

Music-Mike

Harvey Wu, Kaitlin Pet, Lakshmi Bodapati, Husam Abdul-Kafi

May 11, 2017

Contents

1	Introduction	5
1.1	Design Ethos	6
2	Language Tutorial	7
3	Language Reference Manual	9
3.1	Types, Operators, and Expressions	9
3.1.1	Basic Types	9
3.2	Expressions	11
3.3	Variables and Assignment	13
3.4	Operators	14
3.4.1	Arithmetic Operators	14
3.4.2	Logical and Comparison Operators	15
3.4.3	Pitch Operators	15
3.4.4	List Operators	15
3.5	Control Flow	16
3.5.1	Statements	16

3.5.2	Expressions	16
3.5.3	If-Then-Else	16
3.5.4	Block	16
3.6	Functions and Program Structure	17
3.6.1	Functions	17
3.6.2	Built-in Functions	17
3.7	Comments	18
3.8	Scoping	18
3.9	Built in Functions	18
3.9.1	Printint	18
3.9.2	Printstr	18
3.9.3	Printffloat	18
3.9.4	Printlist	19
3.9.5	Printrlist	19
3.9.6	Synth	19
3.9.7	Merge	19
3.9.8	Make_midi	19
4	Project Plan	20
4.1	Process	20
4.1.1	Planning	20
4.1.2	Specifications	21
4.1.3	Development	21

4.1.4	Testing	22
4.2	Programming Style Guide	22
4.3	Timeline	23
4.3.1	Roles and Responsibilities	24
4.3.2	Software Development Environment	24
4.3.3	Project Log	25
5	Architectural Design	46
5.1	Block Diagram	46
5.2	Interfaces	48
5.2.1	Scanner (Lakshmi, Husam, Harvey)	48
5.2.2	Parser (Kaitlin, Husam, Harvey, Lakshmi)	48
5.2.3	Type Inference (Harvey)	49
5.2.4	Semant (Lakshmi, Husam, Harvey)	49
5.2.5	The Code Generator (Husam, Kaitlin, Lakshmi)	50
6	Test Plan	51
6.1	Example tests	51
6.2	Test Suite	126
6.3	Test Suites	138
6.4	Test Automation	138
7	Lessons Learned	139
7.1	Team Reflections	139

7.1.1	Harvey Wu	139
7.1.2	Kaitlin Pet	140
7.1.3	Lakshmi Bodapati	140
7.1.4	Husam Abdul-Kafi	141
8	Appendix	143

1. Introduction

Western music is usually notated on a five-line staff, on which *notes* are given a *duration* based on symbol type, and *pitch* based on location in the staff. Composers can use proprietary software such as Sibelius or Finale to manipulate a virtual five-line staff through mouse clicks or keyboard gestures. Fans of computer music might instead use music synthesis libraries to programmatically create music in languages such as C++, but such libraries can be unintuitive for musicians unfamiliar with signals and waves.

We propose Music-mike, a compiled, strongly typed, programming language to give users an alternative option in music creation. Music-mike is designed for users to create music based on varied manipulations of short patterns. We owe this idea to Note Hashtag, a previous project completed in COMS W4115. However, unlike Note Hashtag, Music-mike is *modal* rather than *key-based*. Furthermore, lists - treated as the fundamental building block of music - are manipulated with special list operators (syntactic sugar) which create an intuitive interface based on traditional staff notation.

1.1 Design Ethos

The most basic unit in music is a *note*, which can be decomposed into pitch and duration. A simple melody can thus be described as two lists: one list of pitches and another of durations. A *chord* is a collection of notes played simultaneously.

A mapping of pitches is defined as a *mode*. All modes are subsets of the chromatic scale, which contains all twelve pitch classes used in Western music. Most music constrains the pitches of its notes to a small set of familiar modes, such as the major and minor scales. The sound of a *chord* is very much dependent on the *mode* that its notes come from.

Music-mike is based on the following observations regarding Western music: one, that Western music is fundamentally *chordal* and *modal*. Two, that Western music is repetitive and manipulative: simple building blocks of music are modified, then repeated multiple times in a piece. Finally, and most importantly, that these simple building blocks can be described using lists and altered using a functional paradigm.

2. Language Tutorial

Basics

Let's try defining a variable first:

```
x = 5;
```

Now, let's write a function:

```
def AddFive a = a + 5;
```

Notice that we don't need to add any type annotations. Type inference handles everything for us. We can also write polymorphic functions, like the identity function:

```
def Identity x = x; y = Identity("Who am I"); Printstr(y);
```

Function calls require parentheses around arguments, but no commas to separate. For example:

```
def Add j k = j + k; sum = Add(4 5);
```

Note that arithmetic operators do not overload: we cannot pass in an floats to our Add function:

```
/* This fails! */ wrong = Add(4.5 3);
```

A Musical Hello World

First we need to define a mode:

```
/* A major scale */ major = [1 3 5 6 8 10 12 13];
```

A rhythm list is defined with the r:[] constructor:

```
r1 = r:[s s s s s s e s s s s s s e];
```

A pitch list is defined with the p:[] constructor:

```
p1 = p:[1 1 5 5 6 6 5 4 4 3 3 2 2 1];
```

Now, we can generate a music string with the Synth function, which takes a pitch list, rhythm list, mode, start node, and channel:

```
startnote = 50;  
  
s = Synth(p1 r1 major startnote 1); /* Make_midi outputs  
a midi file */ Make_midi(s "twinkle.midi");
```

And that is twinkle twinkle little star!

3. Language Reference Manual

3.1 Types, Operators, and Expressions

All types are immutable in Music-Mike.

3.1.1 Basic Types

- **Unit (unit)**

The only value that unit can take is () .

- **Boolean (bool)**

Takes two values: true or false.

- **Integer (int)**

A 32-bit signed integer.

- **Float (float)**

A 64-bit floating point number - follows the specifications of IEEE 754.

Must contain a decimal point and either an integer or fractional component. The missing component is treated as a zero.

In the context of a rhythm list, floats can also be one of 6 characters that get scanned in as floats.

char	float
q	1.0
w	4.0
h	2.0
t	0.33
e	0.5
s	0.25

Examples: 5. 6.43 3.1415 .42 q e

- **String (string)**

A simple string enclosed by double quotations not spanning multiple lines.

Examples: "hello" "music mike looks good AND sounds good"

- **Integer List**

A list of 32-bit signed integers surrounded by square brackets [], delimited by spaces.

Examples: [1 2 3 4] [57 0 65]

- **Rhythm List**

A list of 64-bit floating numbers and float characters (q, w, h, t, e, s) surrounded to the left by r:[and to the right by] and de-

limited by spaces. A Rhythm List is used to denote the length of each pitch/chord at each corresponding position in a Chord List.

Examples: `r: [0.5 .6 1.9 37.0]` `r: [q w s t]` `r: [.75 e s .09]`

- Chord List

A list of chords. A chord cannot appear anywhere else. A chord is a list of one or more pitches delimited by a vertical bar |. A pitch is an integer pre-operated by zero or more `\wedge` and `\vee`s or post-operated by zero or more `\flat` or `\sharp`. The chord list is delimited by spaces and surrounded to the right by `p: [` and to the left by `]`. A `\wedge` semantically represents an Octave Up. A `\vee` semantically represents an Octave Down. A `\flat` semantically represents a Flat. A `\sharp` semantically represents a Sharp.

Examples:

`p: [1 2 3]`

`p: [1|3|5 5 6 7]`

`p: [\wedge1|\wedge4|\wedge2\# 5 \vee6|7|8\#\#\flat\#\#]`

3.2 Expressions

All expressions have return values in Music-Mike. An expression could be:

1. An integer, float or boolean literal constant:

c

2. A string constant enclosed by quotation marks

“*string*”

3. An arithmetic operation:

$e_1 \text{ op } e_2$

4. An *if – then – else* statement:

if e_1 then e_2 else e_3

5. Variable declarations and assignment:

$\mathbf{x} = e_1$

6. Variables:

\mathbf{x}

7. A block of expressions. The last expression (e_n) is the value of the block:

$\{e_1; e_2; \dots e_{n-1}; e_n\}$

8. A Function Declarations. The name of the function starts with a Capital letter. The number of arguments n is greater than or equal to 0.

def Fname $arg_1 \dots arg_n = e$

9. Function call. The arguments are delimited by whitespace.

Fname ($e_1 \dots e_n$)

10. A white-space separated list of integers:

[$int_1 \dots int_n$]

11. A white-space separated list of floats and/or float characters:

r : [$float_1 \dots float_n$]

12. A white-space separated list of chords

p : [$chord_1 \dots chord_n$]

13. A concatenation of two list expressions:

$e_1 @ e_2$

14. Subsetting a list:

$e.[int]$

3.3 Variables and Assignment

Variable Identifiers in Music-Mike are strings that can be expressed using the regular expression:

['a' 'c'-'u' 'w'-'z'] | ['a' 'c'-'u' 'w'-'z']
['a'-'z' 'A'-'Z' '0'-'9' '_'] *

Essentially, Variable Identifiers cannot start with a b or v.

Function Identifiers in Music-Mike are strings that can be expressed using the regular expression:

```
[ 'A'-'Z' ] | [ 'A'-'Z' ] [ 'a'-'z' 'A'-'Z' '0'-'9' ' '_* ]
```

Essentially, Function Identifiers have to start with an Uppercase letter.

There are no type annotations in Music-mike due to use of the Hindley-Milner type system. We can assign a value to a variable using the following syntax:

```
identifier = expr
```

and type-inference will figure out the type. Note that the assignment operator is non-associative.

3.4 Operators

3.4.1 Arithmetic Operators

Binary arithmetic operators are strongly-typed; both operands must be of the same type, and use the correct operator for their type.

Binary integer operators in order of precedence: `*` `/` `+` `-` `.`

Binary float operators in order of precedence: `*.` `/.` `+. .` `-.`

3.4.2 Logical and Comparison Operators

Comparison operators support integers and floats; both operands must be of the same type:

< > == !=

The following boolean comparison operators are listed in order of precedence:

== != && ||

3.4.3 Pitch Operators

All pitch operators are unary. The postfix operators ‘#’ ‘b’ raise or lower pitch by a half step. The prefix operators ‘^’ and ‘v’ increment/decrement the octave of pitch by one.

3.4.4 List Operators

- Concatenation- One list followed by another of same type connected by an ‘@’ symbol.
- Index- gets value of element of list using . [] operator

```
def Get_second x = x.[1]
```

3.5 Control Flow

3.5.1 Statements

A statement is an expression followed by a semicolon.

3.5.2 Expressions

Expressions always have a return value. A constant expression returns its literal value, a variable expression returns the value in that variable. All of the list expressions return the list. Each of the expressions in a sequence of expressions has a value. Function declarations return the function itself as a value.

3.5.3 If-Then-Else

If statements are structured as `if boolean-condition then expr else expr`. If-Then-Else statements are themselves expressions and thus have return types. The expressions after `then` and `else` must have the same type.

```
fun iszero x = if x == 0 then true else false
```

3.5.4 Block

Blocks of code consist of semicolon delimited expressions and are enclosed by brackets ended by a semicolon.

3.6 Functions and Program Structure

3.6.1 Functions

Functions can be defined using the keyword `def`:

```
def Name arg1 ...argN = expr
```

Note that the first character of a function's name has to be capitalized. Here's an example:

```
def Plusfive x = x + 5;
```

We can also define our function to return more complex expressions:

```
fun Iszero x =
  if x == 0 then true
  else false
```

There is no function overloading in Music Mike. Declaring a different function of the same name is not legal. `Iszero 5.0` is not valid.

Functions are almost first-class citizens: they can be passed in as arguments to other functions and returned by functions, but user-defined functions cannot be nested. Thus we avoid the funarg problem and handling closure.

3.6.2 Built-in Functions

```
Printint Printstr Printfloat Printlist Printrlist
Synth Make_midi Merge
```

3.7 Comments

Comments are enclosed by `\ * * \`. There is no special single-line comment syntax, and nested comments are not supported.

3.8 Scoping

Once a variable has been defined, it cannot be redefined. All variables are stored in a global symbol table. In this sense Music-Mike is dynamically scoped.

3.9 Built in Functions

3.9.1 Printint

Given an integer value, prints it to standard output.

3.9.2 Printstr

Given a string value, prints it to standard output.

3.9.3 Printfloat

Given a float value, prints it to standard output.

3.9.4 Printlist

Given an integer list, prints it to standard output.

3.9.5 Prinrlist

Given a rhythm list, prints it to standard output.

3.9.6 Synth

Takes a chord list, rhythm list, integer list (mode), integer starting note, and integer channel number, returns a CFugue string representation.

3.9.7 Merge

Takes two strings and returns them concatenated together.

3.9.8 Make_midi

Takes a CFugue string representation and a filename, generates the Midi file represented by that string and saves it.

4. Project Plan

4.1 Process

4.1.1 Planning

Our team met regularly twice a week on Wednesdays to meet with our TA Jacob Graff and on Sundays to work together as a team, debrief and set the course for the rest of the week. We used our Wednesday meetings as an opportunity to track and gauge our progress and also ask questions about difficult problems we came across during the previous week. We also talked about goals and milestones for the next Wednesday meeting and talked about any potential problems related to the difficulties of the goals we defined but also about any foreseeable road blocks related to tests, projects, other classes etc. that might hinder our progress. We used these meetings to make sure our project was progressing but also to shift our timeline to account for future roadblocks and delays.

4.1.2 Specifications

We spent the first three weeks deciding the specifications of our language. We all met near a piano either in the basement of the dorms or in Lerner and went over intuitive ways for musicians to express music. Once we chose how we wanted to abstract notes, pitches, chords, tempos and more, we started talking about how to structure our language. We initially chose to do a functional language modelled after OCaml, but as we progressed, we realized that for the use cases we were targeting a fully functional language wouldn't give us the kind of ease of use and usage we'd like. Our first concrete specifications were the abstractions and then we decided on syntax. Despite having a very concrete definition of specifications early on, we still changed specifications as we worked on our language when it was necessary to be able to finish within our timeline.

4.1.3 Development

Our team used github issues to define specifications and tasks that needed to be implemented or completed. We used github to help with organizing our development. Each of used a separate branch to develop the feature that we were working on and then submitted a pull request to the main branch once we thought it was ready. Then, another member of the team would review that request and merge the request. This ensured that all the code we pushed had been code reviewed and helped us maintain the quality of our mainline

code.

4.1.4 Testing

We developed a test suite that tested individual components of the compiler. Every time one of us was working on a small component, we first wrote a test for how that component was supposed to work once it was finished. When writing In this regard, we used some principles of Test-First Programming to make sure we were preserving the functionality of the older features but also ensuring functionality of the new ones. While some of these tests were forward looking and failed early-on, the error messages told the developer whether we were making progress towards making these larger full stack tests work or if it was failing in whatever module the developer was working on.

4.2 Programming Style Guide

- Landin’s pseudo law: Treat the indentation of your programs as if it determines the meaning of your programs. Keep indentation consistent with that of the MicroC code.
- Keep lines shorter than 80 characters.
- A function should always fit within one screenful (of about 70 lines), or in exceptional cases two, at the very most three. To go beyond this is unreasonable.

Justification: When a function goes beyond one screenful, it's time to divide it into subproblems and handle them independently. Beyond a screenful, one gets lost in the code. The indentation is not readable and is difficult to keep correct.

- The change in indentation between successive lines of the program is 2 spaces.
- Using the tab character (ASCII character 9) is absolutely not recommended. Change your .vimrc if you have tabs.
- Use underscores instead of Camel case

4.3 Timeline

Date	Milestone
Jan 29th	First commit to repository
Feb 8th	Project Proposal and White Paper Completed and Submitted
Feb 22	Language Reference Manual Completed and Submitted
Mar 21	Basic Scanner Complete
Mar 26	Basic AST and Parser Complete
Mar 29	Hello World runs
Apr 7	Testing Framework Complete
Apr 23	Final Scanner, Parser and AST Complete

4.3.1 Roles and Responsibilities

While we had defined project roles at the beginning of the semester, about three weeks in the roles became a lot more fluid. Our assigned roles were Tester, System Architect, Project Manager and Language Guru. Each member was involved in developing certain functionalities and portions of components. The team frequently worked together either in-person or teleconference.

Team Member	Role	Details
Husam Abdul-Kafi	Systems Architect	Code generation, Testing Architecture
Lakshmi Bodapati	Project Manager	Compiler Front-end, Documentation, Polymorphic Function Typing
Kaitlin Pet	Tester	Pitch and Chord Full Stack Abstraction, Library Linking, Testing Architecture
Harvey Wu	Language Guru	Type Inference, Compiler Front-end

4.3.2 Software Development Environment

1. Version Control: Git, Github
2. Languages: OCaml 4.04.0, C, Bash
3. Text Editor: Vim, Sublime Text, Atom
4. Operating System: Ubuntu 16.04, Mac OS X, Windows 10

5. Virtual Machine: Google Cloud

4.3.3 Project Log

05/10/17 11:25:27 C:\Users\husam\OneDrive\PLT\music-mike\log.log

- 1 c400293 was Harvey Wu, 35 minutes ago, message: Added stuff to Final Report
- 2 9f98afa was Harvey Wu, 44 minutes ago, message: Added fail on redefining stdlib functions
- 3 03489fb was Harvey Wu, 69 minutes ago, message: And more cleanup
- 4 447b852 was Harvey Wu, 70 minutes ago, message: More cleanup.:
- 5 356bda2 was Harvey Wu, 2 hours ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
- 6 dc595a1 was Harvey Wu, 2 hours ago, message: Lots of cleanup. Style issues. More tests.
- 7 db9c39d was Mounika, 2 hours ago, message: Merge pull request #78 from wuharvey/documentation
- 8 98c5e80 was Lakshmi Bodapati, 2 hours ago, message: completed first draft. Needs to be latexed and Appendix filled out and personal reflections filled out and codegen section. Other sections either need diagrams or examples. LRM needs to be updated and made a part of this document
- 9 9945ce8 was habdulkafi, 2 hours ago, message: added test script
- 10 23c2de9 was Lakshmi Bodapati, 2 hours ago, message: added in more details about modules
- 11 4915533 was Lakshmi Bodapati, 3 hours ago, message: presentation ppt and pdf
- 12 c62f155 was Harvey Wu, 4 hours ago, message: Important stuff for presentation
- 13 84101fc was Harvey Wu, 4 hours ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
- 14 4973127 was kpet123, 5 hours ago, message: Merge pull request #77 from wuharvey/husam-synth-new
- 15 99acf48 was Harvey Wu, 5 hours ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
- 16 1f9059e was Harvey Wu, 5 hours ago, message: Fixed semant
- 17 ef13d4f was kpet, 5 hours ago, message: yay!
- 18 b0edb37 was habdulkafi, 5 hours ago, message: stuff
- 19 ecd6017 was Husam Abdul-Kafi, 6 hours ago, message: Merge pull request #76 from wuharvey/synth-2
- 20 75725c1 was Husam Abdul-Kafi, 6 hours ago, message: Merge branch 'master' into synth-2
- 21 c11f8db was kpet, 6 hours ago, message: fixed make midi

```
22 8fa796c was kpet, 6 hours ago, message: Merge branch 'synth-2'
of https://github.com/wuharvey/music-mike into synth-2
23 e867c0d was kpet, 6 hours ago, message: test worksgit add
..../testall.sh
24 b02e6de was habdulkafi, 6 hours ago, message: updated cfugue
exe w/ cmd line args
25 f90fd79 was Husam Abdul-Kafi, 7 hours ago, message: Merge pull
request #75 from wuharvey/husam-strcat
26 e37d735 was habdulkafi, 7 hours ago, message: added string
concatenation
27 47d4f11 was Husam Abdul-Kafi, 7 hours ago, message: Merge pull
request #74 from wuharvey/semant-mounika
28 8ec4c14 was habdulkafi, 7 hours ago, message: fixed deleting
of non-poly funs
29 743bdf9 was kpet, 7 hours ago, message: switching branches
changed synth to read channels
30 e1cc90a was Harvey Wu, 8 hours ago, message: Removed wrong
tests
31 68ebe67 was Lakshmi Bodapati, 8 hours ago, message: Merge
branch 'synth-2' of https://github.com/wuharvey/music-mike
into synth-2
32 0f12965 was Lakshmi Bodapati, 8 hours ago, message: makemidi c
33 2d1f56a was Harvey Wu, 8 hours ago, message: Inference for
blocks
34 fb03be5 was Mounika, 8 hours ago, message: parens fixes
35 9a986d8 was Lakshmi Bodapati, 8 hours ago, message: make_midi
Mounika changes, testing on cloud
36 5076048 was Husam Abdul-Kafi, 8 hours ago, message: Merge pull
request #73 from wuharvey/master
37 f5bc384 was Harvey Wu, 9 hours ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike
38 c1f4047 was Harvey Wu, 9 hours ago, message: FIXED! INFER IS
BACK TO NORMAL
39 1174871 was Harvey Wu, 9 hours ago, message: Debugging
printing for Chord
40 83c5206 was Harvey Wu, 9 hours ago, message: Fixing inference
41 c266d3d was kpet123, 9 hours ago, message: Merge pull request
#71 from wuharvey/junk-branch
42 f730a08 was habdulkafi, 9 hours ago, message: Merge branch
'semant-mounika' of https://github.com/wuharvey/music-mike
into semant-mounika
43 4c0c348 was Husam Abdul-Kafi, 9 hours ago, message: Merge pull
```

```
request #70 from wuharvey/husam-functions
44 9fe34be was kpet, 9 hours ago, message: rests should work now
45 cfd68b9 was Harvey Wu, 9 hours ago, message: Merge branch
  'master' of https://github.com/wuharvey/music-mike
46 212208b was Harvey Wu, 9 hours ago, message: Merge pull
  request #69 from wuharvey/semant-mounika
47 596ac10 was Harvey Wu, 10 hours ago, message: Merge branch
  'semant-mounika' of https://github.com/wuharvey/music-mike
  into semant-mounika
48 dc907df was Harvey Wu, 10 hours ago, message: Semant testing
49 adafd38 was Mounika, 10 hours ago, message: project proposal
  update
50 6a192d9 was habdulkafi, 10 hours ago, message: fixed order of
  arguments in function decls
51 75b28b4 was Mounika, 10 hours ago, message: Harvey's infer
52 1451ec9 was habdulkafi, 10 hours ago, message: FUNCTIONS HERE
  WE COME
53 07a857f was Mounika, 10 hours ago, message: semant fixed
54 0712686 was kpet, 10 hours ago, message: sorry on master
55 2573816 was Harvey Wu, 11 hours ago, message: Merge branch
  'master' of https://github.com/wuharvey/music-mike
56 1b1a4b5 was habdulkafi, 11 hours ago, message: Merge branch
  'master' of https://github.com/wuharvey/music-mike
57 e3e17bf was habdulkafi, 11 hours ago, message: fixed up tests
58 037d186 was Mounika, 11 hours ago, message: Merge pull request
  #68 from wuharvey/semant
59 1235262 was Husam Abdul-Kafi, 23 hours ago, message: Merge
  pull request #67 from wuharvey/synth-in-codegen
60 30ae318 was habdulkafi, 23 hours ago, message: added new exe
  file (no idea if its different
61 36e5725 was habdulkafi, 24 hours ago, message: fixed length of
  rhythm
62 e2a2a0c was habdulkafi, 24 hours ago, message: debugging print
  statements everywhere
63 f47e2f6 was habdulkafi, 24 hours ago, message: added stuff we
  were missing
64 226bb6e was habdulkafi, 25 hours ago, message: changed test
  file
65 2592a3e was habdulkafi, 25 hours ago, message: commented out
  main. debugging prints
66 eb641c2 was habdulkafi, 25 hours ago, message: added
  chordlengths
```

67 3169529 was habdulkafi, 25 hours ago, message: fixed up
testall so it links properly
68 3f15a4f was habdulkafi, 27 hours ago, message: IT ALMOST
WORKS. IT'S NOT LINKING PROPERLY
69 883c57b was habdulkafi, 28 hours ago, message: trying to fix
synth in codegen
70 febce5f was habdulkafi, 32 hours ago, message: changed around
a bunch of types
71 0ab1fae was habdulkafi, 32 hours ago, message: reversed call
list
72 23d0d51 was habdulkafi, 32 hours ago, message: switched floats
to doubles
73 93e322e was kpet123, 32 hours ago, message: Merge pull request
#66 from wuharvey/test-cases
74 f1860c3 was kpet, 32 hours ago, message: more test cases
75 322fa4a was kpet, 33 hours ago, message: added new error cases
76 6b047ec was kpet, 33 hours ago, message: made more fail tests
cases
77 a313859 was habdulkafi, 2 days ago, message: fixed up
assignment of rhythm list and codegen for chord list
78 364dd81 was habdulkafi, 2 days ago, message: fixed up some
tests
79 21f3e5e was habdulkafi, 2 days ago, message: added more
expressive error messages for some errors
80 033c0bb was habdulkafi, 2 days ago, message: added printing
for rhythm list
81 189ef73 was kpet, 2 days ago, message: pushing compiling
branch
82 6b0ba65 was Harvey Wu, 2 days ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike
83 6015176 was kpet, 2 days ago, message: switching branches
(refactored chord list), need to test *)
84 9e50a35 was habdulkafi, 2 days ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike
85 932a181 was habdulkafi, 2 days ago, message: added pretty
printing for expr(Fun())
86 3611c25 was Husam Abdul-Kafi, 2 days ago, message: Merge pull
request #65 from wuharvey/semant-mounika
87 901d757 was Husam Abdul-Kafi, 2 days ago, message: Merge pull
request #64 from wuharvey/revert-63-semant-mounika
88 1052c70 was Husam Abdul-Kafi, 2 days ago, message: Revert
"Semant mounika"

