2021 -0400

    update comments to be uniform in style

commit 587703a95dfde8ac177a5d0664258101ceca1c5e
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sat Mar 27 18:06:00 2021 -0400

    update comments in codegen

commit 4d7748b7c0b148b6162839c6658f44abc2f7d012
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Sat Mar 27 17:55:50 2021 -0400

    Class semantic transform (#13)

    * starting on class semantic checks

    * adding semantic checks for classes

    * add support for calling method within a class

    * adding lift and shift for class methods

    * move helper function to helper function section

    * class seems to be working!

    * classes seem to be working

    * update comments

commit be1aba55bd40fa3639a31a551306f450520404f6
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Thu Mar 25 16:22:20 2021 -0400

    update comments, to clarify sections of semantics code
```

```
commit ea42edb87f1573fa8fd1db353fb0f3a256a163ca
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Thu Mar 25 16:16:06 2021 -0400

    bug fixing class access and methods (#12)

commit c22e985c2de2f82c4d11bd1454d8b3c585311f22
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Thu Mar 25 15:34:35 2021 -0400

    add new tests for classes; fix constructor issue (#11)

commit e96d22c79f2ac69e679ca2939d0a553365cc26d5
Merge: 7c86fe0 6408171
Author: Michelle Lin <63328989+Snowfleece@users.noreply.github.com>
Date:   Tue Mar 23 20:24:40 2021 -0400

    Merge pull request #7 from mmfrenkel/main

    Include Meg's updates

commit 31073ed87316f40ee267ed6a73542392559cf8fe
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Mar 23 19:39:33 2021 -0400

    maybe this code to scan imports and generate asts works ?? lol

commit 64081719071e589b463f15275fbaf8ff99ad8697
Author: Michelle Lin <63328989+Snowfleece@users.noreply.github.com>
Date:   Tue Mar 23 19:25:28 2021 -0400

    Tests in test_semant (#10)

    * Add a couple fail_binop tests

    * Added preliminary semant tests

    * Added preliminary tests

    * Finished creating test files
```

```
commit 3442a16cd07024550ee4204aaee4a3924f7c3344
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Tue Mar 23 14:05:04 2021 -0400

    making running hello world clearer

commit c150dad9cc29a6416c953423f050b217eb5f2507
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Tue Mar 23 13:58:17 2021 -0400

    Array class semantics (#9)

    * update comments

    * add support for array assignment in semantic checks

commit d6340ff5e8702c729e92c9fc94746cc56454d60a
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Mar 23 12:37:12 2021 -0400

    all of the functions are there, just need to clean them up and make sure they do what we want them to

commit 58fb0477e405a5596e38ec00704033d81b339a4c
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Mar 23 12:14:29 2021 -0400

    add exception ImportNotFound

commit 0f35a9b441dda6b6aad826c6c17840347e1b2b92
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Mar 23 11:50:11 2021 -0400

    restore semant.ml to main - import logic is moving to import.ml

commit 9e2b285e53a331cd7c60645eff4600aa36bd7b19
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Mar 23 11:48:21 2021 -0400

    AST type program_with_imports returned my Import.add
```

```
commit 0fb800913604f6d8fc7baee2a015d852c53a139c
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Mar 23 11:43:10 2021 -0400

    before calling Semant.check, perform import-specific checks on the importing
    program's ast and generate an ast_with_imports that appends functions and classes
    from the imports to the importing program's ast. the ast_with_imports returned
    has type func_decl list * class_decl list

commit 795e1cb0a995abd35bc7f1a0c217e38dfcffeeaf
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Mar 23 11:00:40 2021 -0400

    check for duplicate import file names

commit 50e1519f2c8a71f72abc5aab71e86265632c77cf
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Mar 23 10:57:16 2021 -0400

    add import functions and classes to the importing program's functions and classes; then check_duplicates

commit 82169f77c5a7d53e5bacad701b20053013edb97e
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Tue Mar 23 09:58:04 2021 -0400

    add array assignment as type of supported statement

commit d5ae6be694b9bfa088db915816af5d6311595628
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Tue Mar 23 09:36:56 2021 -0400

    fix file name convention for tests

commit a9190470ea80c4911c163fccba32c3e8a5de002a
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Tue Mar 23 09:34:48 2021 -0400

    add tests for printing bool and floats
```

```
commit 7c86fe049217add0a3b3cf4de85ba600074c59bb
Author: Michelle <ml4080@columbia.edu>
Date:   Mon Mar 22 14:40:15 2021 -0400