89 01d3abc was Husam Abdul-Kafi, 2 days ago, message: Merge pull
request #63 from wuharvey/semant-mounika
90 683951c was habdulkafi, 2 days ago, message: added printing
for functions
91 be2a32e was Lakshmi Bodapati, 2 days ago, message: @
92 38a5f0b was Mounika, 2 days ago, message: Merge pull request
#62 from wuharvey/LRM-edits
93 737bad0 was Lakshmi Bodapati, 2 days ago, message: aexpr list
vs. aexpr problem
94 afad984 was Lakshmi Bodapati, 2 days ago, message: compile
fixes
95 85e28ab was Lakshmi Bodapati, 2 days ago, message: all the
logic is here?
96 17fb20f was Harvey Wu, 2 days ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike
97 da24b60 was Lakshmi Bodapati, 2 days ago, message: semant with
husam
98 352a82b was Lakshmi Bodapati, 2 days ago, message: need to
replace Afun dummy with Afun with types using the polycalls.
Struggling with syntax for mapping Acall to Afun with correct
types
99 758f93b was kpet, 2 days ago, message: pushing
100 fa93632 was kpet, 2 days ago, message: pushing branch,
inference doesn't work with other stuff
101 bef71bb was Harvey Wu, 2 days ago, message: Starting semantic
checker.
102 69a145f was kpet, 2 days ago, message: going to switch out
files from inference
103 26839b9 was kpet, 2 days ago, message: mergeMerge branch
'synth-in-codegen' of https://github.com/wuharvey/music-mike
into synth-in-codegen
104 0aadcc40 was Mounika, 3 days ago, message: remove Midi test
output
105 14870da was Mounika, 3 days ago, message: Merge pull request
#61 from wuharvey/synth-make
106 9037ae0 was Mounika, 3 days ago, message: working Makefile
that produces synth.o
107 1f0c93a was Lakshmi Bodapati, 3 days ago, message: make file
mimic microC
108 374bf36 was kpet, 3 days ago, message: nice version of actual
operators
109 b1166e9 was kpet, 3 days ago, message: synth in codegen

```
compiles, still needs testing
110 fc86caa was kpet, 3 days ago, message: getting weird area for
internal map function
111 23b19d6 was kpet, 3 days ago, message: switching branches
112 ec236fa was kpet, 3 days ago, message: generalized map
function
113 72bf5bb was Harvey Wu, 3 days ago, message: Updated Final
Report, changed style stuff.
114 196982f was kpet, 3 days ago, message: need to change branches
115 0bc072e was Harvey Wu, 3 days ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike
116 31ac72f was kpet, 4 days ago, message: added pointer list
structs
117 0969b09 was kpet, 4 days ago, message: yesMerge branch
'master' of https://github.com/wuharvey/music-mike
118 d8184b5 was Husam Abdul-Kafi, 4 days ago, message: Merge pull
request #58 from wuharvey/midi
119 047f260 was Harvey Wu, 4 days ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike
120 2bcb519 was habdulkafi, 4 days ago, message: Merge branch
'midi' of https://github.com/wuharvey/music-mike into midi
121 da8fb62 was habdulkafi, 4 days ago, message: commented the big
print function
122 e079301 was Mounika, 4 days ago, message: Merge pull request
#57 from wuharvey/midi
123 145e728 was Lakshmi Bodapati, 4 days ago, message: fixing
oopsie
124 6c5ea40 was Lakshmi Bodapati, 4 days ago, message: remove midi
125 68bd854 was Lakshmi Bodapati, 4 days ago, message: Merge
remote-tracking branch 'origin' into midi
126 dc21ddb was Lakshmi Bodapati, 4 days ago, message: oops and
CFugue deleted
127 76f8e19 was Lakshmi Bodapati, 4 days ago, message: remove
CFugue repos
128 984d4d7 was Mounika, 4 days ago, message: compiling synth with
the executable to produce the Midi thingy
129 4dce401 was Lakshmi Bodapati, 4 days ago, message: synth slash
fix
130 20b593d was Mounika, 4 days ago, message: song
131 5a4001d was Mounika, 4 days ago, message: C testing
132 4df745d was Mounika, 4 days ago, message: testing
133 751ae46 was Mounika, 4 days ago, message: C fixeseseseses
```

```
134 64187be was Lakshmi Bodapati, 4 days ago, message: synth to
linux
135 e7e9dab was Harvey Wu, 4 days ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike
136 a6af71b was habdulkafi, 4 days ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike
137 924378f was habdulkafi, 4 days ago, message: fixed subset?
138 f6b78ef was Harvey Wu, 4 days ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike
139 0de5779 was Harvey Wu, 4 days ago, message: Fixed subset to
type typeof return
140 97072a9 was Lakshmi Bodapati, 4 days ago, message: exec1
141 f1c6b92 was Mounika, 4 days ago, message: Merge pull request
#56 from wuharvey/final_report
142 d1e8865 was Lakshmi Bodapati, 4 days ago, message: Merge
branch 'master' of https://github.com/wuharvey/music-mike into
midi
143 ec8969f was Husam Abdul-Kafi, 4 days ago, message: Merge pull
request #55 from wuharvey/husam-struct-arrays
144 791c766 was habdulkafi, 4 days ago, message: added more
expressive not found error + took away list.rev
145 ad3a012 was habdulkafi, 4 days ago, message: fixed list type
to be struct
146 86db086 was habdulkafi, 4 days ago, message: added Printlist
function type
147 0ab3584 was Husam Abdul-Kafi, 4 days ago, message: Merge
branch 'master' into husam-struct-arrays
148 d14caa9 was kpet, 4 days ago, message: getting updated
versionMerge branch 'master' of
https://github.com/wuharvey/music-mike
149 db8cc19 was kpet123, 4 days ago, message: Create synth.c
150 56dad56 was Harvey Wu, 5 days ago, message: Subset type
inference
151 359d996 was Harvey Wu, 5 days ago, message: Useless commit
152 2f89a3d was Harvey Wu, 5 days ago, message: Debug information
for inference only printed with -s flag. Added rhythm list
inference
153 3666fa4 was Mounika, 6 days ago, message: duration syntax
154 6b9e414 was Mounika, 6 days ago, message: added brackets to
numbers
155 fa0f431 was Mounika, 6 days ago, message: trying out numerical
random chords 62q+65q+123q 3a+4h+8h
```

156 51fe8da was Mounika, 6 days ago, message: testing a chord
157 ed4eb40 was Harvey Wu, 6 days ago, message: Codegen takes
AEXPR instead of EXPR now.
158 55d53cc was Harvey Wu, 6 days ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike
159 7f1e9fd was Harvey Wu, 6 days ago, message: Merge pull request
#54 from wuharvey/inference
160 8d25ebc was Harvey Wu, 6 days ago, message: merge
161 75595a0 was Harvey Wu, 6 days ago, message: Trying to set up
flags for debug messages
162 e612a0e was Harvey Wu, 6 days ago, message: Merge branch
'inference' of https://github.com/wuharvey/music-mike into
inference
163 2c84db9 was Harvey Wu, 6 days ago, message: Update function
types in environment
164 0f8ae1f was Harvey Wu, 6 days ago, message: Type inference for
functions.
165 c152c94 was habdulkafi, 6 days ago, message: fixed subset into
list. added markings for where work is actually being done in
iter through list
166 3633d86 was habdulkafi, 6 days ago, message: GOT LLVM PRINTING
OF A LIST TO WORK
167 b3b4c09 was Harvey Wu, 7 days ago, message: (Almost) no lines
wrap over 80 chars now.
168 e6254cb was Harvey Wu, 7 days ago, message: Merge branch
'master' into inference
169 8da11fa was Harvey Wu, 7 days ago, message: Added Chord type
to ast
170 d0983f1 was Harvey Wu, 7 days ago, message: Added Keyword Set,
Assign inference, style tweaks to infer.ml. Corrected the
order of inference. TODO: Replace dummy type in env after
Assign.
171 563f853 was Harvey Wu, 7 days ago, message: Fixed some stuff
for strings and pretty printing
172 8209193 was Harvey Wu, 7 days ago, message: Changed pretty
printing to aid debugging. Fixed list inference
173 611e202 was habdulkafi, 7 days ago, message: fixed up sample
app so it takes cmd line args
174 4869072 was Mounika, 7 days ago, message: sample ap changes
that aren't compiling
175 66abc45 was Mounika, 7 days ago, message: Mounika testing
stuff

176 704eb76 was Harvey Wu, 8 days ago, message: Added parentheses
for prec escalation
177 9286291 was Harvey Wu, 8 days ago, message: Type inference
works for If statements
178 3e2779f was habdulkafi, 8 days ago, message: fixed simple
printing error
179 fb53591 was habdulkafi, 8 days ago, message: added pretty
printing for chordlist
180 93e4360 was Harvey Wu, 8 days ago, message: Added lib.ml for
stdlib. Changed Makefile to include lib. Modified musicmike.ml
to take -s flag to test type inference. Related changes.
181 005a03b was habdulkafi, 8 days ago, message: changed structure
to only take expressions
182 04ee773 was habdulkafi, 8 days ago, message: messed with ID
wrt Call and Fun
183 7c9cb0f was habdulkafi, 8 days ago, message: fixed up codegen
wrt function definitions
184 6391ab6 was Harvey Wu, 8 days ago, message: make frontend
compiles now
185 3f32795 was Harvey Wu, 8 days ago, message: Merge pull request
#53 from wuharvey/infer
186 d445d78 was Harvey Wu, 8 days ago, message: Merge branch
'master' into infer
187 66187ed was Harvey Wu, 8 days ago, message: Added pretty
printing for aexprs to AST
188 0e3f7a4 was Harvey Wu, 9 days ago, message: Changed Makefile
and musicmike.ml
189 682765a was Harvey Wu, 10 days ago, message: Inference module
now compiles. Changed fdecl in parser. Minor modifications to
ast.ml
190 be3a01f was Harvey Wu, 2 weeks ago, message: Added stuff to
infer. Got rid of user defined types.
191 192ea8d was Harvey Wu, 2 weeks ago, message: Added more stuff
to infer.
192 3db649f was Lakshmi Bodapati, 2 weeks ago, message: Wrote more
stuff for type inference.
193 ec47966 was Harvey Wu, 2 weeks ago, message: Did a shit ton of
stuff yo don't even remember what i did
194 252748e was Mounika, 2 weeks ago, message: Merge pull request
#37 from wuharvey/music-operators
195 9b25747 was kpet, 2 weeks ago, message: hacked rhythmlist
196 8ded65a was Lakshmi Bodapati, 2 weeks ago, message: TestPlan

197 cb5883d was Mounika, 2 weeks ago, message: Merge branch
'master' into music-operators

198 14c150c was Lakshmi Bodapati, 2 weeks ago, message:
architecture section

199 a198521 was Lakshmi Bodapati, 2 weeks ago, message: I tried to
somehow define what each of us worked on but it's kinda hard
since we didn't do it by an assigned module basis like other
groups seemed to have

200 a069108 was kpet, 2 weeks ago, message: changed name of
pitchlist to chordlist

201 91b29a7 was kpet, 2 weeks ago, message: added empty rhythm
test

202 8d03848 was kpet, 2 weeks ago, message: compiles- rhythmlist

203 2b2e880 was kpet, 2 weeks ago, message: added bar to scanner

204 b41299f was Lakshmi Bodapati, 2 weeks ago, message: rough
roles and responsibilities portion

205 a8c239a was Lakshmi Bodapati, 2 weeks ago, message: Timeline

206 16fac51 was Lakshmi Bodapati, 2 weeks ago, message: finished
project process and style guide

207 820e911 was Mounika, 2 weeks ago, message: Merge pull request
#36 from wuharvey/mounika_semant

208 0e52b05 was Lakshmi Bodapati, 2 weeks ago, message: mergeMerge
branch 'master' of <https://github.com/wuharvey/music-mike> into
mounika_semant

209 6dc6478 was Lakshmi Bodapati, 2 weeks ago, message:
Specifications

210 62f7662 was kpet, 2 weeks ago, message: the last one didn't
work- getting weird error : Scanner.is_pat ref ->
Lexing.lexbuf -> Parser.token

211 8742ef2 was kpet, 2 weeks ago, message: fixed indexing error

212 4b3f962 was kpet, 2 weeks ago, message: it comiles! pitches as
expressions

213 7223852 was kpet, 2 weeks ago, message: getting weird bug in
codegen pitches: Error: This expression has type int *
'a' but an expression was expected of type 'b * 'c * 'd

214 9083b6d was Lakshmi Bodapati, 2 weeks ago, message: project
plan halfway done

215 d89d880 was Lakshmi Bodapati, 2 weeks ago, message: added
stuff we need from proposal

216 00e650b was kpet, 2 weeks ago, message: merging with scanner
changes Merge branch 'master' of
<https://github.com/wuharvey/music-mike>

217 ac99249 was kpet, 2 weeks ago, message: forgot these
218 ba681f3 was kpet, 2 weeks ago, message: implemented rlist in
codegen, need to sync with semant
219 3d63c44 was Husam Abdul-Kafi, 2 weeks ago, message: Merge pull
request #35 from wuharvey/husam-if-stmnt
220 76f6c38 was habdulkafi, 2 weeks ago, message: added tests for
basic if statements
221 dd129df was habdulkafi, 2 weeks ago, message: added basic if-
then-else functionality
222 f3dbb5a was kpet, 2 weeks ago, message: llvm seems to be
working
223 6fd77a5 was kpet, 2 weeks ago, message: it compiles!
224 6533bcd was kpet, 2 weeks ago, message: changed orientation
again (inner functions sandwiched between mallocs and iter)
225 3e1b780 was habdulkafi, 2 weeks ago, message: added basic test
for function call
226 24d5cd3 was habdulkafi, 2 weeks ago, message: added basics of
function calls
227 408867d was habdulkafi, 2 weeks ago, message: got rid of
reduce/reduce conflicts. added ability to have 0 params fun
def and fun calls
228 3ad1502 was kpet, 2 weeks ago, message: pushing code
229 fe06d32 was kpet, 2 weeks ago, message: modified version of
pitch list no syntax errors at least
230 787b2e4 was Mounika, 2 weeks ago, message: Merge pull request
#33 from wuharvey/mounika_semant
231 1e3e04c was Mounika, 2 weeks ago, message: change FID
232 a922768 was Mounika, 2 weeks ago, message: Merge pull request
#32 from wuharvey/mounika_semant
233 9fc77c3 was Mounika, 2 weeks ago, message: test output
234 671efd8 was Mounika, 2 weeks ago, message: add rhythm test
file
235 47e260c was Mounika, 2 weeks ago, message: parsing error now!
236 315da1f was kpet, 2 weeks ago, message: simplified pitch
+ast.ml
237 45418dd was kpet, 2 weeks ago, message: simplitified pitch
238 a78bcf8 was Mounika, 2 weeks ago, message: compiles
239 5aab008 was Mounika, 3 weeks ago, message: sorta
working...can't quite debug
240 aab45ba was Lakshmi Bodapati, 3 weeks ago, message: change
call to scanner
241 cfce11d was Lakshmi Bodapati, 3 weeks ago, message: parse

rhythm separately

242 f8d1851 was kpet, 3 weeks ago, message: reformatted list.iteri

243 5d04360 was kpet, 3 weeks ago, message: changing to master

244 4d73d55 was kpet, 3 weeks ago, message: let-in doesn't match, tried reducing problem to list of lists

245 caf80ac was kpet, 3 weeks ago, message: fixed indices

246 9fe5bab was kpet, 3 weeks ago, message: second crack, now at least code makes some sense

247 75e8944 was kpet, 3 weeks ago, message: implemented pitch list still need to test :'(

248 41b3ba4 was kpet, 3 weeks ago, message: working on pitchlist

249 9d1edd2 was habdulkafi, 3 weeks ago, message: fixed up pitch list in parser, ast. commented out pitch list in codegen for now

250 9db67f2 was kpet, 3 weeks ago, message: codegen attempt to parser pitch list

251 feffec9 was kpet, 3 weeks ago, message: annotated parser. Issues:cannot have empty lists

252 210e481 was kpet, 3 weeks ago, message: it compilesgit add parser.mly ast.mlgit add parser.mly ast.mlgit add parser.mly ast.ml!

253 604a686 was kpet, 3 weeks ago, message: getting weird error: This expression has type Ast.chord = Ast.pitch list but an expression was expected of type Ast.pitch = int list * int * int list

254 4033493 was kpet, 3 weeks ago, message: working on ast/parser structure

255 dd8124c was kpet, 3 weeks ago, message: scanner and parser with logically consistant updates

256 ccc12d7 was kpet, 3 weeks ago, message: restructured parser so pitches easier to access

257 62afa61 was kpet, 4 weeks ago, message: changed parser and ast

258 3bb8123 was kpet, 4 weeks ago, message: added operations in codegen

259 5580c7b was kpet, 4 weeks ago, message: first commit in pitch operation edits

260 dd6a056 was kpet, 4 weeks ago, message: added LRM and edited some parts

261 f864f7a was kpet, 4 weeks ago, message: nothing worth saving on Kaitlin end Merge branch 'master' of <https://github.com/wuharvey/music-mike>

262 c5ea544 was kpet, 4 weeks ago, message: saving codegen and

scanner so can pull
263 e28b6bf was Husam Abdul-Kafi, 4 weeks ago, message: Merge pull
request #31 from wuharvey/husam-fun-decl
264 a15e49d was habdulkafi, 4 weeks ago, message: added basic
function declaration codegen and test file
265 7e7563e was Harvey Wu, 4 weeks ago, message: Removed mike-
files directory.
266 99be096 was Lakshmi Bodapati, 4 weeks ago, message: scanner
change
267 2a534f0 was Lakshmi Bodapati, 4 weeks ago, message: testing
syntax
268 4fd20f3 was Lakshmi Bodapati, 4 weeks ago, message: more tests
269 e91f565 was Lakshmi Bodapati, 4 weeks ago, message: test
270 97e9950 was Lakshmi Bodapati, 4 weeks ago, message: testing
271 4f6c493 was Lakshmi Bodapati, 4 weeks ago, message: oopsie
272 afea966 was Lakshmi Bodapati, 4 weeks ago, message: testing
273 3316a04 was Lakshmi Bodapati, 4 weeks ago, message: scanner
274 426d243 was Lakshmi Bodapati, 4 weeks ago, message: new
approach
275 ff3495f was Lakshmi Bodapati, 4 weeks ago, message: add seed
for rhythm list
276 86d53dd was Lakshmi Bodapati, 4 weeks ago, message: parser
compiles with no conflicts
277 15b8b06 was Lakshmi Bodapati, 4 weeks ago, message: Rlist
278 254bda2 was Lakshmi Bodapati, 4 weeks ago, message: changed
parser and scanner to scan in rhythm list
279 58149cc was Husam Abdul-Kafi, 4 weeks ago, message: Merge pull
request #30 from wuharvey/husam-fun-def
280 8e59d52 was habdulkafi, 4 weeks ago, message: added test to
test block return
281 ddafdf82 was habdulkafi, 4 weeks ago, message: reversed list of
exprs in block
282 da614ff was habdulkafi, 4 weeks ago, message: added code
generation for blocks
283 8ab5162 was Husam Abdul-Kafi, 4 weeks ago, message: Merge pull
request #29 from wuharvey/husam-fun-def
284 bf36fe8 was habdulkafi, 4 weeks ago, message: fixing
differences in the merge
285 b7aebef was habdulkafi, 4 weeks ago, message: added parens in
fn calls. fixed test files for prints
286 beb578e was kpet123, 4 weeks ago, message: Update parser.mly
287 366d161 was kpet123, 4 weeks ago, message: Merge pull request

```
#28 from wuharvey/temp
288 a63dc12 was kpet, 4 weeks ago, message: deleted test files
289 99220ce was Harvey Wu, 4 weeks ago, message: Merge branch
  'mounika_semant' of https://github.com/wuharvey/music-mike
  into mounika_semant
290 ec5900a was Harvey Wu, 4 weeks ago, message: removed EOF
291 0de0e08 was kpet, 4 weeks ago, message: added test files
292 4cdc308 was Harvey Wu, 4 weeks ago, message: Corrected
  spelling for RList
293 d89408f was Lakshmi Bodapati, 4 weeks ago, message: make
  uppercase in ast
294 a4dba31 was Lakshmi Bodapati, 4 weeks ago, message: pretty
  print rhythm list
295 ad48a09 was Lakshmi Bodapati, 4 weeks ago, message: ast to
  include rhythm list
296 f45108c was Lakshmi Bodapati, 4 weeks ago, message: mounika
  list attempts
297 3c8db43 was Lakshmi Bodapati, 4 weeks ago, message: possible
  working scanner lists?
298 bf2a686 was habdulkafi, 4 weeks ago, message: added ; in
  parser and beginning of function declarations in codegen.  BAD
  STATE
299 c269baa was habdulkafi, 4 weeks ago, message: changed test
  files to have ;
300 cee5ee9 was Lakshmi Bodapati, 4 weeks ago, message: merge
  Merge branch 'master' of https://github.com/wuharvey/music-
  mike into mounika_semant
301 d18378f was Lakshmi Bodapati, 4 weeks ago, message: fix all
  function calls
302 230d8ca was Harvey Wu, 4 weeks ago, message: Updated string
  stuff for scanner
303 e561352 was Lakshmi Bodapati, 4 weeks ago, message: L and R
  paren stuff + fixed a bad merge
304 f332007 was Lakshmi Bodapati, 4 weeks ago, message: merged
305 32531a9 was Lakshmi Bodapati, 4 weeks ago, message:
  parenthesis to functions compiles with I just make the parser
306 647e542 was Lakshmi Bodapati, 4 weeks ago, message: add
  parenthesis to parsing function calls
307 22b3205 was Lakshmi Bodapati, 4 weeks ago, message: commenting
  out stuff I don't think we need
308 3341ebd was habdulkafi, 4 weeks ago, message: fixed up error
  files for tests.  modified codegen to output error
```

309 f10564c was habdulkafi, 5 weeks ago, message: Merge branch
'master' of <https://github.com/wuharvey/music-mike> i'm not
sure what's going on..

310 4f3c550 was habdulkafi, 5 weeks ago, message: fixed issue #26
! added a list.rev to exprs list. fixed test cases

311 d8fd50e was Husam Abdul-Kafi, 5 weeks ago, message: Merge pull
request #27 from wuharvey/husam-arrays

312 1800b54 was habdulkafi, 5 weeks ago, message: added subsetting
to codegen and test files

313 54314d8 was habdulkafi, 5 weeks ago, message: elements in the
list were reversed - fixed now. fixed subsetting

314 1c2759c was habdulkafi, 5 weeks ago, message: added .[to
scanner

315 7a2ca69 was habdulkafi, 5 weeks ago, message: changed Sub -->
Subset

316 07ad1e3 was habdulkafi, 5 weeks ago, message: assigning arrays
to variables works now

317 550072e was habdulkafi, 5 weeks ago, message: fixed up
assignment and lookup using a hash table

318 bde3c88 was habdulkafi, 5 weeks ago, message: added SEMI after
assignment. The output of the parser is still wrong - the
expressions are reverse order

319 26bb9f2 was habdulkafi, 5 weeks ago, message: added tests that
*shouldn't fail (but they do)

320 e74a42a was habdulkafi, 5 weeks ago, message: added verbose
output to Makefile

321 4e3e4d7 was habdulkafi, 5 weeks ago, message: added start of
array implementation

322 d99dd56 was habdulkafi, 6 weeks ago, message: fixed up test
files, merged codegen

323 e821cc8 was habdulkafi, 6 weeks ago, message: fixed float
printing. added assignment

324 bc5e576 was Harvey Wu, 6 weeks ago, message: Merge pull
request #25 from wuharvey/Harvey-Codegen2

325 0469af1 was Husam Abdul-Kafi, 6 weeks ago, message: Merge pull
request #24 from wuharvey/Harvey-Codegen1

326 7aed9be was habdulkafi, 6 weeks ago, message: fixed up codegen
and added test set-up

327 8da1996 was Harvey Wu, 6 weeks ago, message: Added float
printing to Codegen. Added test files.

328 d648034 was Harvey Wu, 6 weeks ago, message: Merge branch
'Harvey-Codegen1' of <https://github.com/wuharvey/music-mike>

```
    into Harvey-Codegen1
329 423ce70 was habdulkafi, 6 weeks ago, message: HELLO WORLD
WORKSgit statusgit status!
330 fbaaab2 was Harvey Wu, 6 weeks ago, message: Merge branch
'Harvey-Codegen1' of https://github.com/wuharvey/music-mike
into Harvey-Codegen1
331 2dec649 was habdulkafi, 6 weeks ago, message: p-->P simple
makefile edit
332 d01e532 was Harvey Wu, 6 weeks ago, message: Merge branch
'Harvey-Codegen1' of https://github.com/wuharvey/music-mike
into Harvey-Codegen1
333 4218f5d was Harvey Wu, 6 weeks ago, message: Changed to
capital P
334 4ea71f2 was Mounika, 6 weeks ago, message: fix syntax for far
away test file
335 ac8bd84 was Mounika, 6 weeks ago, message: test file syntax
fixes
336 b040d13 was Harvey Wu, 6 weeks ago, message: Renamed microc ->
musicmike in Makefile and .ml file. Added more stuff to
codegen while maintaining compilability
337 83b0bd5 was Harvey Wu, 6 weeks ago, message: Update README.md
338 1bf659d was Harvey Wu, 6 weeks ago, message: Merge pull
request #21 from wuharvey/Harvey-Codegen1
339 fa036e9 was habdulkafi, 6 weeks ago, message: added string
conversion to ast file
340 30f25d5 was habdulkafi, 6 weeks ago, message: codegen is all
commented out, but general structure of what we want is still
there
341 6d9d337 was habdulkafi, 6 weeks ago, message: commented out
unused tokens cause it complained about them being unused
342 08e6e29 was habdulkafi, 6 weeks ago, message: added ocamlbuild
workaround so it compiles on my system
343 9ea5b9a was habdulkafi, 6 weeks ago, message: added the
microc.ml file (unchanged). added semant.ml file w/ all of it
commented out and it returns unit()
344 d9e92e0 was Harvey Wu, 6 weeks ago, message: Removed
irrelevant things in codegen.
345 03e16b2 was Harvey Wu, 6 weeks ago, message: Added more
pattern matching to type expr in AST.
346 68cf1e4 was Harvey Wu, 6 weeks ago, message: Added auxillary
functions for 3-tuple in header. Changed high-level structure
```

of grammar to lists of expressions, fdecls, and tdecls. Added assign_list to specify lists of assignments for tdecl.

347 63fd50e was Harvey Wu, 6 weeks ago, message: Added make command for parser/ast combination compilation

348 94e0249 was Harvey Wu, 7 weeks ago, message: Removed statement from ast and added relevant types to EXPR. Fixed small issues in parser.

349 6bf70cc was Harvey Wu, 7 weeks ago, message: Added Makefile (only use currently is make clean)

350 60c03ac was Harvey Wu, 7 weeks ago, message: Merged codegen edits with master

351 132db13 was Harvey Wu, 7 weeks ago, message: Cleaned up Codegen, minor modifications to parser/scanner

352 032ee30 was Lakshmi Bodapati, 7 weeks ago, message: repo organization

353 77266ae was Kaitlin Pet, 7 weeks ago, message: adding tests to master

354 b7ee13e was Harvey Wu, 7 weeks ago, message: Merge pull request #19 from wuharvey/Kaitlin_edits_to_parser

355 bec83ba was Harvey Wu, 7 weeks ago, message: Merge branch 'master' into Kaitlin_edits_to_parser

356 7144505 was Harvey Wu, 7 weeks ago, message: Added FID to AST and added float stuff to codegen

357 fa776de was Harvey Wu, 7 weeks ago, message: Merge pull request #18 from wuharvey/cleanup

358 1f08473 was Harvey Wu, 7 weeks ago, message: Merge branch 'master' into cleanup

359 dcd2a25 was kpet123, 7 weeks ago, message: ADDED SET

360 69f015d was kpet123, 7 weeks ago, message: added set

361 1fb4aa7 was kpet123, 7 weeks ago, message: Update parser.mly

362 16d40fb was kpet123, 7 weeks ago, message: Update parser.mly

363 8e19bf5 was Kaitlin Pet, 7 weeks ago, message: about to push it

364 2af2b9a was Kaitlin Pet, 7 weeks ago, message: created enclosed_expression, which can go to id or stuff in parenthesis

365 720079c was Kaitlin Pet, 7 weeks ago, message: moved assign to primaries and sub to expr

366 b6e8d1c was Kaitlin Pet, 7 weeks ago, message: made semi semi for sequencing and connected funct_list explicitly to function declaration, also created actuals_list

367 5f9615c was Harvey Wu, 7 weeks ago, message: Fixed function call SR conflicts by adding a %prec

368 b52bbc5 was Harvey Wu, 7 weeks ago, message: Negating floats
should not require a -. so deleted corresponding rule in
parser.

369 8399c5b was Harvey Wu, 7 weeks ago, message: Added precedence
for rhythmdot. Fixed other S/R conflicts - five left and an
R/R.

370 4bf34ba was Harvey Wu, 7 weeks ago, message: added a primaries
category per python and renamed FLiteral to FloatLit for
consistency

371 9b446dc was Harvey Wu, 7 weeks ago, message: found source of
most S/R conflicts and moved to complex expr section

372 d9d3def was Harvey Wu, 7 weeks ago, message: fixed some yacc
errors. now to fix 127 sr conflicts

373 008ac56 was Harvey Wu, 7 weeks ago, message: Removed
references to whitesp_list and rewrote formals/actuals stuff +
other minor cleanup

374 30b8d67 was Husam Abdul-Kafi, 8 weeks ago, message: Ha codegen
(#15)

375 3c01fde was kpet123, 8 weeks ago, message: Update parser.mly

376 79a043c was Harvey Wu, 8 weeks ago, message: added some
definitions of types and fixed style issues for ast.ml

377 9a8a145 was Harvey Wu, 8 weeks ago, message: Readded
formals_list. Fixed style issues. Fixed regexes for scanner.

378 816d99a was kpet123, 8 weeks ago, message: Update ast.ml

379 b66a557 was kpet123, 8 weeks ago, message: Update parser.mly

380 a3f3d07 was kpet123, 8 weeks ago, message: still needs for
loop + concat

381 1780564 was kpet123, 8 weeks ago, message: Update ast.ml

382 85cd517 was kpet123, 8 weeks ago, message: Update parser.mly

383 a927f7a was kpet123, 8 weeks ago, message: most changes are
commented in code

384 799cd1c was kpet123, 8 weeks ago, message: Update scanner.mll

385 c6a1be9 was Harvey Wu, 8 weeks ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike

386 f4488c1 was Harvey Wu, 8 weeks ago, message: Removed most
MicroC stuff from parser. Added proper fdecl.

387 59b108a was Harvey Wu, 8 weeks ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike

388 b75cb5f was Harvey Wu, 8 weeks ago, message: Merge branch
'master' of https://github.com/wuharvey/music-mike

389 5863ba3 was Harvey Wu, 8 weeks ago, message: Added float
operations to parser. Added new type FID (function ID) to

scanner for later convenience. Corrected regex for identifiers to account for reserved single characters.

390 58136e4 was kpet123, 9 weeks ago, message: Update cfg.txt
391 ae697ee was kpet123, 9 weeks ago, message: Update cfg.txt
392 c8d4fce was kpet123, 9 weeks ago, message: Update cfg.txt
393 18a5016 was kpet123, 9 weeks ago, message: Update cfg.txt
394 45cc6cf was kpet123, 9 weeks ago, message: Update cfg.txt
395 7948745 was kpet123, 9 weeks ago, message: Update cfg.txt
396 5acae19 was kpet123, 9 weeks ago, message: Update cfg.txt
397 979e163 was kpet123, 9 weeks ago, message: Add files via upload

398 c72ad97 was kpet123, 10 weeks ago, message: Update ast.ml
399 4e002d4 was habdulkafi, 10 weeks ago, message: added the tokens to the token list in the parser and the associativity (very wrong, but it's a start)

400 97bf924 was habdulkafi, 10 weeks ago, message: added microc parser and ast

401 e728c44 was habdulkafi, 10 weeks ago, message: added brackets, indexing, fops, pitchops, concat, and float literals to scanner

402 11f1b6f was habdulkafi, 10 weeks ago, message: added microc scanner

403 5f0b992 was Harvey Wu, 3 months ago, message: Update Project Proposal.md

404 5fd2a6b was Harvey Wu, 8 weeks ago, message: Added float operations to parser. Added new type FID (function ID) to scanner for later convenience. Corrected regex for identifiers to account for reserved single characters.

405 7be535e was kpet123, 9 weeks ago, message: Update cfg.txt
406 d44e2bb was kpet123, 9 weeks ago, message: Update cfg.txt
407 e6648b6 was kpet123, 9 weeks ago, message: Update cfg.txt
408 1361d6a was kpet123, 9 weeks ago, message: Update cfg.txt
409 ca233b9 was kpet123, 9 weeks ago, message: Update cfg.txt
410 21b7115 was kpet123, 9 weeks ago, message: Update cfg.txt
411 31eb6c8 was kpet123, 9 weeks ago, message: Update cfg.txt
412 5eb91d5 was kpet123, 9 weeks ago, message: Add files via upload

413 b611496 was kpet123, 10 weeks ago, message: Update ast.ml
414 4cbc462 was habdulkafi, 10 weeks ago, message: added the tokens to the token list in the parser and the associativity (very wrong, but it's a start)

415 6845db8 was habdulkafi, 10 weeks ago, message: added microc

parser and ast

416 c0e9527 was habdulkafi, 10 weeks ago, message: added brackets, indexing, fops, pitchops, concat, and float literals to scanner

417 3d1705a was habdulkafi, 10 weeks ago, message: added microc scanner

418 0b9047f was Harvey Wu, 3 months ago, message: Update Project Proposal.md

419 b02d6bf was Harvey Wu, 3 months ago, message: Update Project Proposal.md

420 752894c was Harvey Wu, 3 months ago, message: Update Project Proposal.md

421 3ab535a was Harvey Wu, 3 months ago, message: Update Project Proposal.md

422 01d9e79 was Harvey Wu, 3 months ago, message: Update Project Proposal.md

423 36d7140 was Harvey Wu, 3 months ago, message: Update Project Proposal.md

424 a50edc7 was Harvey Wu, 3 months ago, message: Update Project Proposal.md

425 d72ea47 was Harvey Wu, 3 months ago, message: Update Project Proposal.md

426 0de79a0 was Harvey Wu, 3 months ago, message: Update Project Proposal.md

427 7209a75 was Harvey Wu, 3 months ago, message: Update Project Proposal.md

428 e0727d4 was Harvey Wu, 3 months ago, message: Update Project Proposal.md

429 ef5b0b6 was Harvey Wu, 3 months ago, message: Update Project Proposal.md

430 072c5a1 was Harvey Wu, 3 months ago, message: Create README.md

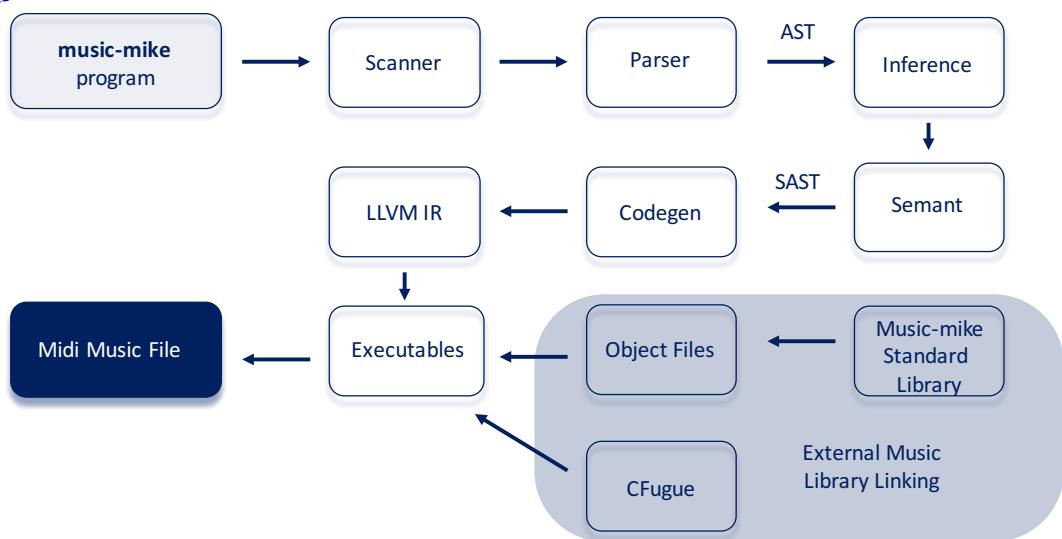
431 2e89fe7 was Harvey Wu, 3 months ago, message: Project Proposal draft

5. Architectural Design

5.1 Block Diagram



Compiler Architecture



5.2 Interfaces

5.2.1 Scanner (Lakshmi, Husam, Harvey)

Relevant Files: scanner.mll

The scanner is written in Ocamllex and takes the .mike input to the compiler and tokenizes it into literals, identifiers, operators and keywords. It removes the white space and block comments. If any character cannot be lexed by the scanner or if any identifier or literal is not syntactically valid, the scanner throws an error. The tokens created by the scanner are used by the Parser to create an Abstract Syntax Tree. The scanner is context sensitive, so uses different pattern matching inside rhythm and pitch lists.

5.2.2 Parser (Kaitlin, Husam, Harvey, Lakshmi)

Relevant Files: parser.mly

The parser is written in Ocaml yacc and takes in a series of tokens. It uses the grammar described in parser.mly and datatypes defined in ast.ml to generate an Abstract Syntax Tree. In parser.mly, we define the Music-mike context-free grammar using productions. If the tokens produced by the scanner are successfully parsed that means that the .mike file is syntactically (though perhaps not semantically) correct.

Certain operations are also directly translated to values in the parser. For example, the pitch prefield and postfield operators (Raise by 1 octave, Lower

by 1 octave, Raise by half step, and lower by half step) were parsed directly into numerical values to be applied to the note value during the calling of the Synth function.

5.2.3 Type Inference (Harvey)

Relevant Files: infer.ml

The type inference module takes in the untyped AST of expressions and runs Algorithm W (Hindley Milner Type Inference). Type checking is also done in one swift pass - a list, for example, cannot contain elements of two different types. Scope is also handled in this module - a single global map stores variables and their types and thus typing is dynamic. Variables and functions cannot be redefined. Other semantic checking, such as argument counts

5.2.4 Semant (Lakshmi, Husam, Harvey)

Relevant Files: semant.ml

The semant module is an extension of type inference, and handles polymorphic functions. It first finds all the polymorphic functions, then finds all the calls made to these functions and adds in typed function definitions in place of the dummy type polymorphic functions.

5.2.5 The Code Generator (Husam, Kaitlin, Lakshmi)

Relevant Files: codegen.ml

Our Codegen was similar to that of MicroC with the exception of a few key features. First of all, everything is an expression and evaluates to either a value or void. This structure made it easy for us to nest expressions within one another and take advantage of OCaml's pattern-matching abilities.

Especially notable was our plethora of structures to manipulate lists. To keeps track of lengths of lists, all lists are not pure pointers but consist of a struct where the first field is length and the second field is a pointer to the list itself. Because the second field contains a pointer, it is sufficient for all single-layer lists.

For multi-layer lists such as the Chordlist structure, this struct was repeated across multiple layers. The data structure consists of a chordlist consisting of chords where each chord consists of pitches and each pitch consists of a 3-tuple. In this case, length information was necessary for chordlist and chord, so those structures were the structs as described above. Since length information was not important for pitches (malloced every time at a set length of 3), pointers to pitches were just int *.

6. Test Plan

6.1 Example tests

05/10/17 11:47:33 C:\Users\husam\Downloads\bach.mike

```
1 minor1 = [11 13 14 16 18 20 22 23 9];
2 minor2 = [11 13 14 16 18 20 22 23 10];
3 theme = p:[0 1 3 5 8 7 8 7b 8 6 8 6b 8 5 8 7 8 4# 6 3 2 3 4# 2
1 v7b 5];
4 r1 = r:[0. .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1
.1 .1 .1 .1 .1 .1 .1 .2 .2];
5 r2 = r:[2.5 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1
.1 .1 .1 .1 .1 .1 .1 .2 .2];
6
7 counter = p:[0 5 6 5 4 3 2 4 3 2 1 v7 1 2 v7 1];
8 r3 = r:[.1 .1 .1 .1 .1 .1 .1 .5 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1];
9 arp = p:[0 v5 v7 2 4 v7 2 4 6 5 4 6 5 4 3 2 1 v7 v6 v5 3];
10 r4 = r:[s s s s s s s s s s s s s s s s s s s s s s 1.25];
11
12 rone = Synth(theme r1 minor1 50 1);
13 rtwo = Synth(counter r3 minor1 (50 + 7) 1);
14 lone = Synth(theme r2 minor1 33 2);
15
16 song = Merge(rone lone);
17 song1 = Merge(song rtwo);
18 Make_midi(song1 "bach.midi");
```

05/10/17 11:44:39 C:\Users\husam\Downloads\bach.txt

```
1
2 duleID = 'MusicMike'
3
4 %list_struct = type <{ i32, i32* }>
5 %chordlist_struct = type <{ i32, %chord_struct* }>
6 %chord_struct = type <{ i32, i32** }>
7 %list_struct_f = type <{ i32, double* }>
8
9 @fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
10 @str = private unnamed_addr constant [4 x i8] c"%s\0A\00"
11 @flt = private unnamed_addr constant [4 x i8] c"%f\0A\00"
12 @str.1 = private unnamed_addr constant [3 x i8] c"%c\00"
13 @fmt.2 = private unnamed_addr constant [4 x i8] c"%d \00"
14 @fmt.3 = private unnamed_addr constant [4 x i8] c"%f \00"
15 @0 = private unnamed_addr constant [10 x i8] c"bach.midi\00"
16
17 declare i32 @printf(i8*, ...)
18
19 define i32 @main() {
20 entry:
21     %array_struct = alloca %list_struct
22     %length = getelementptr inbounds %list_struct,
23     %list_struct* %array_struct, i32 0, i32 0
24     store i32 9, i32* %length
25     %array = alloca i32, i32 9
26     %elem = getelementptr i32, i32* %array, i32 0
27     store i32 11, i32* %elem
28     %elem1 = getelementptr i32, i32* %array, i32 1
29     store i32 13, i32* %elem1
30     %elem2 = getelementptr i32, i32* %array, i32 2
31     store i32 14, i32* %elem2
32     %elem3 = getelementptr i32, i32* %array, i32 3
33     store i32 16, i32* %elem3
34     %elem4 = getelementptr i32, i32* %array, i32 4
35     store i32 18, i32* %elem4
36     %elem5 = getelementptr i32, i32* %array, i32 5
37     store i32 20, i32* %elem5
38     %elem6 = getelementptr i32, i32* %array, i32 6
39     store i32 22, i32* %elem6
40     %elem7 = getelementptr i32, i32* %array, i32 7
```

```

40  store i32 23, i32* %elem7
41  %elem8 = getelementptr i32, i32* %array, i32 8
42  store i32 9, i32* %elem8
43  %actual_list = getelementptr inbounds %list_struct,
44    %list_struct* %array_struct, i32 0, i32 1
45  store i32* %array, i32** %actual_list
46  %minor1 = alloca %list_struct*
47  store %list_struct* %array_struct, %list_struct** %minor1
48  %array_struct9 = alloca %list_struct
49  %length10 = getelementptr inbounds %list_struct,
50    %list_struct* %array_struct9, i32 0, i32 0
51  store i32 9, i32* %length10
52  %array11 = alloca i32, i32 9
53  %elem12 = getelementptr i32, i32* %array11, i32 0
54  store i32 11, i32* %elem12
55  %elem13 = getelementptr i32, i32* %array11, i32 1
56  store i32 13, i32* %elem13
57  %elem14 = getelementptr i32, i32* %array11, i32 2
58  store i32 14, i32* %elem14
59  %elem15 = getelementptr i32, i32* %array11, i32 3
60  store i32 16, i32* %elem15
61  %elem16 = getelementptr i32, i32* %array11, i32 4
62  store i32 18, i32* %elem16
63  %elem17 = getelementptr i32, i32* %array11, i32 5
64  store i32 20, i32* %elem17
65  %elem18 = getelementptr i32, i32* %array11, i32 6
66  store i32 22, i32* %elem18
67  %elem19 = getelementptr i32, i32* %array11, i32 7
68  store i32 23, i32* %elem19
69  %elem20 = getelementptr i32, i32* %array11, i32 8
70  store i32 10, i32* %elem20
71  %actual_list21 = getelementptr inbounds %list_struct,
72    %list_struct* %array_struct9, i32 0, i32 1
73  store i32* %array11, i32** %actual_list21
74  %minor2 = alloca %list_struct*
75  store %list_struct* %array_struct9, %list_struct** %minor2
    %malloccall = tail call i8* @malloc(i32 ptrtoint
      (%chordlist_struct* getelementptr (%chordlist_struct,
      %chordlist_struct* null, i32 1) to i32))
76  %cl_struct = bitcast i8* %malloccall to %chordlist_struct*
77  %length22 = getelementptr inbounds %chordlist_struct,
    %chordlist_struct* %cl_struct, i32 0, i32 0

```

```

76    store i32 27, i32* %length22
77    %malloccall123 = tail call i8* @malloc(i32 mul (i32 ptrtoint
    (%chord_struct* getelementptr (%chord_struct, %chord_struct*
    null, i32 1) to i32), i32 27))
78    %chord_pointer_array = bitcast i8* %malloccall123 to
    %chord_struct*
79    %pointer_chord_elem_list = getelementptr %chord_struct,
    %chord_struct* %chord_pointer_array, i32 0
80    %malloccall124 = tail call i8* @malloc(i32 ptrtoint
    (%chord_struct* getelementptr (%chord_struct, %chord_struct*
    null, i32 1) to i32))
81    %chord_struct = bitcast i8* %malloccall124 to %chord_struct*
82    %length25 = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct, i32 0, i32 0
83    store i32 1, i32* %length25
84    %malloccall126 = tail call i8* @malloc(i32 ptrtoint (i1**
    getelementptr (i1*, i1** null, i32 1) to i32))
85    %arr_pitch = bitcast i8* %malloccall126 to i32**
86    %pitch_pointer_elem = getelementptr i32*, i32** %arr_pitch,
    i32 0
87    %malloccall127 = tail call i8* @malloc(i32 mul (i32 ptrtoint
    (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
88    %array28 = bitcast i8* %malloccall127 to i32*
89    %prefield_elem = getelementptr i32, i32* %array28, i32 0
90    store i32 0, i32* %prefield_elem
91    %scaledegreer_elem = getelementptr i32, i32* %array28, i32
    1
92    store i32 0, i32* %scaledegreer_elem
93    %postfield_elem = getelementptr i32, i32* %array28, i32 2
94    store i32 0, i32* %postfield_elem
95    store i32* %array28, i32** %pitch_pointer_elem
96    %struct_c_pointer = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct, i32 0, i32 1
97    store i32** %arr_pitch, i32*** %struct_c_pointer
98    %actual_chord_struct = load %chord_struct, %chord_struct*
    %chord_struct
99    store %chord_struct %actual_chord_struct, %chord_struct*
    %pointer_chord_elem_list
100   %pointer_chord_elem_list29 = getelementptr %chord_struct,
    %chord_struct* %chord_pointer_array, i32 1
101   %malloccall130 = tail call i8* @malloc(i32 ptrtoint
    (%chord_struct* getelementptr (%chord_struct, %chord_struct*

```

```

    null, i32 1) to i32))
102    %chord_struct31 = bitcast i8* %malloccall30 to
    %chord_struct*
103    %length32 = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct31, i32 0, i32 0
104    store i32 1, i32* %length32
105    %malloccall33 = tail call i8* @malloc(i32 ptrtoint (i1**
        getelementptr (i1*, i1** null, i32 1) to i32))
106    %arr_pitch34 = bitcast i8* %malloccall33 to i32**
107    %pitch_pointer_elem35 = getelementptr i32*, i32**
        %arr_pitch34, i32 0
108    %malloccall36 = tail call i8* @malloc(i32 mul (i32 ptrtoint
        (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
109    %array37 = bitcast i8* %malloccall36 to i32*
110    %prefield_elem38 = getelementptr i32, i32* %array37, i32 0
111    store i32 0, i32* %prefield_elem38
112    %scaledegreer_elem39 = getelementptr i32, i32* %array37,
        i32 1
113    store i32 1, i32* %scaledegreer_elem39
114    %postfield_elem40 = getelementptr i32, i32* %array37, i32 2
115    store i32 0, i32* %postfield_elem40
116    store i32* %array37, i32** %pitch_pointer_elem35
117    %struct_c_pointer41 = getelementptr inbounds %chord_struct,
        %chord_struct* %chord_struct31, i32 0, i32 1
118    store i32** %arr_pitch34, i32*** %struct_c_pointer41
119    %actual_chord_struct42 = load %chord_struct, %chord_struct*
        %chord_struct31
120    store %chord_struct %actual_chord_struct42, %chord_struct*
        %pointer_chord_elem_list29
121    %pointer_chord_elem_list43 = getelementptr %chord_struct,
        %chord_struct* %chord_pointer_array, i32 2
122    %malloccall44 = tail call i8* @malloc(i32 ptrtoint
        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
            null, i32 1) to i32))
123    %chord_struct45 = bitcast i8* %malloccall44 to
        %chord_struct*
124    %length46 = getelementptr inbounds %chord_struct,
        %chord_struct* %chord_struct45, i32 0, i32 0
125    store i32 1, i32* %length46
126    %malloccall47 = tail call i8* @malloc(i32 ptrtoint (i1**
        getelementptr (i1*, i1** null, i32 1) to i32))
127    %arr_pitch48 = bitcast i8* %malloccall47 to i32**

```

```

128    %pitch_pointer_elem49 = getelementptr i32*, i32**
129        %arr_pitch48, i32 0
130    %malloccall150 = tail call i8* @malloc(i32 mul (i32 ptrtoint
131        (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
132    %array51 = bitcast i8* %malloccall150 to i32*
133    %prefield_elem52 = getelementptr i32, i32* %array51, i32 0
132    store i32 0, i32* %prefield_elem52
133    %scaledegreer_elem53 = getelementptr i32, i32* %array51,
132 1
134    store i32 3, i32* %scaledegreer_elem53
135    %postfield_elem54 = getelementptr i32, i32* %array51, i32 2
136    store i32 0, i32* %postfield_elem54
137    store i32* %array51, i32** %pitch_pointer_elem49
138    %struct_c_pointer55 = getelementptr inbounds %chord_struct,
139        %chord_struct* %chord_struct45, i32 0, i32 1
139    store i32** %arr_pitch48, i32*** %struct_c_pointer55
140    %actual_chord_struct56 = load %chord_struct, %chord_struct*
141        %chord_struct45
141    store %chord_struct %actual_chord_struct56, %chord_struct*
142        %pointer_chord_elem_list43
142    %pointer_chord_elem_list57 = getelementptr %chord_struct,
143        %chord_struct* %chord_pointer_array, i32 3
143    %malloccall158 = tail call i8* @malloc(i32 ptrtoint
144        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
144            null, i32 1) to i32))
144    %chord_struct59 = bitcast i8* %malloccall158 to
145        %chord_struct*
145    %length60 = getelementptr inbounds %chord_struct,
146        %chord_struct* %chord_struct59, i32 0, i32 0
146    store i32 1, i32* %length60
147    %malloccall161 = tail call i8* @malloc(i32 ptrtoint (i1**
148        getelementptr (i1*, i1** null, i32 1) to i32))
148    %arr_pitch62 = bitcast i8* %malloccall161 to i32**
149    %pitch_pointer_elem63 = getelementptr i32*, i32**
149        %arr_pitch62, i32 0
150    %malloccall164 = tail call i8* @malloc(i32 mul (i32 ptrtoint
151        (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
151    %array65 = bitcast i8* %malloccall164 to i32*
152    %prefield_elem66 = getelementptr i32, i32* %array65, i32 0
153    store i32 0, i32* %prefield_elem66
154    %scaledegreer_elem67 = getelementptr i32, i32* %array65,
154        i32 1

```

```

155 store i32 5, i32* %scaledegreer_elem67
156 %postfield_elem68 = getelementptr i32, i32* %array65, i32 2
157 store i32 0, i32* %postfield_elem68
158 store i32* %array65, i32** %pitch_pointer_elem63
159 %struct_c_pointer69 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct59, i32 0, i32 1
160 store i32** %arr_pitch62, i32*** %struct_c_pointer69
161 %actual_chord_struct70 = load %chord_struct, %chord_struct*
%chord_struct59
162 store %chord_struct %actual_chord_struct70, %chord_struct*
%pointer_chord_elem_list57
163 %pointer_chord_elem_list71 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array, i32 4
164 %malloccall172 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
165 %chord_struct73 = bitcast i8* %malloccall172 to
%chord_struct*
166 %length74 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct73, i32 0, i32 0
167 store i32 1, i32* %length74
168 %malloccall175 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
169 %arr_pitch76 = bitcast i8* %malloccall175 to i32**
170 %pitch_pointer_elem77 = getelementptr i32*, i32**
%arr_pitch76, i32 0
171 %malloccall178 = tail call i8* @malloc(i32 mul (i32 ptrtoint
(i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
172 %array79 = bitcast i8* %malloccall178 to i32*
173 %prefield_elem80 = getelementptr i32, i32* %array79, i32 0
174 store i32 0, i32* %prefield_elem80
175 %scaledegreer_elem81 = getelementptr i32, i32* %array79,
i32 1
176 store i32 8, i32* %scaledegreer_elem81
177 %postfield_elem82 = getelementptr i32, i32* %array79, i32 2
178 store i32 0, i32* %postfield_elem82
179 store i32* %array79, i32** %pitch_pointer_elem77
180 %struct_c_pointer83 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct73, i32 0, i32 1
181 store i32** %arr_pitch76, i32*** %struct_c_pointer83
182 %actual_chord_struct84 = load %chord_struct, %chord_struct*
%chord_struct73

```

```

183   store %chord_struct %actual_chord_struct84, %chord_struct*
184     %pointer_chord_elem_list71
185   %pointer_chord_elem_list85 = getelementptr %chord_struct,
186     %chord_struct* %chord_pointer_array, i32 5
187   %malloccall86 = tail call i8* @malloc(i32 ptrtoint
188     (%chord_struct* getelementptr (%chord_struct, %chord_struct*
189       null, i32 1) to i32))
190   %chord_struct87 = bitcast i8* %malloccall86 to
191     %chord_struct*
192   %length88 = getelementptr inbounds %chord_struct,
193     %chord_struct* %chord_struct87, i32 0, i32 0
194   store i32 1, i32* %length88
195   %malloccall89 = tail call i8* @malloc(i32 ptrtoint (i1**
196     getelementptr (i1*, i1** null, i32 1) to i32))
197   %arr_pitch90 = bitcast i8* %malloccall89 to i32**
198   %pitch_pointer_elem91 = getelementptr i32*, i32**
199     %arr_pitch90, i32 0
200   %malloccall192 = tail call i8* @malloc(i32 mul (i32 ptrtoint
201     (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
202   %array93 = bitcast i8* %malloccall192 to i32*
203   %prefield_elem94 = getelementptr i32, i32* %array93, i32 0
204   store i32 0, i32* %prefield_elem94
205   %scaledegreer_elem95 = getelementptr i32, i32* %array93,
206     i32 1
207   store i32 7, i32* %scaledegreer_elem95
208   %postfield_elem96 = getelementptr i32, i32* %array93, i32 2
209   store i32 0, i32* %postfield_elem96
210   store i32* %array93, i32** %pitch_pointer_elem91
211   %struct_c_pointer97 = getelementptr inbounds %chord_struct,
212     %chord_struct* %chord_struct87, i32 0, i32 1
213   store i32** %arr_pitch90, i32*** %struct_c_pointer97
214   %actual_chord_struct98 = load %chord_struct, %chord_struct*
215     %chord_struct87
216   store %chord_struct %actual_chord_struct98, %chord_struct*
217     %pointer_chord_elem_list85
218   %pointer_chord_elem_list99 = getelementptr %chord_struct,
219     %chord_struct* %chord_pointer_array, i32 6
220   %malloccall100 = tail call i8* @malloc(i32 ptrtoint
221     (%chord_struct* getelementptr (%chord_struct, %chord_struct*
222       null, i32 1) to i32))
223   %chord_struct101 = bitcast i8* %malloccall100 to
224     %chord_struct*

```

```

208    %length102 = getelementptr inbounds %chord_struct,
209    %chord_struct* %chord_struct101, i32 0, i32 0
210    store i32 1, i32* %length102
211    %malloccall103 = tail call i8* @malloc(i32 ptrtoint (i1**
212        getelementptr (i1*, i1** null, i32 1) to i32))
213    %arr_pitch104 = bitcast i8* %malloccall103 to i32**
214    %pitch_pointer_elem105 = getelementptr i32*, i32**
215    %arr_pitch104, i32 0
216    %malloccall106 = tail call i8* @malloc(i32 mul (i32
217        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
218        i32 3))
219    %array107 = bitcast i8* %malloccall106 to i32*
220    %prefield_elem108 = getelementptr i32, i32* %array107, i32
221    0
222    store i32 0, i32* %prefield_elem108
223    %scaledegreer_elem109 = getelementptr i32, i32* %array107,
224    i32 1
225    store i32 8, i32* %scaledegreer_elem109
226    %postfield_elem110 = getelementptr i32, i32* %array107, i32
227    2
228    store i32 0, i32* %postfield_elem110
229    store i32* %array107, i32** %pitch_pointer_elem105
230    %struct_c_pointer111 = getelementptr inbounds
231    %chord_struct, %chord_struct* %chord_struct101, i32 0, i32 1
232    store i32** %arr_pitch104, i32*** %struct_c_pointer111
233    %actual_chord_struct112 = load %chord_struct,
234    %chord_struct* %chord_struct101
235    store %chord_struct %actual_chord_struct112, %chord_struct*
236    %pointer_chord_elem_list99
237    %pointer_chord_elem_list113 = getelementptr %chord_struct,
238    %chord_struct* %chord_pointer_array, i32 7
239    %malloccall114 = tail call i8* @malloc(i32 ptrtoint
240        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
241            null, i32 1) to i32))
242    %chord_struct115 = bitcast i8* %malloccall114 to
243    %chord_struct*
244    %length116 = getelementptr inbounds %chord_struct,
245    %chord_struct* %chord_struct115, i32 0, i32 0
246    store i32 1, i32* %length116
247    %malloccall117 = tail call i8* @malloc(i32 ptrtoint (i1**
248        getelementptr (i1*, i1** null, i32 1) to i32))
249    %arr_pitch118 = bitcast i8* %malloccall117 to i32**

```

```

233    %pitch_pointer_elem119 = getelementptr i32*, i32**
234        %arr_pitch118, i32 0
235    %malloccall1120 = tail call i8* @malloc(i32 mul (i32
236        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
237        i32 3))
238    %array121 = bitcast i8* %malloccall1120 to i32*
239    %prefield_elem122 = getelementptr i32, i32* %array121, i32
240        0
241    store i32 0, i32* %prefield_elem122
242    %scaledegreer_elem123 = getelementptr i32, i32* %array121,
243        i32 1
244    store i32 7, i32* %scaledegreer_elem123
245    %postfield_elem124 = getelementptr i32, i32* %array121, i32
246        2
247    store i32 -1, i32* %postfield_elem124
248    store i32* %array121, i32** %pitch_pointer_elem119
249    %struct_c_pointer125 = getelementptr inbounds
250        %chord_struct, %chord_struct* %chord_struct115, i32 0, i32 1
251    store i32** %arr_pitch118, i32*** %struct_c_pointer125
252    %actual_chord_struct126 = load %chord_struct,
253        %chord_struct* %chord_struct115
254    store %chord_struct %actual_chord_struct126, %chord_struct*
255        %pointer_chord_elem_list113
256    %pointer_chord_elem_list127 = getelementptr %chord_struct,
257        %chord_struct* %chord_pointer_array, i32 8
258    %malloccall1128 = tail call i8* @malloc(i32 ptrtoint
259        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
260            null, i32 1) to i32))
261    %chord_struct129 = bitcast i8* %malloccall1128 to
262        %chord_struct*
263    %length130 = getelementptr inbounds %chord_struct,
264        %chord_struct* %chord_struct129, i32 0, i32 0
265    store i32 1, i32* %length130
266    %malloccall1131 = tail call i8* @malloc(i32 ptrtoint (i1**
267        getelementptr (i1*, i1** null, i32 1) to i32))
268    %arr_pitch132 = bitcast i8* %malloccall1131 to i32**
269    %pitch_pointer_elem133 = getelementptr i32*, i32**
270        %arr_pitch132, i32 0
271    %malloccall1134 = tail call i8* @malloc(i32 mul (i32
272        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
273        i32 3))
274    %array135 = bitcast i8* %malloccall1134 to i32*