    Finished creating test files

commit 5bdd3960a914550d4ddf17559bed7d8cce3f3274
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Sun Mar 21 18:38:36 2021 -0400

    Custom function (#8)

    * custom functions working, simple math

    * checking division

    * update comment

    * update test script

commit e95f49e28a54c64cb1aa7f1e597d85fd66e6273c
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Sun Mar 21 12:14:06 2021 -0400

    starting on binop llvm (#7)

commit a0e1101a639d319ea3abb2b0f10e57d9d30b6e64
Merge: 41ba077 474f4cb
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Mar 21 11:38:04 2021 -0400

    Merge branch 'main' of github.com:mmfrenkel/meowlang into main

commit 41ba07710a1b2779be4ca9b54905c93ca1470707
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Mar 21 11:37:57 2021 -0400

    adding forgotten output files

commit 474f4cb97689787a115eb620c1b8376b38bc85e3
```

```
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Sun Mar 21 11:36:48 2021 -0400

    Update README.md

commit a90f3376c92d2e3798a35611235a5aca460103ee
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Sun Mar 21 11:34:45 2021 -0400

    Update README.md

commit 26a03fe61fe3865013c6dfd2d6edc7e2fedbd291
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Sun Mar 21 11:34:05 2021 -0400

    Update README.md

commit ea4f962248cddfdcf19ae753258ea3fdb93df0f2
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Sun Mar 21 11:28:54 2021 -0400

    Test assignment (#6)

    * adding local variable creation; Meow not working because expects string

    * printing strings and ints work!

    * updating tests for semantic/full pipeline tests

    * add additional tests

commit c17d31b7d60d6a40ec33582c8d9ebb0894e0a93b
Merge: a815e2b 12f4db0
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sat Mar 20 09:36:52 2021 -0400

    Merge branch 'main' of github.com:mmfrenkel/meowlang into main

commit a815e2b124114651a128882ef48b8bfaf0a458d9
Author: mmfrenkel <megan.frenkel@gmail.com>
```

```
Date:   Sat Mar 20 09:36:39 2021 -0400

    update build to use microc _tags approach

commit 12f4db0fa345705497aa803a1162160e19b16bf9
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Fri Mar 19 21:17:32 2021 -0400

    checking that assignment is assignment to index variable is not
    actually required, even if otherwise would be nonsensical.

commit 23d15d36cda59596d0f9ef59253b3745e62c82dd
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Fri Mar 19 20:50:46 2021 -0400

    must declare index

commit 0afd33890917b3b06592507904341c76a41a94db
Author: Michelle <ml4080@columbia.edu>
Date:   Fri Mar 19 15:17:36 2021 -0400

    Added preliminary tests

commit 506c9b325af6c371195ea81bd98b5d8623d169b0
Author: Michelle <ml4080@columbia.edu>
Date:   Fri Mar 19 15:15:37 2021 -0400

    Added preliminary semant tests

commit 8b185595800048a7152874265af73fb83179d87f
Merge: 83d125c 16f6465
Author: Michelle <ml4080@columbia.edu>
Date:   Fri Mar 19 11:32:32 2021 -0400

    Merge branch 'main' of https://github.com/Snowfleece/meowlang into main

commit 16f6465f6664bbcee1b59d11c9a4c17b0768d5c8
Merge: be8206a fb2395d
Author: Michelle Lin <63328989+Snowfleece@users.noreply.github.com>
Date:   Fri Mar 19 11:32:27 2021 -0400
```

```
    Merge pull request #6 from mmfrenkel/main

    Hello world (#5)

commit 83d125ced5eaf96feb4b332a02f4fa2f61220200
Author: Michelle <ml4080@columbia.edu>
Date:   Fri Mar 19 11:31:53 2021 -0400

    Add a couple fail_binop tests

commit fb2395db5791b9ce6bcac9292f191dce075bde08
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Thu Mar 18 14:25:16 2021 -0400

    Hello world (#5)

    * semantic check of main function working

    * hello world working!