```

```

257  %prefield_elem136 = getelementptr i32, i32* %array135, i32
258  0
259  store i32 0, i32* %prefield_elem136
260  %scaledegreer_elem137 = getelementptr i32, i32* %array135,
261  i32 1
262  store i32 8, i32* %scaledegreer_elem137
263  %postfield_elem138 = getelementptr i32, i32* %array135, i32
264  2
265  store i32 0, i32* %postfield_elem138
266  store i32* %array135, i32** %pitch_pointer_elem133
267  %struct_c_pointer139 = getelementptr inbounds
268  %chord_struct, %chord_struct* %chord_struct129, i32 0, i32 1
269  store i32** %arr_pitch132, i32*** %struct_c_pointer139
270  %actual_chord_struct140 = load %chord_struct,
271  %chord_struct* %chord_struct129
272  store %chord_struct %actual_chord_struct140, %chord_struct*
273  %pointer_chord_elem_list127
274  %pointer_chord_elem_list141 = getelementptr %chord_struct,
275  %chord_struct* %chord_pointer_array, i32 9
276  %malloccall142 = tail call i8* @malloc(i32 ptrtoint
277  (%chord_struct* getelementptr (%chord_struct, %chord_struct*
278  null, i32 1) to i32))
279  %chord_struct143 = bitcast i8* %malloccall142 to
280  %chord_struct*
281  %length144 = getelementptr inbounds %chord_struct,
282  %chord_struct* %chord_struct143, i32 0, i32 0
283  store i32 1, i32* %length144
284  %malloccall145 = tail call i8* @malloc(i32 ptrtoint (i1**
285  getelementptr (i1*, i1** null, i32 1) to i32))
286  %arr_pitch146 = bitcast i8* %malloccall145 to i32**
287  %pitch_pointer_elem147 = getelementptr i32*, i32**
288  %arr_pitch146, i32 0
289  %malloccall148 = tail call i8* @malloc(i32 mul (i32
290  ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
291  i32 3))
292  %array149 = bitcast i8* %malloccall148 to i32*
293  %prefield_elem150 = getelementptr i32, i32* %array149, i32
294  0
295  store i32 0, i32* %prefield_elem150
296  %scaledegreer_elem151 = getelementptr i32, i32* %array149,
297  i32 1
298  store i32 6, i32* %scaledegreer_elem151

```

```

282    %postfield_elem152 = getelementptr i32, i32* %array149, i32
2
283    store i32 0, i32* %postfield_elem152
284    store i32* %array149, i32** %pitch_pointer_elem147
285    %struct_c_pointer153 = getelementptr inbounds
286        %chord_struct, %chord_struct* %chord_struct143, i32 0, i32 1
287    store i32** %arr_pitch146, i32*** %struct_c_pointer153
288    %actual_chord_struct154 = load %chord_struct,
289        %chord_struct* %chord_struct143
290    store %chord_struct %actual_chord_struct154, %chord_struct*
291        %pointer_chord_elem_list141
292    %pointer_chord_elem_list155 = getelementptr %chord_struct,
293        %chord_struct* %chord_pointer_array, i32 10
294    %malloccall156 = tail call i8* @malloc(i32 ptrtoint
295        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
296            null, i32 1) to i32))
297    %chord_struct157 = bitcast i8* %malloccall156 to
298        %chord_struct*
299    %length158 = getelementptr inbounds %chord_struct,
300        %chord_struct* %chord_struct157, i32 0, i32 0
301    store i32 1, i32* %length158
302    %malloccall159 = tail call i8* @malloc(i32 ptrtoint (i1**
303        getelementptr (i1*, i1** null, i32 1) to i32))
304    %arr_pitch160 = bitcast i8* %malloccall159 to i32**
305    %pitch_pointer_elem161 = getelementptr i32*, i32**
306        %arr_pitch160, i32 0
307    %malloccall162 = tail call i8* @malloc(i32 mul (i32
308        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
309        i32 3))
310    %array163 = bitcast i8* %malloccall162 to i32*
311    %prefield_elem164 = getelementptr i32, i32* %array163, i32
312    0
313    store i32 0, i32* %prefield_elem164
314    %scaledegreer_elem165 = getelementptr i32, i32* %array163,
315    i32 1
316    store i32 8, i32* %scaledegreer_elem165
317    %postfield_elem166 = getelementptr i32, i32* %array163, i32
318    2
319    store i32 0, i32* %postfield_elem166
320    store i32* %array163, i32** %pitch_pointer_elem161
321    %struct_c_pointer167 = getelementptr inbounds
322        %chord_struct, %chord_struct* %chord_struct157, i32 0, i32 1

```

```

307    store i32** %arr_pitch160, i32*** %struct_c_pointer167
308    %actual_chord_struct168 = load %chord_struct,
309    %chord_struct* %chord_struct157
310    store %chord_struct %actual_chord_struct168, %chord_struct*
311    %pointer_chord_elem_list155
310    %pointer_chord_elem_list169 = getelementptr %chord_struct,
311    %chord_struct* %chord_pointer_array, i32 11
311    %malloccall170 = tail call i8* @malloc(i32 ptrtoint
312    (%chord_struct* getelementptr (%chord_struct, %chord_struct*
313    null, i32 1) to i32))
312    %chord_struct171 = bitcast i8* %malloccall170 to
313    %chord_struct*
313    %length172 = getelementptr inbounds %chord_struct,
314    %chord_struct* %chord_struct171, i32 0, i32 0
314    store i32 1, i32* %length172
315    %malloccall173 = tail call i8* @malloc(i32 ptrtoint (i1**
316    getelementptr (i1*, i1** null, i32 1) to i32))
316    %arr_pitch174 = bitcast i8* %malloccall173 to i32**
317    %pitch_pointer_elem175 = getelementptr i32*, i32**
317    %arr_pitch174, i32 0
318    %malloccall176 = tail call i8* @malloc(i32 mul (i32
319    ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
320    i32 3))
319    %array177 = bitcast i8* %malloccall176 to i32*
320    %prefield_elem178 = getelementptr i32, i32* %array177, i32
321    0
321    store i32 0, i32* %prefield_elem178
322    %scaledegreer_elem179 = getelementptr i32, i32* %array177,
323    i32 1
323    store i32 6, i32* %scaledegreer_elem179
324    %postfield_elem180 = getelementptr i32, i32* %array177, i32
324    2
325    store i32 -1, i32* %postfield_elem180
326    store i32* %array177, i32** %pitch_pointer_elem175
327    %struct_c_pointer181 = getelementptr inbounds
327    %chord_struct, %chord_struct* %chord_struct171, i32 0, i32 1
328    store i32** %arr_pitch174, i32*** %struct_c_pointer181
329    %actual_chord_struct182 = load %chord_struct,
329    %chord_struct* %chord_struct171
330    store %chord_struct %actual_chord_struct182, %chord_struct*
330    %pointer_chord_elem_list169
331    %pointer_chord_elem_list183 = getelementptr %chord_struct,

```

```

%chord_struct* %chord_pointer_array, i32 12
332  %malloccall184 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
333  %chord_struct185 = bitcast i8* %malloccall184 to
%chord_struct*
334  %length186 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct185, i32 0, i32 0
335  store i32 1, i32* %length186
336  %malloccall187 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
337  %arr_pitch188 = bitcast i8* %malloccall187 to i32**
338  %pitch_pointer_elem189 = getelementptr i32*, i32**
%arr_pitch188, i32 0
339  %malloccall190 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
340  %array191 = bitcast i8* %malloccall190 to i32*
341  %prefield_elem192 = getelementptr i32, i32* %array191, i32
0
342  store i32 0, i32* %prefield_elem192
343  %scaledegreer_elem193 = getelementptr i32, i32* %array191,
i32 1
344  store i32 8, i32* %scaledegreer_elem193
345  %postfield_elem194 = getelementptr i32, i32* %array191, i32
2
346  store i32 0, i32* %postfield_elem194
347  store i32* %array191, i32** %pitch_pointer_elem189
348  %struct_c_pointer195 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct185, i32 0, i32 1
349  store i32** %arr_pitch188, i32*** %struct_c_pointer195
350  %actual_chord_struct196 = load %chord_struct,
%chord_struct* %chord_struct185
351  store %chord_struct %actual_chord_struct196, %chord_struct*
%pointer_chord_elem_list183
352  %pointer_chord_elem_list197 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array, i32 13
353  %malloccall198 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
354  %chord_struct199 = bitcast i8* %malloccall198 to
%chord_struct*

```

```

355    %length200 = getelementptr inbounds %chord_struct,
356    %chord_struct* %chord_struct199, i32 0, i32 0
357    store i32 1, i32* %length200
358    %malloccall201 = tail call i8* @malloc(i32 ptrtoint (i1**
359        getelementptr (i1*, i1** null, i32 1) to i32))
360    %arr_pitch202 = bitcast i8* %malloccall201 to i32**
361    %pitch_pointer_elem203 = getelementptr i32*, i32**
362        %arr_pitch202, i32 0
363    %malloccall204 = tail call i8* @malloc(i32 mul (i32
364        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
365        i32 3))
366    %array205 = bitcast i8* %malloccall204 to i32*
367    %prefield_elem206 = getelementptr i32, i32* %array205, i32
368        0
369    store i32 0, i32* %prefield_elem206
370    %scaledegreer_elem207 = getelementptr i32, i32* %array205,
371        i32 1
372    store i32 5, i32* %scaledegreer_elem207
373    %postfield_elem208 = getelementptr i32, i32* %array205, i32
374        2
375    store i32 0, i32* %postfield_elem208
376    store i32* %array205, i32** %pitch_pointer_elem203
377    %struct_c_pointer209 = getelementptr inbounds
378        %chord_struct, %chord_struct* %chord_struct199, i32 0, i32 1
379    store i32** %arr_pitch202, i32*** %struct_c_pointer209
380    %actual_chord_struct210 = load %chord_struct,
381        %chord_struct* %chord_struct199
382    store %chord_struct %actual_chord_struct210, %chord_struct*
383        %pointer_chord_elem_list197
384    %pointer_chord_elem_list211 = getelementptr %chord_struct,
385        %chord_struct* %chord_pointer_array, i32 14
386    %malloccall212 = tail call i8* @malloc(i32 ptrtoint
387        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
388            null, i32 1) to i32))
389    %chord_struct213 = bitcast i8* %malloccall212 to
390        %chord_struct*
391    %length214 = getelementptr inbounds %chord_struct,
392        %chord_struct* %chord_struct213, i32 0, i32 0
393    store i32 1, i32* %length214
394    %malloccall215 = tail call i8* @malloc(i32 ptrtoint (i1**
395        getelementptr (i1*, i1** null, i32 1) to i32))
396    %arr_pitch216 = bitcast i8* %malloccall215 to i32**

```

```

380    %pitch_pointer_elem217 = getelementptr i32*, i32**
381        %arr_pitch216, i32 0
382    %malloccall218 = tail call i8* @malloc(i32 mul (i32
383        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
384        i32 3))
385    %array219 = bitcast i8* %malloccall218 to i32*
386    %prefield_elem220 = getelementptr i32, i32* %array219, i32
387        0
388    store i32 0, i32* %prefield_elem220
389    %scaledegreer_elem221 = getelementptr i32, i32* %array219,
390        i32 1
391    store i32 8, i32* %scaledegreer_elem221
392    %postfield_elem222 = getelementptr i32, i32* %array219, i32
393        2
394    store i32 0, i32* %postfield_elem222
395    store i32* %array219, i32** %pitch_pointer_elem217
396    %struct_c_pointer223 = getelementptr inbounds
397        %chord_struct, %chord_struct* %chord_struct213, i32 0, i32 1
398    store i32** %arr_pitch216, i32*** %struct_c_pointer223
399    %actual_chord_struct224 = load %chord_struct,
400        %chord_struct* %chord_struct213
401    store %chord_struct %actual_chord_struct224, %chord_struct*
402        %pointer_chord_elem_list211
403    %pointer_chord_elem_list225 = getelementptr %chord_struct,
404        %chord_struct* %chord_pointer_array, i32 15
405    %malloccall226 = tail call i8* @malloc(i32 ptrtoint
406        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
407            null, i32 1) to i32))
408    %chord_struct227 = bitcast i8* %malloccall226 to
409        %chord_struct*
410    %length228 = getelementptr inbounds %chord_struct,
411        %chord_struct* %chord_struct227, i32 0, i32 0
412    store i32 1, i32* %length228
413    %malloccall229 = tail call i8* @malloc(i32 ptrtoint (i1**
414        getelementptr (i1*, i1** null, i32 1) to i32))
415    %arr_pitch230 = bitcast i8* %malloccall229 to i32**
416    %pitch_pointer_elem231 = getelementptr i32*, i32**
417        %arr_pitch230, i32 0
418    %malloccall232 = tail call i8* @malloc(i32 mul (i32
419        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
420        i32 3))
421    %array233 = bitcast i8* %malloccall232 to i32*

```

```

404    %prefield_elem234 = getelementptr i32, i32* %array233, i32
405        0
406        store i32 0, i32* %prefield_elem234
407        %scaledegreer_elem235 = getelementptr i32, i32* %array233,
408            i32 1
409        store i32 7, i32* %scaledegreer_elem235
410        %postfield_elem236 = getelementptr i32, i32* %array233, i32
411            2
412        store i32 0, i32* %postfield_elem236
413        store i32* %array233, i32** %pitch_pointer_elem231
414        %struct_c_pointer237 = getelementptr inbounds
415            %chord_struct, %chord_struct* %chord_struct227, i32 0, i32 1
416        store i32** %arr_pitch230, i32*** %struct_c_pointer237
417        %actual_chord_struct238 = load %chord_struct,
418            %chord_struct* %chord_struct227
419        store %chord_struct %actual_chord_struct238, %chord_struct*
420            %pointer_chord_elem_list225
421        %pointer_chord_elem_list239 = getelementptr %chord_struct,
422            %chord_struct* %chord_pointer_array, i32 16
423        %malloccall240 = tail call i8* @malloc(i32 ptrtoint
424            (%chord_struct* getelementptr (%chord_struct, %chord_struct*
425                null, i32 1) to i32))
426        %chord_struct241 = bitcast i8* %malloccall240 to
427            %chord_struct*
428        %length242 = getelementptr inbounds %chord_struct,
429            %chord_struct* %chord_struct241, i32 0, i32 0
430        store i32 1, i32* %length242
431        %malloccall243 = tail call i8* @malloc(i32 ptrtoint (i1**
432            getelementptr (i1*, i1** null, i32 1) to i32))
433        %arr_pitch244 = bitcast i8* %malloccall243 to i32**
434        %pitch_pointer_elem245 = getelementptr i32*, i32**
435            %arr_pitch244, i32 0
436        %malloccall246 = tail call i8* @malloc(i32 mul (i32
437            ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
438            i32 3))
439        %array247 = bitcast i8* %malloccall246 to i32*
440        %prefield_elem248 = getelementptr i32, i32* %array247, i32
441            0
442        store i32 0, i32* %prefield_elem248
443        %scaledegreer_elem249 = getelementptr i32, i32* %array247,
444            i32 1
445        store i32 8, i32* %scaledegreer_elem249

```

```

429  %postfield_elem250 = getelementptr i32, i32* %array247, i32
2
430  store i32 0, i32* %postfield_elem250
431  store i32* %array247, i32** %pitch_pointer_elem245
432  %struct_c_pointer251 = getelementptr inbounds
    %chord_struct, %chord_struct* %chord_struct241, i32 0, i32 1
433  store i32** %arr_pitch244, i32*** %struct_c_pointer251
434  %actual_chord_struct252 = load %chord_struct,
    %chord_struct* %chord_struct241
435  store %chord_struct %actual_chord_struct252, %chord_struct*
    %pointer_chord_elem_list239
436  %pointer_chord_elem_list253 = getelementptr %chord_struct,
    %chord_struct* %chord_pointer_array, i32 17
437  %malloccall254 = tail call i8* @malloc(i32 ptrtoint
    (%chord_struct* getelementptr (%chord_struct, %chord_struct*
        null, i32 1) to i32))
438  %chord_struct255 = bitcast i8* %malloccall254 to
    %chord_struct*
439  %length256 = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct255, i32 0, i32 0
440  store i32 1, i32* %length256
441  %malloccall257 = tail call i8* @malloc(i32 ptrtoint (i1**
    getelementptr (i1*, i1** null, i32 1) to i32))
442  %arr_pitch258 = bitcast i8* %malloccall257 to i32**
443  %pitch_pointer_elem259 = getelementptr i32*, i32**
    %arr_pitch258, i32 0
444  %malloccall260 = tail call i8* @malloc(i32 mul (i32
    ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
    i32 3))
445  %array261 = bitcast i8* %malloccall260 to i32*
446  %prefield_elem262 = getelementptr i32, i32* %array261, i32
0
447  store i32 0, i32* %prefield_elem262
448  %scaledegreer_elem263 = getelementptr i32, i32* %array261,
    i32 1
449  store i32 4, i32* %scaledegreer_elem263
450  %postfield_elem264 = getelementptr i32, i32* %array261, i32
2
451  store i32 1, i32* %postfield_elem264
452  store i32* %array261, i32** %pitch_pointer_elem259
453  %struct_c_pointer265 = getelementptr inbounds
    %chord_struct, %chord_struct* %chord_struct255, i32 0, i32 1

```

```

454     store i32** %arr_pitch258, i32*** %struct_c_pointer265
455     %actual_chord_struct266 = load %chord_struct,
456     %chord_struct* %chord_struct255
457     store %chord_struct %actual_chord_struct266, %chord_struct*
458     %pointer_chord_elem_list253
459     %pointer_chord_elem_list267 = getelementptr %chord_struct,
460     %chord_struct* %chord_pointer_array, i32 18
461     %malloccall268 = tail call i8* @malloc(i32 ptrtoint
462     (%chord_struct* getelementptr (%chord_struct, %chord_struct*
463     null, i32 1) to i32))
464     %chord_struct269 = bitcast i8* %malloccall268 to
465     %chord_struct*
466     %length270 = getelementptr inbounds %chord_struct,
467     %chord_struct* %chord_struct269, i32 0, i32 0
468     store i32 1, i32* %length270
469     %malloccall271 = tail call i8* @malloc(i32 ptrtoint (i1**
470     getelementptr (i1*, i1** null, i32 1) to i32))
471     %arr_pitch272 = bitcast i8* %malloccall271 to i32**
472     %pitch_pointer_elem273 = getelementptr i32*, i32**
473     %arr_pitch272, i32 0
474     %malloccall274 = tail call i8* @malloc(i32 mul (i32
475     ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
476     i32 3))
477     %array275 = bitcast i8* %malloccall274 to i32*
478     %prefield_elem276 = getelementptr i32, i32* %array275, i32
479     0
480     store i32 0, i32* %prefield_elem276
481     %scaledegreer_elem277 = getelementptr i32, i32* %array275,
482     i32 1
483     store i32 6, i32* %scaledegreer_elem277
484     %postfield_elem278 = getelementptr i32, i32* %array275, i32
485     2
486     store i32 0, i32* %postfield_elem278
487     store i32* %array275, i32** %pitch_pointer_elem273
488     %struct_c_pointer279 = getelementptr inbounds
489     %chord_struct, %chord_struct* %chord_struct269, i32 0, i32 1
490     store i32** %arr_pitch272, i32*** %struct_c_pointer279
491     %actual_chord_struct280 = load %chord_struct,
492     %chord_struct* %chord_struct269
493     store %chord_struct %actual_chord_struct280, %chord_struct*
494     %pointer_chord_elem_list267
495     %pointer_chord_elem_list281 = getelementptr %chord_struct,

```

```

    %chord_struct* %chord_pointer_array, i32 19
479    %malloccall282 = tail call i8* @malloc(i32 ptrtoint
        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
        null, i32 1) to i32))
480    %chord_struct283 = bitcast i8* %malloccall282 to
        %chord_struct*
481    %length284 = getelementptr inbounds %chord_struct,
        %chord_struct* %chord_struct283, i32 0, i32 0
482    store i32 1, i32* %length284
483    %malloccall285 = tail call i8* @malloc(i32 ptrtoint (i1***
        getelementptr (i1*, i1** null, i32 1) to i32))
484    %arr_pitch286 = bitcast i8* %malloccall285 to i32**
485    %pitch_pointer_elem287 = getelementptr i32*, i32**%
        %arr_pitch286, i32 0
486    %malloccall288 = tail call i8* @malloc(i32 mul (i32
        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
        i32 3))
487    %array289 = bitcast i8* %malloccall288 to i32*
488    %prefield_elem290 = getelementptr i32, i32* %array289, i32
        0
489    store i32 0, i32* %prefield_elem290
490    %scaledegreer_elem291 = getelementptr i32, i32* %array289,
        i32 1
491    store i32 3, i32* %scaledegreer_elem291
492    %postfield_elem292 = getelementptr i32, i32* %array289, i32
        2
493    store i32 0, i32* %postfield_elem292
494    store i32* %array289, i32** %pitch_pointer_elem287
495    %struct_c_pointer293 = getelementptr inbounds
        %chord_struct, %chord_struct* %chord_struct283, i32 0, i32 1
496    store i32** %arr_pitch286, i32*** %struct_c_pointer293
497    %actual_chord_struct294 = load %chord_struct,
        %chord_struct* %chord_struct283
498    store %chord_struct %actual_chord_struct294, %chord_struct*
        %pointer_chord_elem_list281
499    %pointer_chord_elem_list295 = getelementptr %chord_struct,
        %chord_struct* %chord_pointer_array, i32 20
500    %malloccall296 = tail call i8* @malloc(i32 ptrtoint
        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
        null, i32 1) to i32))
501    %chord_struct297 = bitcast i8* %malloccall296 to
        %chord_struct*

```

```

502    %length298 = getelementptr inbounds %chord_struct,
503    %chord_struct* %chord_struct297, i32 0, i32 0
504    store i32 1, i32* %length298
505    %malloccall299 = tail call i8* @malloc(i32 ptrtoint (i1**
506        getelementptr (i1*, i1** null, i32 1) to i32))
507    %arr_pitch300 = bitcast i8* %malloccall299 to i32**
508    %pitch_pointer_elem301 = getelementptr i32*, i32**
509    %arr_pitch300, i32 0
510    %malloccall302 = tail call i8* @malloc(i32 mul (i32
511        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
512        i32 3))
513    %array303 = bitcast i8* %malloccall302 to i32*
514    %prefield_elem304 = getelementptr i32, i32* %array303, i32
515    0
516    store i32 0, i32* %prefield_elem304
517    %scaledegreer_elem305 = getelementptr i32, i32* %array303,
518    i32 1
519    store i32 2, i32* %scaledegreer_elem305
520    %postfield_elem306 = getelementptr i32, i32* %array303, i32
521    2
522    store i32 0, i32* %postfield_elem306
523    store i32* %array303, i32** %pitch_pointer_elem301
524    %struct_c_pointer307 = getelementptr inbounds
525    %chord_struct, %chord_struct* %chord_struct297, i32 0, i32 1
526    store i32** %arr_pitch300, i32*** %struct_c_pointer307
527    %actual_chord_struct308 = load %chord_struct,
528    %chord_struct* %chord_struct297
529    store %chord_struct %actual_chord_struct308, %chord_struct*
530    %pointer_chord_elem_list295
531    %pointer_chord_elem_list309 = getelementptr %chord_struct,
532    %chord_struct* %chord_pointer_array, i32 21
533    %malloccall310 = tail call i8* @malloc(i32 ptrtoint
534        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
535        null, i32 1) to i32))
536    %chord_struct311 = bitcast i8* %malloccall310 to
537    %chord_struct*
538    %length312 = getelementptr inbounds %chord_struct,
539    %chord_struct* %chord_struct311, i32 0, i32 0
540    store i32 1, i32* %length312
541    %malloccall313 = tail call i8* @malloc(i32 ptrtoint (i1**
542        getelementptr (i1*, i1** null, i32 1) to i32))
543    %arr_pitch314 = bitcast i8* %malloccall313 to i32**

```

```

527    %pitch_pointer_elem315 = getelementptr i32*, i32**
528        %arr_pitch314, i32 0
529    %malloccall316 = tail call i8* @malloc(i32 mul (i32
530        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
531        i32 3))
529    %array317 = bitcast i8* %malloccall316 to i32*
530    %prefield_elem318 = getelementptr i32, i32* %array317, i32
531        0
531    store i32 0, i32* %prefield_elem318
532    %scaledegreer_elem319 = getelementptr i32, i32* %array317,
533        i32 1
533    store i32 3, i32* %scaledegreer_elem319
534    %postfield_elem320 = getelementptr i32, i32* %array317, i32
535        2
535    store i32 0, i32* %postfield_elem320
536    store i32* %array317, i32** %pitch_pointer_elem315
537    %struct_c_pointer321 = getelementptr inbounds
538        %chord_struct, %chord_struct* %chord_struct311, i32 0, i32 1
538    store i32** %arr_pitch314, i32*** %struct_c_pointer321
539    %actual_chord_struct322 = load %chord_struct,
540        %chord_struct* %chord_struct311
540    store %chord_struct %actual_chord_struct322, %chord_struct*
541        %pointer_chord_elem_list309
541    %pointer_chord_elem_list323 = getelementptr %chord_struct,
542        %chord_struct* %chord_pointer_array, i32 22
542    %malloccall324 = tail call i8* @malloc(i32 ptrtoint
543        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
544        null, i32 1) to i32))
543    %chord_struct325 = bitcast i8* %malloccall324 to
544        %chord_struct*
544    %length326 = getelementptr inbounds %chord_struct,
545        %chord_struct* %chord_struct325, i32 0, i32 0
545    store i32 1, i32* %length326
546    %malloccall327 = tail call i8* @malloc(i32 ptrtoint (i1**
547        getelementptr (i1*, i1** null, i32 1) to i32))
547    %arr_pitch328 = bitcast i8* %malloccall327 to i32**
548    %pitch_pointer_elem329 = getelementptr i32*, i32**
549        %arr_pitch328, i32 0
549    %malloccall330 = tail call i8* @malloc(i32 mul (i32
550        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
551        i32 3))
550    %array331 = bitcast i8* %malloccall330 to i32*

```

```

551  %prefield_elem332 = getelementptr i32, i32* %array331, i32
      0
552  store i32 0, i32* %prefield_elem332
553  %scaledegreer_elem333 = getelementptr i32, i32* %array331,
      i32 1
554  store i32 4, i32* %scaledegreer_elem333
555  %postfield_elem334 = getelementptr i32, i32* %array331, i32
      2
556  store i32 1, i32* %postfield_elem334
557  store i32* %array331, i32** %pitch_pointer_elem329
558  %struct_c_pointer335 = getelementptr inbounds
      %chord_struct, %chord_struct* %chord_struct325, i32 0, i32 1
559  store i32** %arr_pitch328, i32*** %struct_c_pointer335
560  %actual_chord_struct336 = load %chord_struct,
      %chord_struct* %chord_struct325
561  store %chord_struct %actual_chord_struct336, %chord_struct*
      %pointer_chord_elem_list323
562  %pointer_chord_elem_list337 = getelementptr %chord_struct,
      %chord_struct* %chord_pointer_array, i32 23
563  %malloccall338 = tail call i8* @malloc(i32 ptrtoint
      (%chord_struct* getelementptr (%chord_struct, %chord_struct*
      null, i32 1) to i32))
564  %chord_struct339 = bitcast i8* %malloccall338 to
      %chord_struct*
565  %length340 = getelementptr inbounds %chord_struct,
      %chord_struct* %chord_struct339, i32 0, i32 0
566  store i32 1, i32* %length340
567  %malloccall341 = tail call i8* @malloc(i32 ptrtoint (i1**
      getelementptr (i1*, i1** null, i32 1) to i32))
568  %arr_pitch342 = bitcast i8* %malloccall341 to i32**
569  %pitch_pointer_elem343 = getelementptr i32*, i32**
      %arr_pitch342, i32 0
570  %malloccall344 = tail call i8* @malloc(i32 mul (i32
      ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
      i32 3))
571  %array345 = bitcast i8* %malloccall344 to i32*
572  %prefield_elem346 = getelementptr i32, i32* %array345, i32
      0
573  store i32 0, i32* %prefield_elem346
574  %scaledegreer_elem347 = getelementptr i32, i32* %array345,
      i32 1
575  store i32 2, i32* %scaledegreer_elem347

```

```

576  %postfield_elem348 = getelementptr i32, i32* %array345, i32
2
577  store i32 0, i32* %postfield_elem348
578  store i32* %array345, i32** %pitch_pointer_elem343
579  %struct_c_pointer349 = getelementptr inbounds
    %chord_struct, %chord_struct* %chord_struct339, i32 0, i32 1
580  store i32** %arr_pitch342, i32*** %struct_c_pointer349
581  %actual_chord_struct350 = load %chord_struct,
    %chord_struct* %chord_struct339
582  store %chord_struct %actual_chord_struct350, %chord_struct*
    %pointer_chord_elem_list337
583  %pointer_chord_elem_list351 = getelementptr %chord_struct,
    %chord_struct* %chord_pointer_array, i32 24
584  %malloccall1352 = tail call i8* @malloc(i32 ptrtoint
    (%chord_struct* getelementptr (%chord_struct, %chord_struct*
        null, i32 1) to i32))
585  %chord_struct353 = bitcast i8* %malloccall1352 to
    %chord_struct*
586  %length354 = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct353, i32 0, i32 0
587  store i32 1, i32* %length354
588  %malloccall1355 = tail call i8* @malloc(i32 ptrtoint (i1**
    getelementptr (i1*, i1** null, i32 1) to i32))
589  %arr_pitch356 = bitcast i8* %malloccall1355 to i32**
590  %pitch_pointer_elem357 = getelementptr i32*, i32**
    %arr_pitch356, i32 0
591  %malloccall1358 = tail call i8* @malloc(i32 mul (i32
    ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
    i32 3))
592  %array359 = bitcast i8* %malloccall1358 to i32*
593  %prefield_elem360 = getelementptr i32, i32* %array359, i32
0
594  store i32 0, i32* %prefield_elem360
595  %scaledegreer_elem361 = getelementptr i32, i32* %array359,
    i32 1
596  store i32 1, i32* %scaledegreer_elem361
597  %postfield_elem362 = getelementptr i32, i32* %array359, i32
2
598  store i32 0, i32* %postfield_elem362
599  store i32* %array359, i32** %pitch_pointer_elem357
600  %struct_c_pointer363 = getelementptr inbounds
    %chord_struct, %chord_struct* %chord_struct353, i32 0, i32 1

```

```

601   store i32** %arr_pitch356, i32*** %struct_c_pointer363
602   %actual_chord_struct364 = load %chord_struct,
603   %chord_struct* %chord_struct353
604   store %chord_struct %actual_chord_struct364, %chord_struct*
605   %pointer_chord_elem_list351
606   %pointer_chord_elem_list365 = getelementptr %chord_struct,
607   %chord_struct* %chord_pointer_array, i32 25
608   %malloccall366 = tail call i8* @malloc(i32 ptrtoint
609   (%chord_struct* getelementptr (%chord_struct, %chord_struct*
610   null, i32 1) to i32))
611   %chord_struct367 = bitcast i8* %malloccall366 to
612   %chord_struct*
613   %length368 = getelementptr inbounds %chord_struct,
614   %chord_struct* %chord_struct367, i32 0, i32 0
615   store i32 1, i32* %length368
616   %malloccall369 = tail call i8* @malloc(i32 ptrtoint (i1**
617   getelementptr (i1*, i1** null, i32 1) to i32))
618   %arr_pitch370 = bitcast i8* %malloccall369 to i32**
619   %pitch_pointer_elem371 = getelementptr i32*, i32**
620   %arr_pitch370, i32 0
621   %malloccall372 = tail call i8* @malloc(i32 mul (i32
622   ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
623   i32 3))
624   %array373 = bitcast i8* %malloccall372 to i32*
625   %prefield_elem374 = getelementptr i32, i32* %array373, i32
626   0
627   store i32 -1, i32* %prefield_elem374
628   %scaledegreer_elem375 = getelementptr i32, i32* %array373,
629   i32 1
630   store i32 7, i32* %scaledegreer_elem375
631   %postfield_elem376 = getelementptr i32, i32* %array373, i32
632   2
633   store i32 -1, i32* %postfield_elem376
634   store i32* %array373, i32** %pitch_pointer_elem371
635   %struct_c_pointer377 = getelementptr inbounds
636   %chord_struct, %chord_struct* %chord_struct367, i32 0, i32 1
637   store i32** %arr_pitch370, i32*** %struct_c_pointer377
638   %actual_chord_struct378 = load %chord_struct,
639   %chord_struct* %chord_struct367
640   store %chord_struct %actual_chord_struct378, %chord_struct*
641   %pointer_chord_elem_list365
642   %pointer_chord_elem_list379 = getelementptr %chord_struct,

```

```

%chord_struct* %chord_pointer_array, i32 26
626    %malloccall380 = tail call i8* @malloc(i32 ptrtoint
        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
        null, i32 1) to i32))
627    %chord_struct381 = bitcast i8* %malloccall380 to
        %chord_struct*
628    %length382 = getelementptr inbounds %chord_struct,
        %chord_struct* %chord_struct381, i32 0, i32 0
629    store i32 1, i32* %length382
630    %malloccall383 = tail call i8* @malloc(i32 ptrtoint (i1**
        getelementptr (i1*, i1** null, i32 1) to i32))
631    %arr_pitch384 = bitcast i8* %malloccall383 to i32**
632    %pitch_pointer_elem385 = getelementptr i32*, i32**
        %arr_pitch384, i32 0
633    %malloccall386 = tail call i8* @malloc(i32 mul (i32
        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
        i32 3))
634    %array387 = bitcast i8* %malloccall386 to i32*
635    %prefield_elem388 = getelementptr i32, i32* %array387, i32
        0
636    store i32 0, i32* %prefield_elem388
637    %scaledegreer_elem389 = getelementptr i32, i32* %array387,
        i32 1
638    store i32 5, i32* %scaledegreer_elem389
639    %postfield_elem390 = getelementptr i32, i32* %array387, i32
        2
640    store i32 0, i32* %postfield_elem390
641    store i32* %array387, i32** %pitch_pointer_elem385
642    %struct_c_pointer391 = getelementptr inbounds
        %chord_struct, %chord_struct* %chord_struct381, i32 0, i32 1
643    store i32** %arr_pitch384, i32*** %struct_c_pointer391
644    %actual_chord_struct392 = load %chord_struct,
        %chord_struct* %chord_struct381
645    store %chord_struct %actual_chord_struct392, %chord_struct*
        %pointer_chord_elem_list379
646    %struct_cl_pointer = getelementptr inbounds
        %chordlist_struct, %chordlist_struct* %cl_struct, i32 0, i32
        1
647    store %chord_struct* %chord_pointer_array, %chord_struct**
        %struct_cl_pointer
648    %theme = alloca %chordlist_struct*
649    store %chordlist_struct* %cl_struct, %chordlist_struct**

```

```

%theme
650  %array_struct393 = alloca %list_struct_f
651  %length394 = getelementptr inbounds %list_struct_f,
652    %list_struct_f* %array_struct393, i32 0, i32 0
653  store i32 27, i32* %length394
654  %array395 = alloca double, i32 27
655  %elem396 = getelementptr double, double* %array395, i32 0
656  store double 0.00000e+00, double* %elem396
657  %elem397 = getelementptr double, double* %array395, i32 1
658  store double 1.00000e-01, double* %elem397
659  %elem398 = getelementptr double, double* %array395, i32 2
660  store double 1.00000e-01, double* %elem398
661  %elem399 = getelementptr double, double* %array395, i32 3
662  store double 1.00000e-01, double* %elem399
663  %elem400 = getelementptr double, double* %array395, i32 4
664  store double 1.00000e-01, double* %elem400
665  %elem401 = getelementptr double, double* %array395, i32 5
666  store double 1.00000e-01, double* %elem401
667  %elem402 = getelementptr double, double* %array395, i32 6
668  store double 1.00000e-01, double* %elem402
669  %elem403 = getelementptr double, double* %array395, i32 7
670  store double 1.00000e-01, double* %elem403
671  %elem404 = getelementptr double, double* %array395, i32 8
672  store double 1.00000e-01, double* %elem404
673  %elem405 = getelementptr double, double* %array395, i32 9
674  store double 1.00000e-01, double* %elem405
675  %elem406 = getelementptr double, double* %array395, i32 10
676  store double 1.00000e-01, double* %elem406
677  %elem407 = getelementptr double, double* %array395, i32 11
678  store double 1.00000e-01, double* %elem407
679  %elem408 = getelementptr double, double* %array395, i32 12
680  store double 1.00000e-01, double* %elem408
681  %elem409 = getelementptr double, double* %array395, i32 13
682  store double 1.00000e-01, double* %elem409
683  %elem410 = getelementptr double, double* %array395, i32 14
684  store double 1.00000e-01, double* %elem410
685  %elem411 = getelementptr double, double* %array395, i32 15
686  store double 1.00000e-01, double* %elem411
687  %elem412 = getelementptr double, double* %array395, i32 16
688  store double 1.00000e-01, double* %elem412
689  %elem413 = getelementptr double, double* %array395, i32 17
690  store double 1.00000e-01, double* %elem413

```

```

690    %elem414 = getelementptr double, double* %array395, i32 18
691    store double 1.00000e-01, double* %elem414
692    %elem415 = getelementptr double, double* %array395, i32 19
693    store double 1.00000e-01, double* %elem415
694    %elem416 = getelementptr double, double* %array395, i32 20
695    store double 1.00000e-01, double* %elem416
696    %elem417 = getelementptr double, double* %array395, i32 21
697    store double 1.00000e-01, double* %elem417
698    %elem418 = getelementptr double, double* %array395, i32 22
699    store double 1.00000e-01, double* %elem418
700    %elem419 = getelementptr double, double* %array395, i32 23
701    store double 1.00000e-01, double* %elem419
702    %elem420 = getelementptr double, double* %array395, i32 24
703    store double 1.00000e-01, double* %elem420
704    %elem421 = getelementptr double, double* %array395, i32 25
705    store double 2.00000e-01, double* %elem421
706    %elem422 = getelementptr double, double* %array395, i32 26
707    store double 2.00000e-01, double* %elem422
708    %actual_list423 = getelementptr inbounds %list_struct_f,
709    %list_struct_f* %array_struct393, i32 0, i32 1
710    store double* %array395, double** %actual_list423
711    %r1 = alloca %list_struct_f*
712    store %list_struct_f* %array_struct393, %list_struct_f**
713    %r1
714    %array_struct424 = alloca %list_struct_f
715    %length425 = getelementptr inbounds %list_struct_f,
716    %list_struct_f* %array_struct424, i32 0, i32 0
717    store i32 27, i32* %length425
718    %array426 = alloca double, i32 27
719    %elem427 = getelementptr double, double* %array426, i32 0
720    store double 2.50000e+00, double* %elem427
721    %elem428 = getelementptr double, double* %array426, i32 1
722    store double 1.00000e-01, double* %elem428
723    %elem429 = getelementptr double, double* %array426, i32 2
724    store double 1.00000e-01, double* %elem429
725    %elem430 = getelementptr double, double* %array426, i32 3
726    store double 1.00000e-01, double* %elem430
727    %elem431 = getelementptr double, double* %array426, i32 4
728    store double 1.00000e-01, double* %elem431
729    %elem432 = getelementptr double, double* %array426, i32 5
730    store double 1.00000e-01, double* %elem432
731    %elem433 = getelementptr double, double* %array426, i32 6

```

```

729 store double 1.00000e-01, double* %elem433
730 %elem434 = getelementptr double, double* %array426, i32 7
731 store double 1.00000e-01, double* %elem434
732 %elem435 = getelementptr double, double* %array426, i32 8
733 store double 1.00000e-01, double* %elem435
734 %elem436 = getelementptr double, double* %array426, i32 9
735 store double 1.00000e-01, double* %elem436
736 %elem437 = getelementptr double, double* %array426, i32 10
737 store double 1.00000e-01, double* %elem437
738 %elem438 = getelementptr double, double* %array426, i32 11
739 store double 1.00000e-01, double* %elem438
740 %elem439 = getelementptr double, double* %array426, i32 12
741 store double 1.00000e-01, double* %elem439
742 %elem440 = getelementptr double, double* %array426, i32 13
743 store double 1.00000e-01, double* %elem440
744 %elem441 = getelementptr double, double* %array426, i32 14
745 store double 1.00000e-01, double* %elem441
746 %elem442 = getelementptr double, double* %array426, i32 15
747 store double 1.00000e-01, double* %elem442
748 %elem443 = getelementptr double, double* %array426, i32 16
749 store double 1.00000e-01, double* %elem443
750 %elem444 = getelementptr double, double* %array426, i32 17
751 store double 1.00000e-01, double* %elem444
752 %elem445 = getelementptr double, double* %array426, i32 18
753 store double 1.00000e-01, double* %elem445
754 %elem446 = getelementptr double, double* %array426, i32 19
755 store double 1.00000e-01, double* %elem446
756 %elem447 = getelementptr double, double* %array426, i32 20
757 store double 1.00000e-01, double* %elem447
758 %elem448 = getelementptr double, double* %array426, i32 21
759 store double 1.00000e-01, double* %elem448
760 %elem449 = getelementptr double, double* %array426, i32 22
761 store double 1.00000e-01, double* %elem449
762 %elem450 = getelementptr double, double* %array426, i32 23
763 store double 1.00000e-01, double* %elem450
764 %elem451 = getelementptr double, double* %array426, i32 24
765 store double 1.00000e-01, double* %elem451
766 %elem452 = getelementptr double, double* %array426, i32 25
767 store double 2.00000e-01, double* %elem452
768 %elem453 = getelementptr double, double* %array426, i32 26
769 store double 2.00000e-01, double* %elem453
770 %actual_list454 = getelementptr inbounds %list_struct_f,

```

```

%list_struct_f* %array_struct424, i32 0, i32 1
771 store double* %array426, double** %actual_list454
772 %r2 = alloca %list_struct_f*
773 store %list_struct_f* %array_struct424, %list_struct_f**
    %r2
774 %malloccall455 = tail call i8* @malloc(i32 ptrtoint
    (%chordlist_struct* getelementptr (%chordlist_struct,
    %chordlist_struct* null, i32 1) to i32))
775 %cl_struct456 = bitcast i8* %malloccall455 to
    %chordlist_struct*
776 %length457 = getelementptr inbounds %chordlist_struct,
    %chordlist_struct* %cl_struct456, i32 0, i32 0
777 store i32 16, i32* %length457
778 %malloccall458 = tail call i8* @malloc(i32 mul (i32
    ptrtoint (%chord_struct* getelementptr (%chord_struct,
    %chord_struct* null, i32 1) to i32), i32 16))
779 %chord_pointer_array459 = bitcast i8* %malloccall458 to
    %chord_struct*
780 %pointer_chord_elem_list460 = getelementptr %chord_struct,
    %chord_struct* %chord_pointer_array459, i32 0
781 %malloccall461 = tail call i8* @malloc(i32 ptrtoint
    (%chord_struct* getelementptr (%chord_struct, %chord_struct*
    null, i32 1) to i32))
782 %chord_struct462 = bitcast i8* %malloccall461 to
    %chord_struct*
783 %length463 = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct462, i32 0, i32 0
784 store i32 1, i32* %length463
785 %malloccall464 = tail call i8* @malloc(i32 ptrtoint (i1**
    getelementptr (i1*, i1** null, i32 1) to i32))
786 %arr_pitch465 = bitcast i8* %malloccall464 to i32**
787 %pitch_pointer_elem466 = getelementptr i32*, i32**
    %arr_pitch465, i32 0
788 %malloccall467 = tail call i8* @malloc(i32 mul (i32
    ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
    i32 3))
789 %array468 = bitcast i8* %malloccall467 to i32*
790 %prefield_elem469 = getelementptr i32, i32* %array468, i32
    0
791 store i32 0, i32* %prefield_elem469
792 %scaleddegreeer_elem470 = getelementptr i32, i32* %array468,
    i32 1

```

```

793   store i32 0, i32* %scaledegreer_elem470
794   %postfield_elem471 = getelementptr i32, i32* %array468, i32
    2
795   store i32 0, i32* %postfield_elem471
796   store i32* %array468, i32** %pitch_pointer_elem466
797   %struct_c_pointer472 = getelementptr inbounds
     %chord_struct, %chord_struct* %chord_struct462, i32 0, i32 1
798   store i32** %arr_pitch465, i32*** %struct_c_pointer472
799   %actual_chord_struct473 = load %chord_struct,
     %chord_struct* %chord_struct462
800   store %chord_struct %actual_chord_struct473, %chord_struct*
     %pointer_chord_elem_list460
801   %pointer_chord_elem_list474 = getelementptr %chord_struct,
     %chord_struct* %chord_pointer_array459, i32 1
802   %malloccall475 = tail call i8* @malloc(i32 ptrtoint
     (%chord_struct* getelementptr (%chord_struct, %chord_struct*
     null, i32 1) to i32))
803   %chord_struct476 = bitcast i8* %malloccall475 to
     %chord_struct*
804   %length477 = getelementptr inbounds %chord_struct,
     %chord_struct* %chord_struct476, i32 0, i32 0
805   store i32 1, i32* %length477
806   %malloccall478 = tail call i8* @malloc(i32 ptrtoint (i1**
     getelementptr (i1*, i1** null, i32 1) to i32))
807   %arr_pitch479 = bitcast i8* %malloccall478 to i32**
808   %pitch_pointer_elem480 = getelementptr i32*, i32**
     %arr_pitch479, i32 0
809   %malloccall481 = tail call i8* @malloc(i32 mul (i32
     ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
     i32 3))
810   %array482 = bitcast i8* %malloccall481 to i32*
811   %prefield_elem483 = getelementptr i32, i32* %array482, i32
    0
812   store i32 0, i32* %prefield_elem483
813   %scaledegreer_elem484 = getelementptr i32, i32* %array482,
     i32 1
814   store i32 5, i32* %scaledegreer_elem484
815   %postfield_elem485 = getelementptr i32, i32* %array482, i32
    2
816   store i32 0, i32* %postfield_elem485
817   store i32* %array482, i32** %pitch_pointer_elem480
818   %struct_c_pointer486 = getelementptr inbounds

```

```

%chord_struct, %chord_struct* %chord_struct476, i32 0, i32 1
819   store i32** %arr_pitch479, i32*** %struct_c_pointer486
820   %actual_chord_struct487 = load %chord_struct,
     %chord_struct* %chord_struct476
821   store %chord_struct %actual_chord_struct487, %chord_struct*
     %pointer_chord_elem_list474
822   %pointer_chord_elem_list488 = getelementptr %chord_struct,
     %chord_struct* %chord_pointer_array459, i32 2
823   %malloccall489 = tail call i8* @malloc(i32 ptrtoint
     (%chord_struct* getelementptr (%chord_struct, %chord_struct*
     null, i32 1) to i32))
824   %chord_struct490 = bitcast i8* %malloccall489 to
     %chord_struct*
825   %length491 = getelementptr inbounds %chord_struct,
     %chord_struct* %chord_struct490, i32 0, i32 0
826   store i32 1, i32* %length491
827   %malloccall492 = tail call i8* @malloc(i32 ptrtoint (i1**
     getelementptr (i1*, i1** null, i32 1) to i32))
828   %arr_pitch493 = bitcast i8* %malloccall492 to i32**
829   %pitch_pointer_elem494 = getelementptr i32*, i32**
     %arr_pitch493, i32 0
830   %malloccall495 = tail call i8* @malloc(i32 mul (i32
     ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
     i32 3))
831   %array496 = bitcast i8* %malloccall495 to i32*
832   %prefield_elem497 = getelementptr i32, i32* %array496, i32
     0
833   store i32 0, i32* %prefield_elem497
834   %scaledegreer_elem498 = getelementptr i32, i32* %array496,
     i32 1
835   store i32 6, i32* %scaledegreer_elem498
836   %postfield_elem499 = getelementptr i32, i32* %array496, i32
     2
837   store i32 0, i32* %postfield_elem499
838   store i32* %array496, i32** %pitch_pointer_elem494
839   %struct_c_pointer500 = getelementptr inbounds
     %chord_struct, %chord_struct* %chord_struct490, i32 0, i32 1
840   store i32** %arr_pitch493, i32*** %struct_c_pointer500
841   %actual_chord_struct501 = load %chord_struct,
     %chord_struct* %chord_struct490
842   store %chord_struct %actual_chord_struct501, %chord_struct*
     %pointer_chord_elem_list488

```

```

843  %pointer_chord_elem_list502 = getelementptr %chord_struct,
844      %chord_struct* %chord_pointer_array459, i32 3
845  %malloccall503 = tail call i8* @malloc(i32 ptrtoint
846      (%chord_struct* getelementptr (%chord_struct, %chord_struct*
847          null, i32 1) to i32))
848  %chord_struct504 = bitcast i8* %malloccall503 to
849      %chord_struct*
850  %length505 = getelementptr inbounds %chord_struct,
851      %chord_struct* %chord_struct504, i32 0, i32 0
852  store i32 1, i32* %length505
853  %malloccall506 = tail call i8* @malloc(i32 ptrtoint (i1**
854      getelementptr (i1*, i1** null, i32 1) to i32))
855  %arr_pitch507 = bitcast i8* %malloccall506 to i32**
856  %pitch_pointer_elem508 = getelementptr i32*, i32**
857      %arr_pitch507, i32 0
858  %malloccall509 = tail call i8* @malloc(i32 mul (i32
859      ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
860      i32 3))
861  %array510 = bitcast i8* %malloccall509 to i32*
862  %prefield_elem511 = getelementptr i32, i32* %array510, i32
863  0
864  store i32 0, i32* %prefield_elem511
865  %scaledegreer_elem512 = getelementptr i32, i32* %array510,
866  i32 1
867  store i32 5, i32* %scaledegreer_elem512
868  %postfield_elem513 = getelementptr i32, i32* %array510, i32
869  2
870  store i32 0, i32* %postfield_elem513
871  store i32* %array510, i32** %pitch_pointer_elem508
872  %struct_c_pointer514 = getelementptr inbounds
873      %chord_struct, %chord_struct* %chord_struct504, i32 0, i32 1
874  store i32** %arr_pitch507, i32*** %struct_c_pointer514
875  %actual_chord_struct515 = load %chord_struct,
876      %chord_struct* %chord_struct504
877  store %chord_struct %actual_chord_struct515, %chord_struct*
878      %pointer_chord_elem_list502
879  %pointer_chord_elem_list516 = getelementptr %chord_struct,
880      %chord_struct* %chord_pointer_array459, i32 4
881  %malloccall517 = tail call i8* @malloc(i32 ptrtoint
882      (%chord_struct* getelementptr (%chord_struct, %chord_struct*
883          null, i32 1) to i32))
884  %chord_struct518 = bitcast i8* %malloccall517 to

```

```

%chord_struct*
867    %length519 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct518, i32 0, i32 0
868    store i32 1, i32* %length519
869    %malloccall520 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
870    %arr_pitch521 = bitcast i8* %malloccall520 to i32**
871    %pitch_pointer_elem522 = getelementptr i32*, i32**
%arr_pitch521, i32 0
872    %malloccall523 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
873    %array524 = bitcast i8* %malloccall523 to i32*
874    %prefield_elem525 = getelementptr i32, i32* %array524, i32
0
875    store i32 0, i32* %prefield_elem525
876    %scaleddegreer_elem526 = getelementptr i32, i32* %array524,
i32 1
877    store i32 4, i32* %scaleddegreer_elem526
878    %postfield_elem527 = getelementptr i32, i32* %array524, i32
2
879    store i32 0, i32* %postfield_elem527
880    store i32* %array524, i32** %pitch_pointer_elem522
881    %struct_c_pointer528 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct518, i32 0, i32 1
882    store i32** %arr_pitch521, i32*** %struct_c_pointer528
883    %actual_chord_struct529 = load %chord_struct,
%chord_struct* %chord_struct518
884    store %chord_struct %actual_chord_struct529, %chord_struct*
%pointer_chord_elem_list516
885    %pointer_chord_elem_list530 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array459, i32 5
886    %malloccall531 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
887    %chord_struct532 = bitcast i8* %malloccall531 to
%chord_struct*
888    %length533 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct532, i32 0, i32 0
889    store i32 1, i32* %length533
890    %malloccall534 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))

```

```

891    %arr_pitch535 = bitcast i8* %malloccall534 to i32**
892    %pitch_pointer_elem536 = getelementptr i32*, i32** %arr_pitch535, i32 0
893    %malloccall537 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
894    %array538 = bitcast i8* %malloccall537 to i32*
895    %prefield_elem539 = getelementptr i32, i32* %array538, i32 0
896    store i32 0, i32* %prefield_elem539
897    %scaledegreer_elem540 = getelementptr i32, i32* %array538, i32 1
898    store i32 3, i32* %scaledegreer_elem540
899    %postfield_elem541 = getelementptr i32, i32* %array538, i32 2
900    store i32 0, i32* %postfield_elem541
901    store i32* %array538, i32** %pitch_pointer_elem536
902    %struct_c_pointer542 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct532, i32 0, i32 1
903    store i32** %arr_pitch535, i32*** %struct_c_pointer542
904    %actual_chord_struct543 = load %chord_struct, %chord_struct* %chord_struct532
905    store %chord_struct %actual_chord_struct543, %chord_struct* %pointer_chord_elem_list530
906    %pointer_chord_elem_list544 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array459, i32 6
907    %malloccall545 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
908    %chord_struct546 = bitcast i8* %malloccall545 to %chord_struct*
909    %length547 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct546, i32 0, i32 0
910    store i32 1, i32* %length547
911    %malloccall548 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
912    %arr_pitch549 = bitcast i8* %malloccall548 to i32**
913    %pitch_pointer_elem550 = getelementptr i32*, i32** %arr_pitch549, i32 0
914    %malloccall551 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))

```

```

915    %array552 = bitcast i8* %malloccall551 to i32*
916    %prefield_elem553 = getelementptr i32, i32* %array552, i32
917    0
918    store i32 0, i32* %prefield_elem553
919    %scaleddegreer_elem554 = getelementptr i32, i32* %array552,
920    i32 1
921    store i32 2, i32* %scaleddegreer_elem554
922    %postfield_elem555 = getelementptr i32, i32* %array552, i32
923    2
924    store i32 0, i32* %postfield_elem555
925    store i32* %array552, i32** %pitch_pointer_elem550
926    %struct_c_pointer556 = getelementptr inbounds
927    %chord_struct, %chord_struct* %chord_struct546, i32 0, i32 1
928    store i32** %arr_pitch549, i32*** %struct_c_pointer556
929    %actual_chord_struct557 = load %chord_struct,
930    %chord_struct* %chord_struct546
931    store %chord_struct %actual_chord_struct557, %chord_struct*
932    %pointer_chord_elem_list544
933    %pointer_chord_elem_list558 = getelementptr %chord_struct,
934    %chord_struct* %chord_pointer_array459, i32 7
935    %malloccall559 = tail call i8* @malloc(i32 ptrtoint
936    (%chord_struct* getelementptr (%chord_struct, %chord_struct*
937    null, i32 1) to i32))
938    %chord_struct560 = bitcast i8* %malloccall559 to
939    %chord_struct*
940    %length561 = getelementptr inbounds %chord_struct,
941    %chord_struct* %chord_struct560, i32 0, i32 0
942    store i32 1, i32* %length561
943    %malloccall562 = tail call i8* @malloc(i32 ptrtoint (i1**
944    getelementptr (i1*, i1** null, i32 1) to i32))
945    %arr_pitch563 = bitcast i8* %malloccall562 to i32**
946    %pitch_pointer_elem564 = getelementptr i32*, i32**
947    %arr_pitch563, i32 0
948    %malloccall565 = tail call i8* @malloc(i32 mul (i32
949    ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
950    i32 3))
951    %array566 = bitcast i8* %malloccall565 to i32*
952    %prefield_elem567 = getelementptr i32, i32* %array566, i32
953    0
954    store i32 0, i32* %prefield_elem567
955    %scaleddegreer_elem568 = getelementptr i32, i32* %array566,
956    i32 1