    * update comment

    * reformatting to make more readible

commit be8206a9a9c673dd69c168ec639e7f016e3f78fb
Merge: e032e2b 8257799
Author: Michelle Lin <63328989+Snowfleece@users.noreply.github.com>
Date:   Tue Mar 16 10:39:06 2021 -0400

    Merge pull request #5 from mmfrenkel/main

    New Updates

commit 82577996b2fac62e33a3d15425d866abdb8dd0f7
Merge: cda3239 bff7fec
Author: Will-Penney <79379418+Will-Penney@users.noreply.github.com>
Date:   Sun Mar 14 17:06:25 2021 -0400

    Merge pull request #4 from mmfrenkel/array_access
```

```
    Array access

commit bff7feca3ab77bfc6fbfe37af4eddf922c689b2d
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Sun Mar 14 17:04:46 2021 -0400

    Update test_array_access.meow

commit 4fbc7b37d0857885ffb8ef0523aea8dab485c19a
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Mar 14 17:02:21 2021 -0400

    test for array access working!!

commit e9cd7190d0e5afcd0749c27fdf5f0e3432cc13c9
Merge: df86bee f8d0680
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Mar 14 16:49:43 2021 -0400

    Merge branch 'array_access' of github.com:mmfrenkel/meowlang into array_access

commit f8d0680fac62968d53987b098a68f25637e3274f
Author: WilliamJ-P <71245276+WilliamJ-P@users.noreply.github.com>
Date:   Sun Mar 14 16:49:29 2021 -0400

    adding missing parser and scanner changes

commit df86beebe2e4198c5584b79bc29fbce6c7017a97
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Mar 14 16:43:57 2021 -0400

    working array access

commit d9d63b91f9f165ee45a51e5d51f6cad0fd38e50a
Author: WilliamJ-P <71245276+WilliamJ-P@users.noreply.github.com>
Date:   Sun Mar 14 16:29:50 2021 -0400

    adding array access
```

```
commit cda3239be9aca15fa85fab62551b468963deafc6
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Mar 14 15:20:55 2021 -0400

    ok hopefully last bug fix for a little while...

commit 3aa1369577ef90ae18d7d22170eda876f6271a07
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Mar 14 15:19:01 2021 -0400

    simplifying call to Check function, taking global vars into account

commit fcc4e94edce5d726ba9dbfe81a7c8e90916b963c
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Mar 14 15:08:29 2021 -0400

    get global log working correctly

commit 0f2207195b31e8817fe5fd6fbea222ac61328e2e
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Sun Mar 14 14:46:16 2021 -0400

    Update run_regression_tests.sh

commit ef61084d3fef8054462acdbef7db3ed88bb542b7
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Mar 14 14:44:04 2021 -0400

    remove unnecessary global logs we dont want to keep around

commit d26077de02c61e57d92a5c9a140fc70fb696f8a4
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Mar 14 14:42:15 2021 -0400

    Slight refactor of testing setup; fix support for failing tests; add new syntax checks

commit e032e2bc3c3f29336345a2d395f0b046cb1acd11
Author: Michelle <ml4080@columbia.edu>
Date:   Sun Mar 14 12:40:17 2021 -0400
```

```
    Added test bash

commit 4be443b95a7709ea21502dfc427842d888f91f6b
Author: Michelle <ml4080@columbia.edu>
Date:   Sun Mar 14 11:02:29 2021 -0400

    Fixed loop check and it compiles now

commit 4a22054898ce284f2bd37b9e08da3eb7dae9747e
Merge: 58e1581 3b6fd84
Author: Michelle Lin <63328989+Snowfleece@users.noreply.github.com>
Date:   Sun Mar 14 10:17:30 2021 -0400

    Merge pull request #4 from mmfrenkel/main

    Loop Semantic Check Update

commit 3b6fd84e8366c8ef94ed6bc4255632114e329308
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Mar 13 21:00:15 2021 -0500

    semant_stmt 'finished' except for Dealloc and ClassAssign to be completed with
    semantics checking on classes. some questions mostly on check_index_assignment

commit 5427f807656d67b820a82645fdf3ee4a72f723a0
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Mar 13 20:53:28 2021 -0500

    added exception ControlFlowIllegalArgument of string ... this exception
    is thrown by all control flow checks with varying messages, not sure if I need
    to break it down into types of illegal argument exceptions. any exception can use
    expr_type_mismatch and op_type_mismatch message templates

commit b3341023509809ff9847379e99556054d049783b
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Mar 12 10:22:22 2021 -0500