```

```

940   store i32 4, i32* %scaledegreer_elem568
941   %postfield_elem569 = getelementptr i32, i32* %array566, i32
942   2
943   store i32 0, i32* %postfield_elem569
944   store i32* %array566, i32** %pitch_pointer_elem564
945   %struct_c_pointer570 = getelementptr inbounds
946   %chord_struct, %chord_struct* %chord_struct560, i32 0, i32 1
947   store i32** %arr_pitch563, i32*** %struct_c_pointer570
948   %actual_chord_struct571 = load %chord_struct,
949   %chord_struct* %chord_struct560
950   store %chord_struct %actual_chord_struct571, %chord_struct*
951   %pointer_chord_elem_list558
952   %pointer_chord_elem_list572 = getelementptr %chord_struct,
953   %chord_struct* %chord_pointer_array459, i32 8
954   %malloccall573 = tail call i8* @malloc(i32 ptrtoint
955   (%chord_struct* getelementptr (%chord_struct, %chord_struct*
956   null, i32 1) to i32))
957   %chord_struct574 = bitcast i8* %malloccall573 to
958   %chord_struct*
959   %length575 = getelementptr inbounds %chord_struct,
960   %chord_struct* %chord_struct574, i32 0, i32 0
961   store i32 1, i32* %length575
962   %malloccall576 = tail call i8* @malloc(i32 ptrtoint (i1**
963   getelementptr (i1*, i1** null, i32 1) to i32))
964   %arr_pitch577 = bitcast i8* %malloccall576 to i32**
965   %pitch_pointer_elem578 = getelementptr i32*, i32**
966   %arr_pitch577, i32 0
967   %malloccall579 = tail call i8* @malloc(i32 mul (i32
968   ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
969   i32 3))
970   %array580 = bitcast i8* %malloccall579 to i32*
971   %prefield_elem581 = getelementptr i32, i32* %array580, i32
972   0
973   store i32 0, i32* %prefield_elem581
974   %scaledegreer_elem582 = getelementptr i32, i32* %array580,
975   i32 1
976   store i32 3, i32* %scaledegreer_elem582
977   %postfield_elem583 = getelementptr i32, i32* %array580, i32
978   2
979   store i32 0, i32* %postfield_elem583
980   store i32* %array580, i32** %pitch_pointer_elem578
981   %struct_c_pointer584 = getelementptr inbounds

```

```

%chord_struct, %chord_struct* %chord_struct574, i32 0, i32 1
966 store i32** %arr_pitch577, i32*** %struct_c_pointer584
967 %actual_chord_struct585 = load %chord_struct,
%chord_struct* %chord_struct574
968 store %chord_struct %actual_chord_struct585, %chord_struct*
%pointer_chord_elem_list572
969 %pointer_chord_elem_list586 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array459, i32 9
970 %malloccall587 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
971 %chord_struct588 = bitcast i8* %malloccall587 to
%chord_struct*
972 %length589 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct588, i32 0, i32 0
973 store i32 1, i32* %length589
974 %malloccall590 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
975 %arr_pitch591 = bitcast i8* %malloccall590 to i32**
976 %pitch_pointer_elem592 = getelementptr i32*, i32**
%arr_pitch591, i32 0
977 %malloccall593 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
978 %array594 = bitcast i8* %malloccall593 to i32*
979 %prefield_elem595 = getelementptr i32, i32* %array594, i32
0
980 store i32 0, i32* %prefield_elem595
981 %scaledegreer_elem596 = getelementptr i32, i32* %array594,
i32 1
982 store i32 2, i32* %scaledegreer_elem596
983 %postfield_elem597 = getelementptr i32, i32* %array594, i32
2
984 store i32 0, i32* %postfield_elem597
985 store i32* %array594, i32** %pitch_pointer_elem592
986 %struct_c_pointer598 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct588, i32 0, i32 1
987 store i32** %arr_pitch591, i32*** %struct_c_pointer598
988 %actual_chord_struct599 = load %chord_struct,
%chord_struct* %chord_struct588
989 store %chord_struct %actual_chord_struct599, %chord_struct*
%pointer_chord_elem_list586

```

```

990  %pointer_chord_elem_list600 = getelementptr %chord_struct,
991  %chord_struct* %chord_pointer_array459, i32 10
992  %malloccall601 = tail call i8* @malloc(i32 ptrtoint
993  (%chord_struct* getelementptr (%chord_struct, %chord_struct*
994  null, i32 1) to i32))
995  %chord_struct602 = bitcast i8* %malloccall601 to
996  %chord_struct*
997  %length603 = getelementptr inbounds %chord_struct,
998  %chord_struct* %chord_struct602, i32 0, i32 0
999  store i32 1, i32* %length603
1000  %malloccall604 = tail call i8* @malloc(i32 ptrtoint (i1**
1001  getelementptr (i1*, i1** null, i32 1) to i32))
1002  %arr_pitch605 = bitcast i8* %malloccall604 to i32**
1003  %pitch_pointer_elem606 = getelementptr i32*, i32**
1004  %arr_pitch605, i32 0
1005  %malloccall607 = tail call i8* @malloc(i32 mul (i32
1006  ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
1007  i32 3))
1008  %array608 = bitcast i8* %malloccall607 to i32*
1009  %prefield_elem609 = getelementptr i32, i32* %array608, i32
1010  0
1011  store i32 0, i32* %prefield_elem609
1012  %scaledegreer_elem610 = getelementptr i32, i32* %array608,
1013  i32 1
1014  store i32 1, i32* %scaledegreer_elem610
1015  %postfield_elem611 = getelementptr i32, i32* %array608, i32
1016  2
1017  store i32 0, i32* %postfield_elem611
1018  store i32* %array608, i32** %pitch_pointer_elem606
1019  %struct_c_pointer612 = getelementptr inbounds
1020  %chord_struct, %chord_struct* %chord_struct602, i32 0, i32 1
1021  store i32** %arr_pitch605, i32*** %struct_c_pointer612
1022  %actual_chord_struct613 = load %chord_struct,
1023  %chord_struct* %chord_struct602
1024  store %chord_struct %actual_chord_struct613, %chord_struct*
1025  %pointer_chord_elem_list600
1026  %pointer_chord_elem_list614 = getelementptr %chord_struct,
1027  %chord_struct* %chord_pointer_array459, i32 11
1028  %malloccall615 = tail call i8* @malloc(i32 ptrtoint
1029  (%chord_struct* getelementptr (%chord_struct, %chord_struct*
1030  null, i32 1) to i32))
1031  %chord_struct616 = bitcast i8* %malloccall615 to

```

```

%chord_struct*
1014    %length617 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct616, i32 0, i32 0
1015    store i32 1, i32* %length617
1016    %malloccall618 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1017    %arr_pitch619 = bitcast i8* %malloccall618 to i32**
1018    %pitch_pointer_elem620 = getelementptr i32*, i32**
%arr_pitch619, i32 0
1019    %malloccall621 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1020    %array622 = bitcast i8* %malloccall621 to i32*
1021    %prefield_elem623 = getelementptr i32, i32* %array622, i32
0
1022    store i32 -1, i32* %prefield_elem623
1023    %scaleddegreer_elem624 = getelementptr i32, i32* %array622,
i32 1
1024    store i32 7, i32* %scaleddegreer_elem624
1025    %postfield_elem625 = getelementptr i32, i32* %array622, i32
2
1026    store i32 0, i32* %postfield_elem625
1027    store i32* %array622, i32** %pitch_pointer_elem620
1028    %struct_c_pointer626 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct616, i32 0, i32 1
1029    store i32** %arr_pitch619, i32*** %struct_c_pointer626
1030    %actual_chord_struct627 = load %chord_struct,
%chord_struct* %chord_struct616
1031    store %chord_struct %actual_chord_struct627, %chord_struct*
%pointer_chord_elem_list614
1032    %pointer_chord_elem_list628 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array459, i32 12
1033    %malloccall629 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1034    %chord_struct630 = bitcast i8* %malloccall629 to
%chord_struct*
1035    %length631 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct630, i32 0, i32 0
1036    store i32 1, i32* %length631
1037    %malloccall632 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))

```

```

1038    %arr_pitch633 = bitcast i8* %malloccall632 to i32**
1039    %pitch_pointer_elem634 = getelementptr i32*, i32**  

%arr_pitch633, i32 0
1040    %malloccall635 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1041    %array636 = bitcast i8* %malloccall635 to i32*
1042    %prefield_elem637 = getelementptr i32, i32* %array636, i32
0
1043    store i32 0, i32* %prefield_elem637
1044    %scaledegreer_elem638 = getelementptr i32, i32* %array636,
i32 1
1045    store i32 1, i32* %scaledegreer_elem638
1046    %postfield_elem639 = getelementptr i32, i32* %array636, i32
2
1047    store i32 0, i32* %postfield_elem639
1048    store i32* %array636, i32** %pitch_pointer_elem634
1049    %struct_c_pointer640 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct630, i32 0, i32 1
1050    store i32** %arr_pitch633, i32*** %struct_c_pointer640
1051    %actual_chord_struct641 = load %chord_struct,
%chord_struct* %chord_struct630
1052    store %chord_struct %actual_chord_struct641, %chord_struct*
%pointer_chord_elem_list628
1053    %pointer_chord_elem_list642 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array459, i32 13
1054    %malloccall643 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1055    %chord_struct644 = bitcast i8* %malloccall643 to
%chord_struct*
1056    %length645 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct644, i32 0, i32 0
1057    store i32 1, i32* %length645
1058    %malloccall646 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1059    %arr_pitch647 = bitcast i8* %malloccall646 to i32**
1060    %pitch_pointer_elem648 = getelementptr i32*, i32**  

%arr_pitch647, i32 0
1061    %malloccall649 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))

```

```

1062    %array650 = bitcast i8* %malloccall649 to i32*
1063    %prefield_elem651 = getelementptr i32, i32* %array650, i32
1064        0
1065        store i32 0, i32* %prefield_elem651
1066        %scaleddegreer_elem652 = getelementptr i32, i32* %array650,
1067            i32 1
1068        store i32 2, i32* %scaleddegreer_elem652
1069        %postfield_elem653 = getelementptr i32, i32* %array650, i32
1070            2
1071        store i32 0, i32* %postfield_elem653
1072        store i32* %array650, i32** %pitch_pointer_elem648
1073        %struct_c_pointer654 = getelementptr inbounds
1074            %chord_struct, %chord_struct* %chord_struct644, i32 0, i32 1
1075        store i32** %arr_pitch647, i32*** %struct_c_pointer654
1076        %actual_chord_struct655 = load %chord_struct,
1077            %chord_struct* %chord_struct644
1078        store %chord_struct %actual_chord_struct655, %chord_struct*
1079            %pointer_chord_elem_list642
1080        %pointer_chord_elem_list656 = getelementptr %chord_struct,
1081            %chord_struct* %chord_pointer_array459, i32 14
1082        %malloccall657 = tail call i8* @malloc(i32 ptrtoint
1083            (%chord_struct* getelementptr (%chord_struct, %chord_struct*
1084                null, i32 1) to i32))
1085        %chord_struct658 = bitcast i8* %malloccall657 to
1086            %chord_struct*
1087        %length659 = getelementptr inbounds %chord_struct,
1088            %chord_struct* %chord_struct658, i32 0, i32 0
1089        store i32 1, i32* %length659
1090        %malloccall660 = tail call i8* @malloc(i32 ptrtoint (i1**
1091            getelementptr (i1*, i1** null, i32 1) to i32))
1092        %arr_pitch661 = bitcast i8* %malloccall660 to i32**
1093        %pitch_pointer_elem662 = getelementptr i32*, i32**
1094            %arr_pitch661, i32 0
1095        %malloccall663 = tail call i8* @malloc(i32 mul (i32
1096            ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
1097            i32 3))
1098        %array664 = bitcast i8* %malloccall663 to i32*
1099        %prefield_elem665 = getelementptr i32, i32* %array664, i32
1100            0
1101        store i32 -1, i32* %prefield_elem665
1102        %scaleddegreer_elem666 = getelementptr i32, i32* %array664,
1103            i32 1

```

```

1087    store i32 7, i32* %scaledegreer_elem666
1088    %postfield_elem667 = getelementptr i32, i32* %array664, i32
1089    2
1090    store i32 0, i32* %postfield_elem667
1091    store i32* %array664, i32** %pitch_pointer_elem662
1092    %struct_c_pointer668 = getelementptr inbounds
1093    %chord_struct, %chord_struct* %chord_struct658, i32 0, i32 1
1094    store i32** %arr_pitch661, i32*** %struct_c_pointer668
1095    %actual_chord_struct669 = load %chord_struct,
1096    %chord_struct* %chord_struct658
1097    store %chord_struct %actual_chord_struct669, %chord_struct*
1098    %pointer_chord_elem_list656
1099    %pointer_chord_elem_list670 = getelementptr %chord_struct,
1100    %chord_struct* %chord_pointer_array459, i32 15
1101    %malloccall671 = tail call i8* @malloc(i32 ptrtoint
1102    (%chord_struct* getelementptr (%chord_struct, %chord_struct*
1103    null, i32 1) to i32))
1104    %chord_struct672 = bitcast i8* %malloccall671 to
1105    %chord_struct*
1106    %length673 = getelementptr inbounds %chord_struct,
1107    %chord_struct* %chord_struct672, i32 0, i32 0
1108    store i32 1, i32* %length673
1109    %malloccall674 = tail call i8* @malloc(i32 ptrtoint (i1**
1110    getelementptr (i1*, i1** null, i32 1) to i32))
1111    %arr_pitch675 = bitcast i8* %malloccall674 to i32**
1112    %pitch_pointer_elem676 = getelementptr i32*, i32**
1113    %arr_pitch675, i32 0
1114    %malloccall677 = tail call i8* @malloc(i32 mul (i32
1115    ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
1116    i32 3))
1117    %array678 = bitcast i8* %malloccall677 to i32*
1118    %prefield_elem679 = getelementptr i32, i32* %array678, i32
1119    0
1120    store i32 0, i32* %prefield_elem679
1121    %scaledegreer_elem680 = getelementptr i32, i32* %array678,
1122    i32 1
1123    store i32 1, i32* %scaledegreer_elem680
1124    %postfield_elem681 = getelementptr i32, i32* %array678, i32
1125    2
1126    store i32 0, i32* %postfield_elem681
1127    store i32* %array678, i32** %pitch_pointer_elem676
1128    %struct_c_pointer682 = getelementptr inbounds

```

```

%chord_struct, %chord_struct* %chord_struct672, i32 0, i32 1
1113 store i32** %arr_pitch675, i32*** %struct_c_pointer682
1114 %actual_chord_struct683 = load %chord_struct,
%chord_struct* %chord_struct672
1115 store %chord_struct %actual_chord_struct683, %chord_struct*
%pointer_chord_elem_list670
1116 %struct_cl_pointer684 = getelementptr inbounds
%chordlist_struct, %chordlist_struct* %cl_struct456, i32 0,
i32 1
1117 store %chord_struct* %chord_pointer_array459,
%chord_struct** %struct_cl_pointer684
1118 %counter = alloca %chordlist_struct*
1119 store %chordlist_struct* %cl_struct456, %chordlist_struct**%
counter
1120 %array_struct685 = alloca %list_struct_f
1121 %length686 = getelementptr inbounds %list_struct_f,
%list_struct_f* %array_struct685, i32 0, i32 0
1122 store i32 16, i32* %length686
1123 %array687 = alloca double, i32 16
1124 %elem688 = getelementptr double, double* %array687, i32 0
1125 store double 1.000000e-01, double* %elem688
1126 %elem689 = getelementptr double, double* %array687, i32 1
1127 store double 1.000000e-01, double* %elem689
1128 %elem690 = getelementptr double, double* %array687, i32 2
1129 store double 1.000000e-01, double* %elem690
1130 %elem691 = getelementptr double, double* %array687, i32 3
1131 store double 1.000000e-01, double* %elem691
1132 %elem692 = getelementptr double, double* %array687, i32 4
1133 store double 1.000000e-01, double* %elem692
1134 %elem693 = getelementptr double, double* %array687, i32 5
1135 store double 1.000000e-01, double* %elem693
1136 %elem694 = getelementptr double, double* %array687, i32 6
1137 store double 1.000000e-01, double* %elem694
1138 %elem695 = getelementptr double, double* %array687, i32 7
1139 store double 1.000000e-01, double* %elem695
1140 %elem696 = getelementptr double, double* %array687, i32 8
1141 store double 5.000000e-01, double* %elem696
1142 %elem697 = getelementptr double, double* %array687, i32 9
1143 store double 1.000000e-01, double* %elem697
1144 %elem698 = getelementptr double, double* %array687, i32 10
1145 store double 1.000000e-01, double* %elem698
1146 %elem699 = getelementptr double, double* %array687, i32 11

```

```

1147 store double 1.000000e-01, double* %elem699
1148 %elem700 = getelementptr double, double* %array687, i32 12
1149 store double 1.000000e-01, double* %elem700
1150 %elem701 = getelementptr double, double* %array687, i32 13
1151 store double 1.000000e-01, double* %elem701
1152 %elem702 = getelementptr double, double* %array687, i32 14
1153 store double 1.000000e-01, double* %elem702
1154 %elem703 = getelementptr double, double* %array687, i32 15
1155 store double 1.000000e-01, double* %elem703
1156 %actual_list704 = getelementptr inbounds %list_struct_f,
%list_struct_f* %array_struct685, i32 0, i32 1
1157 store double* %array687, double** %actual_list704
1158 %r3 = alloca %list_struct_f*
1159 store %list_struct_f* %array_struct685, %list_struct_f**
%r3
1160 %malloccall705 = tail call i8* @malloc(i32 ptrtoint
(%chordlist_struct* getelementptr (%chordlist_struct,
%chordlist_struct* null, i32 1) to i32))
1161 %cl_struct706 = bitcast i8* %malloccall705 to
%chordlist_struct*
1162 %length707 = getelementptr inbounds %chordlist_struct,
%chordlist_struct* %cl_struct706, i32 0, i32 0
1163 store i32 21, i32* %length707
1164 %malloccall708 = tail call i8* @malloc(i32 mul (i32
ptrtoint (%chord_struct* getelementptr (%chord_struct,
%chord_struct* null, i32 1) to i32), i32 21))
1165 %chord_pointer_array709 = bitcast i8* %malloccall708 to
%chord_struct*
1166 %pointer_chord_elem_list710 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array709, i32 0
1167 %malloccall711 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1168 %chord_struct712 = bitcast i8* %malloccall711 to
%chord_struct*
1169 %length713 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct712, i32 0, i32 0
1170 store i32 1, i32* %length713
1171 %malloccall714 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1172 %arr_pitch715 = bitcast i8* %malloccall714 to i32**
1173 %pitch_pointer_elem716 = getelementptr i32*, i32**

```

```

%arr_pitch715, i32 0
1174    %malloccall717 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1175    %array718 = bitcast i8* %malloccall717 to i32*
1176    %prefield_elem719 = getelementptr i32, i32* %array718, i32
0
1177    store i32 0, i32* %prefield_elem719
1178    %scaleddegreer_elem720 = getelementptr i32, i32* %array718,
i32 1
1179    store i32 0, i32* %scaleddegreer_elem720
1180    %postfield_elem721 = getelementptr i32, i32* %array718, i32
2
1181    store i32 0, i32* %postfield_elem721
1182    store i32* %array718, i32** %pitch_pointer_elem716
1183    %struct_c_pointer722 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct712, i32 0, i32 1
1184    store i32** %arr_pitch715, i32*** %struct_c_pointer722
1185    %actual_chord_struct723 = load %chord_struct,
%chord_struct* %chord_struct712
1186    store %chord_struct %actual_chord_struct723, %chord_struct*
%pointer_chord_elem_list710
1187    %pointer_chord_elem_list724 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array709, i32 1
1188    %malloccall725 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1189    %chord_struct726 = bitcast i8* %malloccall725 to
%chord_struct*
1190    %length727 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct726, i32 0, i32 0
1191    store i32 1, i32* %length727
1192    %malloccall728 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1193    %arr_pitch729 = bitcast i8* %malloccall728 to i32**
1194    %pitch_pointer_elem730 = getelementptr i32*, i32**
%arr_pitch729, i32 0
1195    %malloccall731 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1196    %array732 = bitcast i8* %malloccall731 to i32*
1197    %prefield_elem733 = getelementptr i32, i32* %array732, i32

```

```

0
1198  store i32 -1, i32* %prefield_elem733
1199  %scaledegreer_elem734 = getelementptr i32, i32* %array732,
i32 1
1200  store i32 5, i32* %scaledegreer_elem734
1201  %postfield_elem735 = getelementptr i32, i32* %array732, i32
2
1202  store i32 0, i32* %postfield_elem735
1203  store i32* %array732, i32** %pitch_pointer_elem730
1204  %struct_c_pointer736 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct726, i32 0, i32 1
1205  store i32** %arr_pitch729, i32*** %struct_c_pointer736
1206  %actual_chord_struct737 = load %chord_struct,
%chord_struct* %chord_struct726
1207  store %chord_struct %actual_chord_struct737, %chord_struct*
%pointer_chord_elem_list724
1208  %pointer_chord_elem_list738 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array709, i32 2
1209  %malloccall1739 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1210  %chord_struct740 = bitcast i8* %malloccall1739 to
%chord_struct*
1211  %length741 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct740, i32 0, i32 0
1212  store i32 1, i32* %length741
1213  %malloccall1742 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1214  %arr_pitch743 = bitcast i8* %malloccall1742 to i32**
1215  %pitch_pointer_elem744 = getelementptr i32*, i32**
%arr_pitch743, i32 0
1216  %malloccall1745 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1217  %array746 = bitcast i8* %malloccall1745 to i32*
1218  %prefield_elem747 = getelementptr i32, i32* %array746, i32
0
1219  store i32 -1, i32* %prefield_elem747
1220  %scaledegreer_elem748 = getelementptr i32, i32* %array746,
i32 1
1221  store i32 7, i32* %scaledegreer_elem748
1222  %postfield_elem749 = getelementptr i32, i32* %array746, i32

```

```

2
1223 store i32 0, i32* %postfield_elem749
1224 store i32* %array746, i32** %pitch_pointer_elem744
1225 %struct_c_pointer750 = getelementptr inbounds
    %chord_struct, %chord_struct* %chord_struct740, i32 0, i32 1
1226 store i32** %arr_pitch743, i32*** %struct_c_pointer750
1227 %actual_chord_struct751 = load %chord_struct,
    %chord_struct* %chord_struct740
1228 store %chord_struct %actual_chord_struct751, %chord_struct*
    %pointer_chord_elem_list738
1229 %pointer_chord_elem_list752 = getelementptr %chord_struct,
    %chord_struct* %chord_pointer_array709, i32 3
1230 %malloccall753 = tail call i8* @malloc(i32 ptrtoint
    (%chord_struct* getelementptr (%chord_struct, %chord_struct*
        null, i32 1) to i32))
1231 %chord_struct754 = bitcast i8* %malloccall753 to
    %chord_struct*
1232 %length755 = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct754, i32 0, i32 0
1233 store i32 1, i32* %length755
1234 %malloccall756 = tail call i8* @malloc(i32 ptrtoint (i1**
    getelementptr (i1*, i1** null, i32 1) to i32))
1235 %arr_pitch757 = bitcast i8* %malloccall756 to i32**
1236 %pitch_pointer_elem758 = getelementptr i32*, i32**
    %arr_pitch757, i32 0
1237 %malloccall759 = tail call i8* @malloc(i32 mul (i32
    ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
    i32 3))
1238 %array760 = bitcast i8* %malloccall759 to i32*
1239 %prefield_elem761 = getelementptr i32, i32* %array760, i32
    0
1240 store i32 0, i32* %prefield_elem761
1241 %scaledegreer_elem762 = getelementptr i32, i32* %array760,
    i32 1
1242 store i32 2, i32* %scaledegreer_elem762
1243 %postfield_elem763 = getelementptr i32, i32* %array760, i32
    2
1244 store i32 0, i32* %postfield_elem763
1245 store i32* %array760, i32** %pitch_pointer_elem758
1246 %struct_c_pointer764 = getelementptr inbounds
    %chord_struct, %chord_struct* %chord_struct754, i32 0, i32 1
1247 store i32** %arr_pitch757, i32*** %struct_c_pointer764

```

```

1248     %actual_chord_struct765 = load %chord_struct,
1249     %chord_struct* %chord_struct754
1250     store %chord_struct %actual_chord_struct765, %chord_struct*
1251     %pointer_chord_elem_list752
1252     %pointer_chord_elem_list766 = getelementptr %chord_struct,
1253     %chord_struct* %chord_pointer_array709, i32 4
1254     %malloccall767 = tail call i8* @malloc(i32 ptrtoint
1255     (%chord_struct* getelementptr (%chord_struct, %chord_struct*
1256     null, i32 1) to i32))
1257     %chord_struct768 = bitcast i8* %malloccall767 to
1258     %chord_struct*
1259     %length769 = getelementptr inbounds %chord_struct,
1260     %chord_struct* %chord_struct768, i32 0, i32 0
1261     store i32 1, i32* %length769
1262     %malloccall770 = tail call i8* @malloc(i32 ptrtoint (i1**
1263     getelementptr (i1*, i1** null, i32 1) to i32))
1264     %arr_pitch771 = bitcast i8* %malloccall770 to i32**
1265     %pitch_pointer_elem772 = getelementptr i32*, i32**
1266     %arr_pitch771, i32 0
1267     %malloccall773 = tail call i8* @malloc(i32 mul (i32
1268     ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
1269     i32 3))
1270     %array774 = bitcast i8* %malloccall773 to i32*
1271     %prefield_elem775 = getelementptr i32, i32* %array774, i32
1272     0
1273     store i32 0, i32* %prefield_elem775
1274     %scaledegreer_elem776 = getelementptr i32, i32* %array774,
1275     i32 1
1276     store i32 4, i32* %scaledegreer_elem776
1277     %postfield_elem777 = getelementptr i32, i32* %array774, i32
1278     2
1279     store i32 0, i32* %postfield_elem777
1280     store i32* %array774, i32** %pitch_pointer_elem772
1281     %struct_c_pointer778 = getelementptr inbounds
1282     %chord_struct, %chord_struct* %chord_struct768, i32 0, i32 1
1283     store i32** %arr_pitch771, i32*** %struct_c_pointer778
1284     %actual_chord_struct779 = load %chord_struct,
1285     %chord_struct* %chord_struct768
1286     store %chord_struct %actual_chord_struct779, %chord_struct*
1287     %pointer_chord_elem_list766
1288     %pointer_chord_elem_list780 = getelementptr %chord_struct,
1289     %chord_struct* %chord_pointer_array709, i32 5

```

```

1272    %malloccall1781 = tail call i8* @malloc(i32 ptrtoint
1273        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
1274            null, i32 1) to i32))
1273    %chord_struct782 = bitcast i8* %malloccall1781 to
1274        %chord_struct*
1274    %length783 = getelementptr inbounds %chord_struct,
1275        %chord_struct* %chord_struct782, i32 0, i32 0
1275    store i32 1, i32* %length783
1276    %malloccall1784 = tail call i8* @malloc(i32 ptrtoint (i1**
1277        getelementptr (i1*, i1** null, i32 1) to i32))
1277    %arr_pitch785 = bitcast i8* %malloccall1784 to i32**
1278    %pitch_pointer_elem786 = getelementptr i32*, i32**
1278    %arr_pitch785, i32 0
1279    %malloccall1787 = tail call i8* @malloc(i32 mul (i32
1280        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
1280        i32 3))
1280    %array788 = bitcast i8* %malloccall1787 to i32*
1281    %prefield_elem789 = getelementptr i32, i32* %array788, i32
1281    0
1282    store i32 -1, i32* %prefield_elem789
1283    %scaledegreer_elem790 = getelementptr i32, i32* %array788,
1283    i32 1
1284    store i32 7, i32* %scaledegreer_elem790
1285    %postfield_elem791 = getelementptr i32, i32* %array788, i32
1285    2
1286    store i32 0, i32* %postfield_elem791
1287    store i32* %array788, i32** %pitch_pointer_elem786
1288    %struct_c_pointer792 = getelementptr inbounds
1288        %chord_struct, %chord_struct* %chord_struct782, i32 0, i32 1
1289    store i32** %arr_pitch785, i32*** %struct_c_pointer792
1290    %actual_chord_struct793 = load %chord_struct,
1290        %chord_struct* %chord_struct782
1291    store %chord_struct %actual_chord_struct793, %chord_struct*
1291        %pointer_chord_elem_list780
1292    %pointer_chord_elem_list794 = getelementptr %chord_struct,
1292        %chord_struct* %chord_pointer_array709, i32 6
1293    %malloccall1795 = tail call i8* @malloc(i32 ptrtoint
1293        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
1293            null, i32 1) to i32))
1294    %chord_struct796 = bitcast i8* %malloccall1795 to
1294        %chord_struct*
1295    %length797 = getelementptr inbounds %chord_struct,

```

```

%chord_struct* %chord_struct796, i32 0, i32 0
1296  store i32 1, i32* %length797
1297  %malloccall798 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1298  %arr_pitch799 = bitcast i8* %malloccall798 to i32**
1299  %pitch_pointer_elem800 = getelementptr i32*, i32**
%arr_pitch799, i32 0
1300  %malloccall801 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1301  %array802 = bitcast i8* %malloccall801 to i32*
1302  %prefield_elem803 = getelementptr i32, i32* %array802, i32
0
1303  store i32 0, i32* %prefield_elem803
1304  %scaledegreer_elem804 = getelementptr i32, i32* %array802,
i32 1
1305  store i32 2, i32* %scaledegreer_elem804
1306  %postfield_elem805 = getelementptr i32, i32* %array802, i32
2
1307  store i32 0, i32* %postfield_elem805
1308  store i32* %array802, i32** %pitch_pointer_elem800
1309  %struct_c_pointer806 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct796, i32 0, i32 1
1310  store i32** %arr_pitch799, i32*** %struct_c_pointer806
1311  %actual_chord_struct807 = load %chord_struct,
%chord_struct* %chord_struct796
1312  store %chord_struct %actual_chord_struct807, %chord_struct*
%pointer_chord_elem_list794
1313  %pointer_chord_elem_list808 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array709, i32 7
1314  %malloccall809 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1315  %chord_struct810 = bitcast i8* %malloccall809 to
%chord_struct*
1316  %length811 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct810, i32 0, i32 0
1317  store i32 1, i32* %length811
1318  %malloccall812 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1319  %arr_pitch813 = bitcast i8* %malloccall812 to i32**
1320  %pitch_pointer_elem814 = getelementptr i32*, i32**

```

```

%arr_pitch813, i32 0
1321    %malloccall1815 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1322    %array816 = bitcast i8* %malloccall1815 to i32*
1323    %prefield_elem817 = getelementptr i32, i32* %array816, i32
0
1324    store i32 0, i32* %prefield_elem817
1325    %scaledegreer_elem818 = getelementptr i32, i32* %array816,
i32 1
1326    store i32 4, i32* %scaledegreer_elem818
1327    %postfield_elem819 = getelementptr i32, i32* %array816, i32
2
1328    store i32 0, i32* %postfield_elem819
1329    store i32* %array816, i32** %pitch_pointer_elem814
1330    %struct_c_pointer820 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct810, i32 0, i32 1
1331    store i32** %arr_pitch813, i32*** %struct_c_pointer820
1332    %actual_chord_struct821 = load %chord_struct,
%chord_struct* %chord_struct810
1333    store %chord_struct %actual_chord_struct821, %chord_struct*
%pointer_chord_elem_list808
1334    %pointer_chord_elem_list822 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array709, i32 8
1335    %malloccall1823 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1336    %chord_struct824 = bitcast i8* %malloccall1823 to
%chord_struct*
1337    %length825 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct824, i32 0, i32 0
1338    store i32 1, i32* %length825
1339    %malloccall1826 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1340    %arr_pitch827 = bitcast i8* %malloccall1826 to i32**
1341    %pitch_pointer_elem828 = getelementptr i32*, i32**
%arr_pitch827, i32 0
1342    %malloccall1829 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1343    %array830 = bitcast i8* %malloccall1829 to i32*
1344    %prefield_elem831 = getelementptr i32, i32* %array830, i32

```

```

0
1345  store i32 0, i32* %prefield_elem831
1346  %scaledegreer_elem832 = getelementptr i32, i32* %array830,
i32 1
1347  store i32 6, i32* %scaledegreer_elem832
1348  %postfield_elem833 = getelementptr i32, i32* %array830, i32
2
1349  store i32 0, i32* %postfield_elem833
1350  store i32* %array830, i32** %pitch_pointer_elem828
1351  %struct_c_pointer834 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct824, i32 0, i32 1
1352  store i32** %arr_pitch827, i32*** %struct_c_pointer834
1353  %actual_chord_struct835 = load %chord_struct,
%chord_struct* %chord_struct824
1354  store %chord_struct %actual_chord_struct835, %chord_struct*
%pointer_chord_elem_list822
1355  %pointer_chord_elem_list836 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array709, i32 9
1356  %malloccall1837 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1357  %chord_struct838 = bitcast i8* %malloccall1837 to
%chord_struct*
1358  %length839 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct838, i32 0, i32 0
1359  store i32 1, i32* %length839
1360  %malloccall1840 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1361  %arr_pitch841 = bitcast i8* %malloccall1840 to i32**
1362  %pitch_pointer_elem842 = getelementptr i32*, i32**
%arr_pitch841, i32 0
1363  %malloccall1843 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1364  %array844 = bitcast i8* %malloccall1843 to i32*
1365  %prefield_elem845 = getelementptr i32, i32* %array844, i32
0
1366  store i32 0, i32* %prefield_elem845
1367  %scaledegreer_elem846 = getelementptr i32, i32* %array844,
i32 1
1368  store i32 5, i32* %scaledegreer_elem846
1369  %postfield_elem847 = getelementptr i32, i32* %array844, i32

```

```

2
1370  store i32 0, i32* %postfield_elem847
1371  store i32* %array844, i32** %pitch_pointer_elem842
1372  %struct_c_pointer848 = getelementptr inbounds
      %chord_struct, %chord_struct* %chord_struct838, i32 0, i32 1
1373  store i32** %arr_pitch841, i32*** %struct_c_pointer848
1374  %actual_chord_struct849 = load %chord_struct,
      %chord_struct* %chord_struct838
1375  store %chord_struct %actual_chord_struct849, %chord_struct*
      %pointer_chord_elem_list836
1376  %pointer_chord_elem_list850 = getelementptr %chord_struct,
      %chord_struct* %chord_pointer_array709, i32 10
1377  %malloccall851 = tail call i8* @malloc(i32 ptrtoint
      (%chord_struct* getelementptr (%chord_struct, %chord_struct*
      null, i32 1) to i32))
1378  %chord_struct852 = bitcast i8* %malloccall851 to
      %chord_struct*
1379  %length853 = getelementptr inbounds %chord_struct,
      %chord_struct* %chord_struct852, i32 0, i32 0
1380  store i32 1, i32* %length853
1381  %malloccall854 = tail call i8* @malloc(i32 ptrtoint (i1**
      getelementptr (i1*, i1** null, i32 1) to i32))
1382  %arr_pitch855 = bitcast i8* %malloccall854 to i32**
1383  %pitch_pointer_elem856 = getelementptr i32*, i32**
      %arr_pitch855, i32 0
1384  %malloccall857 = tail call i8* @malloc(i32 mul (i32
      ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
      i32 3))
1385  %array858 = bitcast i8* %malloccall857 to i32*
1386  %prefield_elem859 = getelementptr i32, i32* %array858, i32
      0
1387  store i32 0, i32* %prefield_elem859
1388  %scaledegreer_elem860 = getelementptr i32, i32* %array858,
      i32 1
1389  store i32 4, i32* %scaledegreer_elem860
1390  %postfield_elem861 = getelementptr i32, i32* %array858, i32
      2
1391  store i32 0, i32* %postfield_elem861
1392  store i32* %array858, i32** %pitch_pointer_elem856
1393  %struct_c_pointer862 = getelementptr inbounds
      %chord_struct, %chord_struct* %chord_struct852, i32 0, i32 1
1394  store i32** %arr_pitch855, i32*** %struct_c_pointer862

```

```

1395    %actual_chord_struct863 = load %chord_struct,
1396    %chord_struct* %chord_struct852
1396    store %chord_struct %actual_chord_struct863, %chord_struct*
1397    %pointer_chord_elem_list850
1397    %pointer_chord_elem_list864 = getelementptr %chord_struct,
1398    %chord_struct* %chord_pointer_array709, i32 11
1398    %malloccall865 = tail call i8* @malloc(i32 ptrtoint
1398    (%chord_struct* getelementptr (%chord_struct, %chord_struct*
1398    null, i32 1) to i32))
1399    %chord_struct866 = bitcast i8* %malloccall865 to
1399    %chord_struct*
1400    %length867 = getelementptr inbounds %chord_struct,
1400    %chord_struct* %chord_struct866, i32 0, i32 0
1401    store i32 1, i32* %length867
1402    %malloccall868 = tail call i8* @malloc(i32 ptrtoint (i1**
1402    getelementptr (i1*, i1** null, i32 1) to i32))
1403    %arr_pitch869 = bitcast i8* %malloccall868 to i32**
1404    %pitch_pointer_elem870 = getelementptr i32*, i32**
1404    %arr_pitch869, i32 0
1405    %malloccall871 = tail call i8* @malloc(i32 mul (i32
1405    ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
1405    i32 3))
1406    %array872 = bitcast i8* %malloccall871 to i32*
1407    %prefield_elem873 = getelementptr i32, i32* %array872, i32
1407    0
1408    store i32 0, i32* %prefield_elem873
1409    %scaledegreer_elem874 = getelementptr i32, i32* %array872,
1409    i32 1
1410    store i32 6, i32* %scaledegreer_elem874
1411    %postfield_elem875 = getelementptr i32, i32* %array872, i32
1411    2
1412    store i32 0, i32* %postfield_elem875
1413    store i32* %array872, i32** %pitch_pointer_elem870
1414    %struct_c_pointer876 = getelementptr inbounds
1414    %chord_struct, %chord_struct* %chord_struct866, i32 0, i32 1
1415    store i32** %arr_pitch869, i32*** %struct_c_pointer876
1416    %actual_chord_struct877 = load %chord_struct,
1416    %chord_struct* %chord_struct866
1417    store %chord_struct %actual_chord_struct877, %chord_struct*
1417    %pointer_chord_elem_list864
1418    %pointer_chord_elem_list878 = getelementptr %chord_struct,
1418    %chord_struct* %chord_pointer_array709, i32 12

```

```

1419    %malloccall1879 = tail call i8* @malloc(i32 ptrtoint
        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
        null, i32 1) to i32))
1420    %chord_struct880 = bitcast i8* %malloccall1879 to
        %chord_struct*
1421    %length881 = getelementptr inbounds %chord_struct,
        %chord_struct* %chord_struct880, i32 0, i32 0
1422    store i32 1, i32* %length881
1423    %malloccall1882 = tail call i8* @malloc(i32 ptrtoint (i1**
        getelementptr (i1*, i1** null, i32 1) to i32))
1424    %arr_pitch883 = bitcast i8* %malloccall1882 to i32**
1425    %pitch_pointer_elem884 = getelementptr i32*, i32**
        %arr_pitch883, i32 0
1426    %malloccall1885 = tail call i8* @malloc(i32 mul (i32
        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
        i32 3))
1427    %array886 = bitcast i8* %malloccall1885 to i32*
1428    %prefield_elem887 = getelementptr i32, i32* %array886, i32
        0
1429    store i32 0, i32* %prefield_elem887
1430    %scaledegreer_elem888 = getelementptr i32, i32* %array886,
        i32 1
1431    store i32 5, i32* %scaledegreer_elem888
1432    %postfield_elem889 = getelementptr i32, i32* %array886, i32
        2
1433    store i32 0, i32* %postfield_elem889
1434    store i32* %array886, i32** %pitch_pointer_elem884
1435    %struct_c_pointer890 = getelementptr inbounds
        %chord_struct, %chord_struct* %chord_struct880, i32 0, i32 1
1436    store i32** %arr_pitch883, i32*** %struct_c_pointer890
1437    %actual_chord_struct891 = load %chord_struct,
        %chord_struct* %chord_struct880
1438    store %chord_struct %actual_chord_struct891, %chord_struct*
        %pointer_chord_elem_list878
1439    %pointer_chord_elem_list892 = getelementptr %chord_struct,
        %chord_struct* %chord_pointer_array709, i32 13
1440    %malloccall1893 = tail call i8* @malloc(i32 ptrtoint
        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
        null, i32 1) to i32))
1441    %chord_struct894 = bitcast i8* %malloccall1893 to
        %chord_struct*
1442    %length895 = getelementptr inbounds %chord_struct,

```

```

1443    %chord_struct* %chord_struct894, i32 0, i32 0
1444    store i32 1, i32* %length895
1445    %malloccall896 = tail call i8* @malloc(i32 ptrtoint (i1**
1446        getelementptr (i1*, i1** null, i32 1) to i32))
1447    %arr_pitch897 = bitcast i8* %malloccall896 to i32**
1448    %pitch_pointer_elem898 = getelementptr i32*, i32**
1449    %arr_pitch897, i32 0
1450    %malloccall899 = tail call i8* @malloc(i32 mul (i32
1451        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
1452        i32 3))
1453    %array900 = bitcast i8* %malloccall899 to i32*
1454    %prefield_elem901 = getelementptr i32, i32* %array900, i32
1455    0
1456    store i32 0, i32* %prefield_elem901
1457    %scaledegreer_elem902 = getelementptr i32, i32* %array900,
1458    i32 1
1459    store i32 4, i32* %scaledegreer_elem902
1460    %postfield_elem903 = getelementptr i32, i32* %array900, i32
1461    2
1462    store i32 0, i32* %postfield_elem903
1463    store i32* %array900, i32** %pitch_pointer_elem898
1464    %struct_c_pointer904 = getelementptr inbounds
1465        %chord_struct, %chord_struct* %chord_struct894, i32 0, i32 1
1466    store i32** %arr_pitch897, i32*** %struct_c_pointer904
1467    %actual_chord_struct905 = load %chord_struct,
1468    %chord_struct* %chord_struct894
1469    store %chord_struct %actual_chord_struct905, %chord_struct*
1470    %pointer_chord_elem_list892
1471    %pointer_chord_elem_list906 = getelementptr %chord_struct,
1472    %chord_struct* %chord_pointer_array709, i32 14
1473    %malloccall907 = tail call i8* @malloc(i32 ptrtoint
1474        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
1475        null, i32 1) to i32))
1476    %chord_struct908 = bitcast i8* %malloccall907 to
1477    %chord_struct*
1478    %length909 = getelementptr inbounds %chord_struct,
1479    %chord_struct* %chord_struct908, i32 0, i32 0
1480    store i32 1, i32* %length909
1481    %malloccall910 = tail call i8* @malloc(i32 ptrtoint (i1**
1482        getelementptr (i1*, i1** null, i32 1) to i32))
1483    %arr_pitch911 = bitcast i8* %malloccall910 to i32**
1484    %pitch_pointer_elem912 = getelementptr i32*, i32**

```

```

%arr_pitch911, i32 0
1468    %malloccall1913 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1469    %array914 = bitcast i8* %malloccall1913 to i32*
1470    %prefield_elem915 = getelementptr i32, i32* %array914, i32
0
1471    store i32 0, i32* %prefield_elem915
1472    %scaledegreer_elem916 = getelementptr i32, i32* %array914,
i32 1
1473    store i32 3, i32* %scaledegreer_elem916
1474    %postfield_elem917 = getelementptr i32, i32* %array914, i32
2
1475    store i32 0, i32* %postfield_elem917
1476    store i32* %array914, i32** %pitch_pointer_elem912
1477    %struct_c_pointer918 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct908, i32 0, i32 1
1478    store i32** %arr_pitch911, i32*** %struct_c_pointer918
1479    %actual_chord_struct919 = load %chord_struct,
%chord_struct* %chord_struct908
1480    store %chord_struct %actual_chord_struct919, %chord_struct*
%pointer_chord_elem_list906
1481    %pointer_chord_elem_list920 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array709, i32 15
1482    %malloccall1921 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1483    %chord_struct922 = bitcast i8* %malloccall1921 to
%chord_struct*
1484    %length923 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct922, i32 0, i32 0
1485    store i32 1, i32* %length923
1486    %malloccall1924 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1487    %arr_pitch925 = bitcast i8* %malloccall1924 to i32**
1488    %pitch_pointer_elem926 = getelementptr i32*, i32**
%arr_pitch925, i32 0
1489    %malloccall1927 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1490    %array928 = bitcast i8* %malloccall1927 to i32*
1491    %prefield_elem929 = getelementptr i32, i32* %array928, i32

```

```

0
1492  store i32 0, i32* %prefield_elem929
1493  %scaledegreer_elem930 = getelementptr i32, i32* %array928,
i32 1
1494  store i32 2, i32* %scaledegreer_elem930
1495  %postfield_elem931 = getelementptr i32, i32* %array928, i32
2
1496  store i32 0, i32* %postfield_elem931
1497  store i32* %array928, i32** %pitch_pointer_elem926
1498  %struct_c_pointer932 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct922, i32 0, i32 1
1499  store i32** %arr_pitch925, i32*** %struct_c_pointer932
1500  %actual_chord_struct933 = load %chord_struct,
%chord_struct* %chord_struct922
1501  store %chord_struct %actual_chord_struct933, %chord_struct*
%pointer_chord_elem_list920
1502  %pointer_chord_elem_list934 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array709, i32 16
1503  %malloccall935 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1504  %chord_struct936 = bitcast i8* %malloccall935 to
%chord_struct*
1505  %length937 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct936, i32 0, i32 0
1506  store i32 1, i32* %length937
1507  %malloccall938 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1508  %arr_pitch939 = bitcast i8* %malloccall938 to i32**
1509  %pitch_pointer_elem940 = getelementptr i32*, i32**
%arr_pitch939, i32 0
1510  %malloccall941 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1511  %array942 = bitcast i8* %malloccall941 to i32*
1512  %prefield_elem943 = getelementptr i32, i32* %array942, i32
0
1513  store i32 0, i32* %prefield_elem943
1514  %scaledegreer_elem944 = getelementptr i32, i32* %array942,
i32 1
1515  store i32 1, i32* %scaledegreer_elem944
1516  %postfield_elem945 = getelementptr i32, i32* %array942, i32

```

```

2
1517  store i32 0, i32* %postfield_elem945
1518  store i32* %array942, i32** %pitch_pointer_elem940
1519  %struct_c_pointer946 = getelementptr inbounds
        %chord_struct, %chord_struct* %chord_struct936, i32 0, i32 1
1520  store i32** %arr_pitch939, i32*** %struct_c_pointer946
1521  %actual_chord_struct947 = load %chord_struct,
        %chord_struct* %chord_struct936
1522  store %chord_struct %actual_chord_struct947, %chord_struct*
        %pointer_chord_elem_list934
1523  %pointer_chord_elem_list948 = getelementptr %chord_struct,
        %chord_struct* %chord_pointer_array709, i32 17
1524  %malloccall949 = tail call i8* @malloc(i32 ptrtoint
        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
        null, i32 1) to i32))
1525  %chord_struct950 = bitcast i8* %malloccall949 to
        %chord_struct*
1526  %length951 = getelementptr inbounds %chord_struct,
        %chord_struct* %chord_struct950, i32 0, i32 0
1527  store i32 1, i32* %length951
1528  %malloccall952 = tail call i8* @malloc(i32 ptrtoint (i1**
        getelementptr (i1*, i1** null, i32 1) to i32))
1529  %arr_pitch953 = bitcast i8* %malloccall952 to i32**
1530  %pitch_pointer_elem954 = getelementptr i32*, i32**
        %arr_pitch953, i32 0
1531  %malloccall955 = tail call i8* @malloc(i32 mul (i32
        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
        i32 3))
1532  %array956 = bitcast i8* %malloccall955 to i32*
1533  %prefield_elem957 = getelementptr i32, i32* %array956, i32
        0
1534  store i32 -1, i32* %prefield_elem957
1535  %scaledegreer_elem958 = getelementptr i32, i32* %array956,
        i32 1
1536  store i32 7, i32* %scaledegreer_elem958
1537  %postfield_elem959 = getelementptr i32, i32* %array956, i32
        2
1538  store i32 0, i32* %postfield_elem959
1539  store i32* %array956, i32** %pitch_pointer_elem954
1540  %struct_c_pointer960 = getelementptr inbounds
        %chord_struct, %chord_struct* %chord_struct950, i32 0, i32 1
1541  store i32** %arr_pitch953, i32*** %struct_c_pointer960

```

```

1542    %actual_chord_struct961 = load %chord_struct,
1543    %chord_struct* %chord_struct950
1543    store %chord_struct %actual_chord_struct961, %chord_struct*
1544    %pointer_chord_elem_list948
1544    %pointer_chord_elem_list962 = getelementptr %chord_struct,
1545    %chord_struct* %chord_pointer_array709, i32 18
1545    %malloccall963 = tail call i8* @malloc(i32 ptrtoint
1545    (%chord_struct* getelementptr (%chord_struct, %chord_struct*
1545    null, i32 1) to i32))
1546    %chord_struct964 = bitcast i8* %malloccall963 to
1546    %chord_struct*
1547    %length965 = getelementptr inbounds %chord_struct,
1547    %chord_struct* %chord_struct964, i32 0, i32 0
1548    store i32 1, i32* %length965
1549    %malloccall966 = tail call i8* @malloc(i32 ptrtoint (i1**
1549    getelementptr (i1*, i1** null, i32 1) to i32))
1550    %arr_pitch967 = bitcast i8* %malloccall966 to i32**
1551    %pitch_pointer_elem968 = getelementptr i32*, i32**
1551    %arr_pitch967, i32 0
1552    %malloccall969 = tail call i8* @malloc(i32 mul (i32
1552    ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
1552    i32 3))
1553    %array970 = bitcast i8* %malloccall969 to i32*
1554    %prefield_elem971 = getelementptr i32, i32* %array970, i32
1554    0
1555    store i32 -1, i32* %prefield_elem971
1556    %scaledegreer_elem972 = getelementptr i32, i32* %array970,
1556    i32 1
1557    store i32 6, i32* %scaledegreer_elem972
1558    %postfield_elem973 = getelementptr i32, i32* %array970, i32
1558    2
1559    store i32 0, i32* %postfield_elem973
1560    store i32* %array970, i32** %pitch_pointer_elem968
1561    %struct_c_pointer974 = getelementptr inbounds
1561    %chord_struct, %chord_struct* %chord_struct964, i32 0, i32 1
1562    store i32** %arr_pitch967, i32*** %struct_c_pointer974
1563    %actual_chord_struct975 = load %chord_struct,
1563    %chord_struct* %chord_struct964
1564    store %chord_struct %actual_chord_struct975, %chord_struct*
1564    %pointer_chord_elem_list962
1565    %pointer_chord_elem_list976 = getelementptr %chord_struct,
1565    %chord_struct* %chord_pointer_array709, i32 19

```

```

1566    %malloccall1977 = tail call i8* @malloc(i32 ptrtoint
        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
        null, i32 1) to i32))
1567    %chord_struct978 = bitcast i8* %malloccall1977 to
        %chord_struct*
1568    %length979 = getelementptr inbounds %chord_struct,
        %chord_struct* %chord_struct978, i32 0, i32 0
1569    store i32 1, i32* %length979
1570    %malloccall1980 = tail call i8* @malloc(i32 ptrtoint (i1**
        getelementptr (i1*, i1** null, i32 1) to i32))
1571    %arr_pitch981 = bitcast i8* %malloccall1980 to i32**
1572    %pitch_pointer_elem982 = getelementptr i32*, i32**
        %arr_pitch981, i32 0
1573    %malloccall1983 = tail call i8* @malloc(i32 mul (i32
        ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
        i32 3))
1574    %array984 = bitcast i8* %malloccall1983 to i32*
1575    %prefield_elem985 = getelementptr i32, i32* %array984, i32
        0
1576    store i32 -1, i32* %prefield_elem985
1577    %scaledegreer_elem986 = getelementptr i32, i32* %array984,
        i32 1
1578    store i32 5, i32* %scaledegreer_elem986
1579    %postfield_elem987 = getelementptr i32, i32* %array984, i32
        2
1580    store i32 0, i32* %postfield_elem987
1581    store i32* %array984, i32** %pitch_pointer_elem982
1582    %struct_c_pointer988 = getelementptr inbounds
        %chord_struct, %chord_struct* %chord_struct978, i32 0, i32 1
1583    store i32** %arr_pitch981, i32*** %struct_c_pointer988
1584    %actual_chord_struct989 = load %chord_struct,
        %chord_struct* %chord_struct978
1585    store %chord_struct %actual_chord_struct989, %chord_struct*
        %pointer_chord_elem_list976
1586    %pointer_chord_elem_list990 = getelementptr %chord_struct,
        %chord_struct* %chord_pointer_array709, i32 20
1587    %malloccall1991 = tail call i8* @malloc(i32 ptrtoint
        (%chord_struct* getelementptr (%chord_struct, %chord_struct*
        null, i32 1) to i32))
1588    %chord_struct992 = bitcast i8* %malloccall1991 to
        %chord_struct*
1589    %length993 = getelementptr inbounds %chord_struct,

```

```

%chord_struct* %chord_struct992, i32 0, i32 0
1590 store i32 1, i32* %length993
1591 %malloccall994 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1592 %arr_pitch995 = bitcast i8* %malloccall994 to i32**
1593 %pitch_pointer_elem996 = getelementptr i32*, i32**
%arr_pitch995, i32 0
1594 %malloccall997 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1595 %array998 = bitcast i8* %malloccall997 to i32*
1596 %prefield_elem999 = getelementptr i32, i32* %array998, i32
0
1597 store i32 0, i32* %prefield_elem999
1598 %scaledegreer_elem1000 = getelementptr i32, i32* %array998,
i32 1
1599 store i32 3, i32* %scaledegreer_elem1000
1600 %postfield_elem1001 = getelementptr i32, i32* %array998,
i32 2
1601 store i32 0, i32* %postfield_elem1001
1602 store i32* %array998, i32** %pitch_pointer_elem996
1603 %struct_c_pointer1002 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct992, i32 0, i32 1
1604 store i32** %arr_pitch995, i32*** %struct_c_pointer1002
1605 %actual_chord_struct1003 = load %chord_struct,
%chord_struct* %chord_struct992
1606 store %chord_struct %actual_chord_struct1003,
%chord_struct* %pointer_chord_elem_list990
1607 %struct_cl_pointer1004 = getelementptr inbounds
%chordlist_struct, %chordlist_struct* %cl_struct706, i32 0,
i32 1
1608 store %chord_struct* %chord_pointer_array709,
%chord_struct** %struct_cl_pointer1004
1609 %arp = alloca %chordlist_struct*
1610 store %chordlist_struct* %cl_struct706, %chordlist_struct**
%arp
1611 %array_struct1005 = alloca %list_struct_f
1612 %length1006 = getelementptr inbounds %list_struct_f,
%list_struct_f* %array_struct1005, i32 0, i32 0
1613 store i32 21, i32* %length1006
1614 %array1007 = alloca double, i32 21
1615 %elem1008 = getelementptr double, double* %array1007, i32 0

```

```

1616    store double 2.500000e-01, double* %elem1008
1617    %elem1009 = getelementptr double, double* %array1007, i32 1
1618    store double 2.500000e-01, double* %elem1009
1619    %elem1010 = getelementptr double, double* %array1007, i32 2
1620    store double 2.500000e-01, double* %elem1010
1621    %elem1011 = getelementptr double, double* %array1007, i32 3
1622    store double 2.500000e-01, double* %elem1011
1623    %elem1012 = getelementptr double, double* %array1007, i32 4
1624    store double 2.500000e-01, double* %elem1012
1625    %elem1013 = getelementptr double, double* %array1007, i32 5
1626    store double 2.500000e-01, double* %elem1013
1627    %elem1014 = getelementptr double, double* %array1007, i32 6
1628    store double 2.500000e-01, double* %elem1014
1629    %elem1015 = getelementptr double, double* %array1007, i32 7
1630    store double 2.500000e-01, double* %elem1015
1631    %elem1016 = getelementptr double, double* %array1007, i32 8
1632    store double 2.500000e-01, double* %elem1016
1633    %elem1017 = getelementptr double, double* %array1007, i32 9
1634    store double 2.500000e-01, double* %elem1017
1635    %elem1018 = getelementptr double, double* %array1007, i32
10
1636    store double 2.500000e-01, double* %elem1018
1637    %elem1019 = getelementptr double, double* %array1007, i32
11
1638    store double 2.500000e-01, double* %elem1019
1639    %elem1020 = getelementptr double, double* %array1007, i32
12
1640    store double 2.500000e-01, double* %elem1020
1641    %elem1021 = getelementptr double, double* %array1007, i32
13
1642    store double 2.500000e-01, double* %elem1021
1643    %elem1022 = getelementptr double, double* %array1007, i32
14
1644    store double 2.500000e-01, double* %elem1022
1645    %elem1023 = getelementptr double, double* %array1007, i32
15
1646    store double 2.500000e-01, double* %elem1023
1647    %elem1024 = getelementptr double, double* %array1007, i32
16
1648    store double 2.500000e-01, double* %elem1024
1649    %elem1025 = getelementptr double, double* %array1007, i32
17