    Update parser.mly

commit 0ad5c7ada1568a66298b73bccf5a7751f26d463f
```

```
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Mar 12 10:21:51 2021 -0500

    attending to my ocd

commit 425e9f5ff1f9dd9662f0f8f6d01fe3e8bb520b0c
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Mar 12 09:12:58 2021 -0500

    Update pretty.ml

commit 58e1581367eda425805028fce6716b001356281d
Merge: fa25478 fbf33ef
Author: Michelle Lin <63328989+Snowfleece@users.noreply.github.com>
Date:   Thu Mar 11 17:49:50 2021 -0500

    Merge pull request #3 from mmfrenkel/main

    Semantics Update

commit fbf33efb26094d651b756cb517251298fbf76f5e
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Thu Mar 11 17:42:29 2021 -0500

    add dynamically created obj and arr to symbol table as expr is checked

commit b1719707eff2d66a77dcce9e92308c925e83ab75
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Thu Mar 11 17:13:20 2021 -0500

    add new check for integer literal size of array vs contents

commit fdf0192c186fa677f3ac98bd0128601b303ab712
Merge: e293ae1 7d9a254
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Thu Mar 11 16:58:15 2021 -0500

    Merge branch 'main' of github.com:mmfrenkel/meowlang into main

commit e293ae1fc04924a9dfef44b1e642fa1735a742eb
```

```
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Thu Mar 11 16:57:59 2021 -0500

    update semantic checking to include array types

commit fa25478769d9df820e6d0bda1ccd09881f3dcd75
Merge: 0e09b4b 7d9a254
Author: Michelle Lin <63328989+Snowfleece@users.noreply.github.com>
Date:   Thu Mar 11 09:34:32 2021 -0500

    Merge pull request #2 from mmfrenkel/main

    New Updates

commit 7d9a254f266d9580b1ed1c48cb66c02082362599
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Wed Mar 10 18:36:25 2021 -0500

    Update meowlang.ml

commit 94be03562ca3fab094dbc1f0b8aaf3fab3dafa3e
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Wed Mar 10 18:33:06 2021 -0500

    fixing semantic check

commit 99813c69fed05bc859e549ed552d3b28098a177c
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Wed Mar 10 17:45:55 2021 -0500

    semantic checks, round one...

commit 60d0d508ce7fbd898acfdc8da91006c6fe5e7908
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Fri Mar 5 17:16:04 2021 -0500

    Semantics (#2)

    * getting the ordering of functions and classes to not matter!
```

```
    * adding semantic checking; just a start

commit 540607d0f04d8b951fbb13dd0a15836e743b2ef2
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Fri Feb 26 17:11:51 2021 -0500

    adding in calling methods -- we had forgotten this

commit 9511c72b970e5e0caba0d5dc37ab9c1680f810c5
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Wed Feb 24 19:24:06 2021 -0500

    adding document submissions

commit cde728083acd82e7e2816c0b26bf553a63befa19
Merge: a6be154 f061df6
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Wed Feb 24 18:49:42 2021 -0500

    Merge pull request #1 from mmfrenkel/meg_working

    Meg working

commit f061df6514fb0fb6154ae81baee6f35e503c3802
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Wed Feb 24 18:42:45 2021 -0500

    debugging for different class behaviors

commit bd185b2f84a1effa2ccc6fd27d2e280b75ee8b60
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Wed Feb 24 18:30:34 2021 -0500

    classes appear to be compiling

commit a6be1540d48649d8ef9e2eabf6ceb6cf3e122fdb
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Feb 23 22:15:06 2021 -0500

    readability
```

```
commit b41a1e7d4a54a19a79ca42cb375fa1e604d5c4af
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Feb 23 22:14:31 2021 -0500

    readability

commit 47bfb5617fd63e9c65b5fc002432ddee1774bb60
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Feb 23 21:48:14 2021 -0500

    insturctions for running all tests

commit 79465179f68e873a4f828cbee541680a9ee8f020
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Feb 23 21:44:58 2021 -0500

    rename for file tree; create bash script to run all tests

commit 0cbea7621776accecef3726a745e0e48f90e6bc3
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Feb 23 21:30:12 2021 -0500

    formatting

commit b7631363835683ba6939699b2d71610057446301
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Feb 23 21:06:36 2021 -0500

    improved test

commit e4169293be6cc4c1d8115b6c4a05cef7c5ec4035
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Feb 23 20:54:39 2021 -0500

    mitigate shift/reduce with 'expr_opt expr' since COMMA cannot be tacked onto
    expr_opt by putting the first expr before UPPIN/NERFIN

commit 02a6f02d0ff865b91975cad42120ee91f640aae7
Author: Lauren Pham <lauren.pham@focus.org>
```

```
Date:   Tue Feb 23 20:40:17 2021 -0500

    experimenting with expr_opt COMMA expr. must have COMMA between otherwise shift reduce; but
    parse error with test cases with expr_opt because technically COMMA should be part of the expr_opt,
    as in the case index AN index IZ 15 AN <stmt>. to omit expr, we really want to emit the expr
    and the extra COMMA.

commit 37d98ee785cf3d4935019be5f7c91be2a3c639f5
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Feb 23 20:24:30 2021 -0500

    add AN / COMMA to separate expr in loop declaration

commit de6ec70fa153bd6b8122a907a667be2887f1094b
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Feb 23 19:46:55 2021 -0500

    remove expr_opt

commit bec07448e685213b1b1bfa338441ae2158e33543
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Feb 23 19:39:56 2021 -0500

    remove YR in 'UPPIN YR' / 'NERFIN YR' rules - 'YR' adds no info and is
    already part of the FOR token - 'IM IN YR LOOP'

commit 92131d3788e4f95f79f65817b720def477b7dad4
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Tue Feb 23 19:33:22 2021 -0500

    clearer testing - use CAT

commit 0e09b4bdca2095771bc3703c4e6154dd1beadc2c
Merge: db54aaa fbaa4d8
Author: Michelle Lin <63328989+Snowfleece@users.noreply.github.com>
Date:   Sun Feb 21 17:11:41 2021 -0500

    Merge pull request #1 from mmfrenkel/main

    New updates
```

```
commit fbaa4d832fa3647d94aaa8ab2683c88df8466755
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 22:53:58 2021 -0500

    test using code blocks

commit 2730978e4f8c53c1a31ae58e8ceca2f4dc37642a
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 22:37:50 2021 -0500

    test ommitted else, single statements, and code blocks

commit 7e585f4c194784dcd089e274315fcf5ef920adfb
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 22:18:40 2021 -0500

    don't need for this test

commit 3108233645efd345b7d0d92dbd37054ef02a49d9
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 22:03:21 2021 -0500

    testing both index types

commit 835acebae9b252f5316f63af8b8cada259de2bc0
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 22:01:32 2021 -0500

    loop can use index instantiated outside of the loop or instantiate it in the for ()

commit 5ba482d13ffcbad5e52a83d6ddb4757cf45b8f36
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 22:00:50 2021 -0500

    add expr_opt and Noexpr

commit 74ab8044d6f22fb4b5331fdb7d6f494321e9fcfd
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 21:52:16 2021 -0500
```

```
    move ID ASSIGN expr back to an expr; can become a stmt by matting with expr SEMI pattern of stmt

commit 12d5dfe7c894ae6372d0aa317c82575842799575
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 21:16:08 2021 -0500

    add YA RLY == then, lol I just missed it the first time

commit 5c980a4614bf0f89c847003424b8b4ec371f6fe0
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 21:08:46 2021 -0500

    for code blocks, we need {}. instead of specifying new ones, use HAI and KBYE

commit 39559f3820a5754043d9c5451b72c137af26bdaf
Author: Carolyn <cec2192@barnard.edu>
Date:   Sat Feb 20 15:58:58 2021 -0500

    Create mouse_class.meow

    Adding mouse class test program

commit b87db2f34306a6d1d8481a93df723cc4dbd982b1
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 14:46:28 2021 -0500

    better test and print

commit 16064ecee1e08770f0acf40e74bfd77c929a097a
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 14:08:08 2021 -0500

    added increment and decrement options

commit b29f013d442b2566f69c0fe808a2d4ff84e3a8b4
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 13:43:07 2021 -0500

    barest of for loops works
```

```
commit 459cf151134b33728e166642161ee0f536240ce5
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 01:06:52 2021 -0500

    add test if and if-else

commit da35133ac3c56069f3bc303edf369df25f16e3ea
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 01:06:43 2021 -0500

    fixed pretty print for if / if-else

commit 3bd399e8e528fbd2853a16f4ba256251ea9ca7e3
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 01:01:12 2021 -0500

    eliminate shift reduce by making NOELSE and ELSE %nonassoc

commit 079e842c4a3e9c5684965460d218383625932b5f
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 00:53:49 2021 -0500

    if(expr) stmt else stmt works

commit 89d96a3bc649bc62f06663b5cc518cdfa013d068
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 00:29:22 2021 -0500

    test loop

commit 9bb16bcbc2be9e9eef174da385e7075406ca42b2
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Sat Feb 20 00:24:20 2021 -0500

    if(expr) stmt works - need brackets around stmt and else

commit a2f2337a1be31d806275fa3073f577e90bdd0919
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Fri Feb 19 23:15:40 2021 -0500
```

```
    rename file, add else if to test

commit 8b78671dd585c08d0ef5b384c5638c4396771375
Author: Lauren Pham <lauren.pham@focus.org>
Date:   Fri Feb 19 23:10:17 2021 -0500

    write test with conditional

commit ac0baf2fb863e60f2ff6132983279402934ca801
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Tue Feb 16 16:39:10 2021 -0500

    array implementation working!!!!

commit 938c1186ed77c2ae4595182c29abe827a52c6f2c
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Tue Feb 16 13:50:36 2021 -0500

    update parser sections and update function declaration rules to be more succinct

commit f70787558af81af72d60ce2351644d2b18114975
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Tue Feb 16 13:03:59 2021 -0500

    removing shift reduce errors on desired function call syntax

commit 997462df23bfd09867c88474ec0a21586cd2f8b9
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Tue Feb 16 09:26:15 2021 -0500

    fixing shift/reduce conflicts

commit 4112dff0eebaaa765fd3ea47a598c589f48248e1
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Mon Feb 15 11:34:52 2021 -0500

    update comments for clarity

commit a756c99c32a1080e3910b7f39b22e4a3a61f12d5
```

```
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Mon Feb 15 11:31:07 2021 -0500

    forgot to add pretty printing file

commit 1b48d42ce9475fa1490f7d6cf9e91d821859b772
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Mon Feb 15 11:30:43 2021 -0500

    moving pretty printing to its own module

commit 6186290784772b486e62e4cb952500b5b637f78c
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Mon Feb 15 11:23:51 2021 -0500

    Add helper makefile so that you can build from project root

commit 3f690ef8001bc8e2941f7f0c2fd6a2435dfc7f07
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Mon Feb 15 11:15:53 2021 -0500

    formatting, plus removing old makefile content

commit 34b9626eb4b5c10b03d9a73f0014c9dd6ccc189d
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Mon Feb 15 11:14:03 2021 -0500

    parser can parse program with functions and print as C program

commit b813db6e0053b4806e527b75b1ce115c3d50dcd2
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Feb 14 16:15:40 2021 -0500

    one function parsing pattern works!

commit 0b9b7f74897bf43800899649499cd6d56c12b8d6
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Feb 14 16:03:08 2021 -0500

    something is compiling!
```

```
commit 306ac49020749ba815af4f9748f18b9c4b5d089b
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Sun Feb 14 15:00:06 2021 -0500

    first pass at scanner/parser

commit 6cd6829e46b3a66674f037cd6a862429df6c1c32
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Thu Feb 11 11:08:43 2021 -0500

    adding src folder

commit 92fda73535ac16e68c2fd67c5938253bca5e6e4e
Author: mmfrenkel <megan.frenkel@gmail.com>
Date:   Tue Feb 9 19:20:41 2021 -0500

    Adding files sourced from Hw1 as starter files

commit db54aaa4127778b018a26027905779ec23ca2eaa
Author: Megan Frenkel <38475288+mmfrenkel@users.noreply.github.com>
Date:   Tue Feb 9 18:48:09 2021 -0500

    Initial commit
```