```

```

1650    store double 2.500000e-01, double* %elem1025
1651    %elem1026 = getelementptr double, double* %array1007, i32
1652    18
1653    store double 2.500000e-01, double* %elem1026
1654    %elem1027 = getelementptr double, double* %array1007, i32
1655    19
1656    store double 2.500000e-01, double* %elem1027
1657    %elem1028 = getelementptr double, double* %array1007, i32
1658    20
1659    store double 1.250000e+00, double* %elem1028
1660    %actual_list1029 = getelementptr inbounds %list_struct_f,
1661    %list_struct_f* %array_struct1005, i32 0, i32 1
1662    store double* %array1007, double** %actual_list1029
1663    %r4 = alloca %list_struct_f*
1664    store %list_struct_f* %array_struct1005, %list_struct_f** %r4
1665    %theme1030 = load %chordlist_struct*, %chordlist_struct** %theme
1666    %length1031 = getelementptr inbounds %chordlist_struct,
1667    %chordlist_struct* %theme1030, i32 0, i32 0
1668    %size = load i32, i32* %length1031
1669    %r11032 = load %list_struct_f*, %list_struct_f** %r1
1670    %cur_list_ptr = getelementptr inbounds %list_struct_f,
1671    %list_struct_f* %r11032, i32 0, i32 1
1672    %cur_list = load double*, double** %cur_list_ptr
1673    %minor11033 = load %list_struct*, %list_struct** %minor1
1674    %cur_list_ptr1034 = getelementptr inbounds %list_struct,
1675    %list_struct* %minor11033, i32 0, i32 1
1676    %cur_list1035 = load i32*, i32** %cur_list_ptr1034
1677    %length1036 = getelementptr inbounds %list_struct,
1678    %list_struct* %minor11033, i32 0, i32 0
1679    %size1037 = load i32, i32* %length1036
1680    %mallocsize = mul i32 %size, ptrtoint (i32* getelementptr
1681    (i32, i32* null, i32 1) to i32)
1682    %malloccall1038 = tail call i8* @malloc(i32 %mallocsize)
1683    %return_arr = bitcast i8* %malloccall1038 to i32*
1684    %mallocsize1039 = mul i32 %size, ptrtoint (i1**
1685    getelementptr (i1*, i1** null, i32 1) to i32)
1686    %malloccall1040 = tail call i8* @malloc(i32
1687    %mallocsize1039)
1688    %return_arr1041 = bitcast i8* %malloccall1040 to i32**
1689    %mallocsize1042 = mul i32 %size, ptrtoint (i1**

```

```

      getelementptr (i1*, i1** null, i32 1) to i32)
1679      %malloccall1043 = tail call i8* @malloc(i32
      %mallocsize1042)
1680      %norm_arr = bitcast i8* %malloccall1043 to i32***
1681      %length1044 = getelementptr inbounds %chordlist_struct,
      %chordlist_struct* %theme1030, i32 0, i32 0
1682      %size1045 = load i32, i32* %length1044
1683      %cur_list_ptr1046 = getelementptr inbounds
      %chordlist_struct, %chordlist_struct* %theme1030, i32 0, i32
      1
1684      %cur_list1047 = load %chord_struct*, %chord_struct**
      %cur_list_ptr1046
1685      %cur_index_ptr = alloca i32
1686      store i32 0, i32* %cur_index_ptr
1687      br label %while
1688
1689 while:                                ; preds =
      %while_body, %entry
1690     %cur_index2 = load i32, i32* %cur_index_ptr
1691     %pred = icmp ne i32 %size1045, %cur_index2
1692     br i1 %pred, label %while_body, label %merge
1693
1694 while_body:                            ; preds =
      %while
1695     %cur_indexplz = load i32, i32* %cur_index_ptr
1696     %cur_val = getelementptr inbounds %chord_struct,
      %chord_struct* %cur_list1047, i32 %cur_indexplz
1697     %val_idx = load %chord_struct, %chord_struct* %cur_val
1698     %cur_val1048 = getelementptr inbounds i32**, i32***
      %norm_arr, i32 %cur_indexplz
1699     %stuff = extractvalue %chord_struct %val_idx, 1
1700     %len = getelementptr inbounds i32, i32* %return_arr, i32
      %cur_indexplz
1701     %oldlen = extractvalue %chord_struct %val_idx, 0
1702     %len1049 = getelementptr inbounds i32*, i32**
      %return_arr1041, i32 %cur_indexplz
1703     %mallocsize1050 = mul i32 %oldlen, ptrtoint (i32*
      getelementptr (i32, i32* null, i32 1) to i32)
1704     %malloccall1051 = tail call i8* @malloc(i32
      %mallocsize1050)
1705     %clear_cl_list_elem = bitcast i8* %malloccall1051 to i32*
1706     store i32* %clear_cl_list_elem, i32** %len1049

```

```

1707    store i32 %oldlen, i32* %len
1708    store i32** %stuff, i32*** %cur_val1048
1709    %cur_index = load i32, i32* %cur_index_ptr
1710    %new_idx = add i32 %cur_index, 1
1711    store i32 %new_idx, i32* %cur_index_ptr
1712    br label %while
1713
1714 merge:                                ; preds =
%while
1715     %synth-buffer = tail call i8* @malloc(i32 mul (i32 ptrtoint
1716     (i8* getelementptr (i8, i8* null, i32 1) to i32), i32 1000))
1716     %synth = call i32 @synth(i32*** %norm_arr, i32 %size, i32*
1717     %return_arr, i32 50, i32* %cur_list1035, i32 %size1037,
1718     double* %cur_list, i32** %return_arr1041, i32 1, i8* %synth-
1719     buffer)
1720     %rone = alloca i8*
1721     store i8* %synth-buffer, i8** %rone
1722     %counter1053 = load %chordlist_struct*, %chordlist_struct**
1723     %counter
1724     %length1054 = getelementptr inbounds %chordlist_struct,
1725     %chordlist_struct* %counter1053, i32 0, i32 0
1726     %size1055 = load i32, i32* %length1054
1727     %r31056 = load %list_struct_f*, %list_struct_f** %r3
1728     %cur_list_ptr1057 = getelementptr inbounds %list_struct_f,
1729     %list_struct_f* %r31056, i32 0, i32 1
1730     %cur_list1058 = load double*, double** %cur_list_ptr1057
1731     %minor11059 = load %list_struct*, %list_struct** %minor1
1732     %cur_list_ptr1060 = getelementptr inbounds %list_struct,
1733     %list_struct* %minor11059, i32 0, i32 1
1734     %cur_list1061 = load i32*, i32** %cur_list_ptr1060
1735     %length1062 = getelementptr inbounds %list_struct,
1736     %list_struct* %minor11059, i32 0, i32 0
1737     %size1063 = load i32, i32* %length1062
1738     %mallocsize1064 = mul i32 %size1055, ptrtoint (i32*
1739     getelementptr (i32, i32* null, i32 1) to i32)
1740     %malloccall1065 = tail call i8* @malloc(i32
1741     %mallocsize1064)
1742     %return_arr1066 = bitcast i8* %malloccall1065 to i32*
1743     %mallocsize1067 = mul i32 %size1055, ptrtoint (i1**
1744     getelementptr (i1*, i1** null, i32 1) to i32)
1745     %malloccall1068 = tail call i8* @malloc(i32
1746     %mallocsize1067)

```

```

1735    %return_arr1069 = bitcast i8* %malloccall1068 to i32**
1736    %mallocsize1070 = mul i32 %size1055, ptrtoint (i1**
1737        getelementptr (i1*, i1** null, i32 1) to i32)
1738    %malloccall1071 = tail call i8* @malloc(i32
1739        %mallocsize1070)
1740    %norm_arr1072 = bitcast i8* %malloccall1071 to i32***,
1741    %length1073 = getelementptr inbounds %chordlist_struct,
1742        %chordlist_struct* %counter1053, i32 0, i32 0
1743    %size1074 = load i32, i32* %length1073
1744    %cur_list_ptr1075 = getelementptr inbounds
1745        %chordlist_struct, %chordlist_struct* %counter1053, i32 0,
1746        i32 1
1747    %cur_list1076 = load %chord_struct*, %chord_struct**
1748        %cur_list_ptr1075
1749    %cur_index_ptr1077 = alloca i32
1750    store i32 0, i32* %cur_index_ptr1077
1751    br label %while1078
1752
1753 while1078:                                ; preds =
1754     %while_body1079, %merge
1755     %cur_index21093 = load i32, i32* %cur_index_ptr1077
1756     %pred1094 = icmp ne i32 %size1074, %cur_index21093
1757     br i1 %pred1094, label %while_body1079, label %merge1095
1758
1759 while_body1079:                            ; preds =
1760     %while1078
1761     %cur_indexplz1080 = load i32, i32* %cur_index_ptr1077
1762     %cur_val1081 = getelementptr inbounds %chord_struct,
1763         %chord_struct* %cur_list1076, i32 %cur_indexplz1080
1764     %val_idx1082 = load %chord_struct, %chord_struct*
1765         %cur_val1081
1766     %cur_val1083 = getelementptr inbounds i32**, i32***,
1767         %norm_arr1072, i32 %cur_indexplz1080
1768     %stuff1084 = extractvalue %chord_struct %val_idx1082, 1
1769     %len1085 = getelementptr inbounds i32, i32*
1770         %return_arr1066, i32 %cur_indexplz1080
1771     %oldlen1086 = extractvalue %chord_struct %val_idx1082, 0
1772     %len1087 = getelementptr inbounds i32*, i32**
1773         %return_arr1069, i32 %cur_indexplz1080
1774     %mallocsize1088 = mul i32 %oldlen1086, ptrtoint (i32*
1775         getelementptr (i32, i32* null, i32 1) to i32)
1776     %malloccall1089 = tail call i8* @malloc(i32

```

```

%mallocsize1088)
1763   %clear_cl_list_elem1090 = bitcast i8* %malloccall1089 to
i32*
1764   store i32* %clear_cl_list_elem1090, i32** %len1087
1765   store i32 %oldlen1086, i32* %len1085
1766   store i32** %stuff1084, i32*** %cur_val1083
1767   %cur_index1091 = load i32, i32* %cur_index_ptr1077
1768   %new_idx1092 = add i32 %cur_index1091, 1
1769   store i32 %new_idx1092, i32* %cur_index_ptr1077
1770   br label %while1078
1771
1772 merge1095:                                ; preds =
%while1078
1773   %synth-buffer1097 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32),
i32 1000))
1774   %synth1098 = call i32 @synth(i32*** %norm_arr1072, i32
%size1055, i32* %return_arr1066, i32 57, i32* %cur_list1061,
i32 %size1063, double* %cur_list1058, i32** %return_arr1069,
i32 1, i8* %synth-buffer1097)
1775   %rtwo = alloca i8*
1776   store i8* %synth-buffer1097, i8** %rtwo
1777   %theme1099 = load %chordlist_struct*, %chordlist_struct**%
theme
1778   %length1100 = getelementptr inbounds %chordlist_struct,
%chordlist_struct* %theme1099, i32 0, i32 0
1779   %size1101 = load i32, i32* %length1100
1780   %r21102 = load %list_struct_f*, %list_struct_f** %r2
1781   %cur_list_ptr1103 = getelementptr inbounds %list_struct_f,
%list_struct_f* %r21102, i32 0, i32 1
1782   %cur_list1104 = load double*, double** %cur_list_ptr1103
1783   %minor11105 = load %list_struct*, %list_struct** %minor1
1784   %cur_list_ptr1106 = getelementptr inbounds %list_struct,
%list_struct* %minor11105, i32 0, i32 1
1785   %cur_list1107 = load i32*, i32** %cur_list_ptr1106
1786   %length1108 = getelementptr inbounds %list_struct,
%list_struct* %minor11105, i32 0, i32 0
1787   %size1109 = load i32, i32* %length1108
1788   %mallocsize1110 = mul i32 %size1101, ptrtoint (i32*
getelementptr (i32, i32* null, i32 1) to i32)
1789   %malloccall1111 = tail call i8* @malloc(i32
%mallocsize1110)

```

```

1790    %return_arr1112 = bitcast i8* %malloccall1111 to i32*
1791    %mallocsize1113 = mul i32 %size1101, ptrtoint (i1**
1792        getelementptr (i1*, i1** null, i32 1) to i32)
1793    %malloccall1114 = tail call i8* @malloc(i32
1794        %mallocsize1113)
1795    %return_arr1115 = bitcast i8* %malloccall1114 to i32**
1796    %mallocsize1116 = mul i32 %size1101, ptrtoint (i1**
1797        getelementptr (i1*, i1** null, i32 1) to i32)
1798    %malloccall1117 = tail call i8* @malloc(i32
1799        %mallocsize1116)
1800    %norm_arr1118 = bitcast i8* %malloccall1117 to i32***
```

%

```

1801    %length1119 = getelementptr inbounds %chordlist_struct,
1802        %chordlist_struct* %theme1099, i32 0, i32 0
1803    %size1120 = load i32, i32* %length1119
1804    %cur_list_ptr1121 = getelementptr inbounds
1805        %chordlist_struct, %chordlist_struct* %theme1099, i32 0, i32
1806        1
1807    %cur_list1122 = load %chord_struct*, %chord_struct**
1808        %cur_list_ptr1121
1809    %cur_index_ptr1123 = alloca i32
1810    store i32 0, i32* %cur_index_ptr1123
1811    br label %while1124
1812
1813 while1124:                                ; preds =
1814     %while_body1125, %merge1095
1815     %cur_index21139 = load i32, i32* %cur_index_ptr1123
1816     %pred1140 = icmp ne i32 %size1120, %cur_index21139
1817     br i1 %pred1140, label %while_body1125, label %merge1141
1818
1819 while_body1125:                            ; preds =
1820     %while1124
1821     %cur_indexplz1126 = load i32, i32* %cur_index_ptr1123
1822     %cur_val1127 = getelementptr inbounds %chord_struct,
1823        %chord_struct* %cur_list1122, i32 %cur_indexplz1126
1824     %val_idx1128 = load %chord_struct, %chord_struct*
1825        %cur_val1127
1826     %cur_val1129 = getelementptr inbounds i32**, i32***
```

%

```

1827     %norm_arr1118, i32 %cur_indexplz1126
1828     %stuff1130 = extractvalue %chord_struct %val_idx1128, 1
1829     %len1131 = getelementptr inbounds i32, i32*
1830         %return_arr1112, i32 %cur_indexplz1126
1831     %oldlen1132 = extractvalue %chord_struct %val_idx1128, 0

```

```

1818    %len1133 = getelementptr inbounds i32*, i32**
1819        %return_arr1115, i32 %cur_indexplz1126
1820        %mallocsize1134 = mul i32 %oldlen1132, ptrtoint (i32*
1821            getelementptr (i32, i32* null, i32 1) to i32)
1822        %malloccall1135 = tail call i8* @malloc(i32
1823            %mallocsize1134)
1824        %clear_cl_list_elem1136 = bitcast i8* %malloccall1135 to
1825            i32*
1826            store i32* %clear_cl_list_elem1136, i32** %len1133
1827            store i32 %oldlen1132, i32* %len1131
1828            store i32** %stuff1130, i32*** %cur_val1129
1829            %cur_index1137 = load i32, i32* %cur_index_ptr1123
1830            %new_idx1138 = add i32 %cur_index1137, 1
1831            store i32 %new_idx1138, i32* %cur_index_ptr1123
1832            br label %while1124
1833
1834 merge1141:                                ; preds =
1835     %while1124
1836     %synth-buffer1143 = tail call i8* @malloc(i32 mul (i32
1837         ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32),
1838         i32 1000))
1839     %synth1144 = call i32 @synth(i32*** %norm_arr1118, i32
1840         %size1101, i32* %return_arr1112, i32 33, i32* %cur_list1107,
1841         i32 %size1109, double* %cur_list1104, i32** %return_arr1115,
1842         i32 2, i8* %synth-buffer1143)
1843     %lone = alloca i8*
1844     store i8* %synth-buffer1143, i8** %lone
1845     %new_string = tail call i8* @malloc(i32 mul (i32 ptrtoint
1846         (i8* getelementptr (i8, i8* null, i32 1) to i32), i32 1000))
1847     %first = getelementptr inbounds i8, i8* %new_string, i32 0
1848     store i8 0, i8* %first
1849     %rone1146 = load i8*, i8** %rone
1850     %put_first = call i8* @strcat(i8* %new_string, i8*
1851         %rone1146)
1852     %lone1147 = load i8*, i8** %lone
1853     %put_first1148 = call i8* @strcat(i8* %new_string, i8*
1854         %lone1147)
1855     %song = alloca i8*
1856     store i8* %new_string, i8** %song
1857     %new_string1150 = tail call i8* @malloc(i32 mul (i32
1858         ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32),
1859         i32 1000))

```

```

1845    %first1151 = getelementptr inbounds i8, i8*
1846        %new_string1150, i32 0
1846        store i8 0, i8* %first1151
1847        %song1152 = load i8*, i8** %song
1848        %put_first1153 = call i8* @strcat(i8* %new_string1150, i8*
1848            %song1152)
1849        %rtwo1154 = load i8*, i8** %rtwo
1850        %put_first1155 = call i8* @strcat(i8* %new_string1150, i8*
1850            %rtwo1154)
1851        %song1 = alloca i8*
1852        store i8* %new_string1150, i8** %song1
1853        %song11156 = load i8*, i8** %song1
1854        %make_midi = call i32 @make_midi(i8* %song11156, i8*
1854            getelementptr inbounds ([10 x i8], [10 x i8]* @0, i32 0, i32
1854                0))
1855        ret i32 0
1856    }
1857
1858 declare i32 @synth(i32***, i32, i32*, i32, i32*, i32,
1858     double*, i32**, i32, i8*)
1859
1860 declare i8* @strcat(i8*, i8*)
1861
1862 declare i32 @make_midi(i8*, i8*)
1863
1864 declare noalias i8* @malloc(i32)

```

05/11/17 12:02:23 Untitled

```
1 y = 6;
2 def Fun x = x;
3 z = Fun(y);
4 Printint(z);
```

05/11/17 12:02:06 Untitled

```
1 ; ModuleID = 'MusicMike'
2
3 @fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
4 @str = private unnamed_addr constant [4 x i8] c"%s\0A\00"
5 @flt = private unnamed_addr constant [4 x i8] c"%f\0A\00"
6 @str.1 = private unnamed_addr constant [3 x i8] c"%c\00"
7 @fmt.2 = private unnamed_addr constant [4 x i8] c"%d \00"
8 @fmt.3 = private unnamed_addr constant [4 x i8] c"%f \00"
9
10 declare i32 @printf(i8*, ...)
11
12 define i32 @main() {
13 entry:
14     %y = alloca i32
15     store i32 6, i32* %y
16     %y1 = load i32, i32* %y
17     %Fun = call i32 @Fun(i32 %y1)
18     %z = alloca i32
19     store i32 %Fun, i32* %z
20     %z2 = load i32, i32* %z
21     %printf = call i32 (i8*, ...) @printf(i8* getelementptr
22     inbounds ([4 x i8], [4 x i8]* @fmt, i32 0, i32 0), i32 %z2)
22     ret i32 0
23 }
24
25 declare i32 @synth(i32***, i32, i32*, i32, i32*, i32, double*,
26 i32**, i32, i8*)
27
28 declare i8* @strcat(i8*, i8*)
29
30 declare i32 @make_midi(i8*, i8*)
31
32 define i32 @Fun(i32 %x) {
33 entry:
34     %x1 = alloca i32
35     store i32 %x, i32* %x1
36     %x2 = load i32, i32* %x1
37     ret i32 %x2
37 }
```

6.2 Test Suite

05/10/17 11:59:52 C:\Users\husam\Downloads\out (1).txt

```
1 fail-arith-1.mike
2
3 a=1+.1;
4
5 fail-arith2.mike
6
7 1.0+1.0;
8
9 fail-array2.mike
10
11 arr=[ "c" 2 3 4 ];
12
13 fail-array.mike
14
15 arr=[1 1. 3 ];
16
17 fail-assign.mike
18
19 a=c+1;
20
21 fail-if1.mike
22
23 if true then 5;
24
25 fail-if.mike
26
27 if 42 then Printint(5) else Printint(3);
28
29 fail-illegal-id.mike
30
31 b=2;
32
33 fail-index.mike
34
35 a=[1 2 3 ];
36 b=a.[ 3. ];
37
38 fail-pitch-empty.mike
39
40 p:[3 4 -4 ];
```

```

41
42 /* pitch literals cannot be negative */
43
44 test-1synth1.mike
45
46
47
48 pitch_list= p:[ 1 2 3 4 3 2 1];
49 mode=[1 3 5 6 8 10 12];
50 rhythm_list=r:[ q q q q q q q ];
51 start_note=60;
52
53 s = Synth(pitch_list rhythm_list mode start_note 1 );
54 s1 = Synth(pitch_list rhythm_list mode 63 2 );
55 s2 = Merge(s s1);
56 Make_midi(s2 "plzwork.midi");
57
58 test-arith1.mike
59
60 Printint(4);
61 Printint(4+4);
62 Printint(2*8);
63 Printint(10/2);
64
65
66 test-arith2.mike
67
68 Printffloat (2.22);
69 Printffloat (3.+2.5);
70 Printffloat (10.-.2.);
71 Printffloat (8.*.4.);
72 Printffloat (7./.2.);
73
74
75
76 test-array.mike
77
78 [1 2 3];
79
80 test-assign-array.mike
81
82 a = [1 2 3];

```

```
83
84 test-assign.mike
85
86 a = 5;
87 Printint (a);
88
89 test-assign-print.mike
90
91
92
93 a = 5; Printint(a);
94
95 test-block-return.mike
96
97 /* testing whether {4;5;10;} returns "10" as its value */
98 Printint({4;5;10;});
99
100 test-channel-synth.mike
101
102 p1=p:[ 1 2 3 4 5];
103 mode1= [ 1 2 3 4 5];
104 rlist1=r:[1 1 1 1 1];
105 start_note1=50;
106 channel1=0;
107
108
109
110 test-flat-v.mike
111
112 p:[ v4 v4 6b|4#];
113
114 test-floatlist.mike
115
116 [ 1. 1. 1. 1. ];
117
118 test-float.mike
119
120 Printffloat(1.2345);
121
122 test-func-decl-call.mike
123
124
```

```
125 def Name = {Printint(10); Printstr("hello"); a = 5 + 10; a;};
126
127 Printint(Name());
128
129
130 test-func-decl.mike
131
132
133 def Name hello = {Printint(10); Printstr("hello"); a = 5 + 10;
134 a;};
135 test-if.mike
136
137
138 if true then Printint(5) else Printint(10);
139 if false then Printint(5) else Printint(10);
140
141 Printint(15);
142
143 test-index2.mike
144
145 a=[2 3 4 ];
146 c=a.[0+1];
147 Printint(c);
148
149 test-index-print.mike
150
151 a = [1 2 3]; Printint(a.[2]);
152
153 test-int-add.mike
154
155
156 Printint(1 + 10);
157
158
159 test-int.mike
160
161 Printint(5);
162
163
164 test-map-1.mike
165
```

```

166 l=[ 1 2 3 4 ];
167 def A a = a + 2;
168 Map( A l);
169
170 test-pitch-empty.mike
171
172 p:[ ^3#  ^5#  ^8#];
173
174 test-printadd.mike
175
176 Printint(5 + 5);
177
178 test-printFadd.mike
179
180 Printffloat(5. +. 5.);
181
182 test-print-index2.mike
183
184 a=[ 2 3 4 5];
185 c= a.[0+1];
186 Printint(c);
187
188 test-printint.mike
189
190 Printint(5);
191
192 test-print-list.mike
193
194 l=[ 1 2 3 4 5];
195 Printlist( l);
196
197 test-rdot.mike
198
199 r:[ wo  1. 4.5 8o];
200
201 test-rhythm-empty.mike
202
203 r:[ w 1. 3. h q];
204
205 test-rhythm-list-dec.mike
206
207 a = r:[q w];

```

```
208 Prinrlist(a);
209
210 test-scope.mike
211
212 {
213     a=50;
214     Printint(a);
215 };
216 Printint(a);
217
218 test-string.mike
219
220 Printstr("helloworld");
221
222 test-subset.mike
223
224 a=[ 2 3 4 ];
225 c=a.[2];
226 Printint(c);
227
228 test-type.mike
229
230 /* assume print is not different words */
231 a="hello";
232 Print( a );
233 b=8;
234 Print(b);
235 c='c';
236 Print(c);
237 d=[1 2 3];
238 Print(d);
239
240
241 test-unary.mike
242
243 a= -1;
244 PrintInt a;
245
246 fail-pitch-empty.out
247
248
249 test-1synth1.out
```

```
250
251 V1 [61]/1.00 [63]/1.00 [65]/1.00 [66]/1.00 [65]/1.00 [63]/1.00
     [61]/1.00 V2 [64]/1.00 [66]/1.00 [68]/1.00 [69]/1.00 [68]/1.00
     [66]/1.00 [64]/1.00
252
253 test-arith1.out
254
255 4
256 8
257 16
258 5
259
260 test-arith2.out
261
262 2.220000
263 5.500000
264 8.000000
265 32.000000
266 3.500000
267
268 test-array.out
269
270
271 test-assign-array.out
272
273
274 test-assign.out
275
276 5
277 test-assign-print.out
278
279 5
280
281 test-block-return.out
282
283 10
284
285 test-flat-v.out
286
287
288 test-floatlist.out
289
```

```
290
291 test-float.out
292
293 1.234500
294
295 test-func-decl-call.out
296
297 10
298 hello
299 15
300
301 test-func-decl.out
302
303
304 test-if.out
305
306 5
307 10
308 15
309
310 test-index2.out
311
312 3
313
314 test-index-print.out
315
316 3
317 test-int-add.out
318
319 11
320
321 test-int.out
322
323 5
324
325 test-map-1.out
326
327 4 4 4 4
328
329 test-pitch-empty.out
330
331
```

```
332 test-printadd.out
333
334 10
335
336 test-printFadd.out
337
338 10.000000
339
340 test-print-index2.out
341
342 3
343
344 test-printint.out
345
346 5
347
348 test-print-list.out
349
350 1 2 3 4 5
351
352 test-rdot.out
353
354
355 test-rhythm-empty.out
356
357
358 test-rhythm-list-dec.out
359
360 1.000000 4.000000
361
362 test-scope.out
363
364 50
365
366 test-string.out
367
368 helloworld
369
370 test-subset.out
371
372 4
373
```

```
374 test-type.out
375
376 hello
377 8
378
379 fail-arith-1.err
380
381 Fatal error: exception Failure("Mismatched types")
382
383 fail-arith2.err
384
385 Fatal error: exception Failure("Mismatched types")
386
387 fail-array2.err
388
389 Fatal error: exception Failure("Mismatched types")
390
391 fail-array.err
392
393 Fatal error: exception Failure("Mismatched types")
394
395 fail-assign.err
396
397 Fatal error: exception Failure("c Not Found")
398
399 fail-assign-print.err
400
401 Fatal error: exception Failure("a Not Found")
402
403 fail-if1.err
404
405 Fatal error: exception Parsing.Parse_error
406
407 fail-if.err
408
409 Fatal error: exception Failure("Mismatched types")
410
411 fail-illegal-id.err
412
413 Fatal error: exception Parsing.Parse_error
414
415 fail-index.err
```

```
416
417 Fatal error: exception Parsing.Parse_error
418
```

6.3 Test Suites

We wrote tests for every time we implemented a new feature. We also wrote integration tests to make sure that our modules worked in conjunction with each other. For each test, we also wrote an expected output file.

6.4 Test Automation

Our test automation script, testall.sh, runs every test and compares it to the expected output for that test. The testing script is included in the Appendix.

7. Lessons Learned

7.1 Team Reflections

7.1.1 Harvey Wu

This was, by far, the largest project of my programming career, and the first one that involved a lot of teamwork. I learned to communicate often with my team, even if not much progress was being made, if just to touch base and keep each other interested in the project. Safe to say, I also got to know the ins-and-outs of Git pretty well.

I had never touched functional programming before this class and I loved the "if it compiles it works" aspect of OCaml. By implementing a type inference algorithm I also learned a lot about what makes OCaml tick.

Our group spent too much time early on arguing about syntax and design choices that seemed relatively trivial down the road. For future groups: decide on something, and stick to it. Unless it breaks :)

7.1.2 Kaitlin Pet

For the last three years, I have had two reach classes at Columbia- ones whose content I found fascinating but seemed extremely challenging. PLT was one of those classes, am Im glad I finally got to take it as a second semester senior. Because I am concentrating in computer science, this was my first upper-level project-based course and the second major project Ive done at Columbia (the first being a genetic engineering team-based composition that had little to do with computer science). Learning OCaml was an overall fun experience, as someone who hasnt taken Fundamentals the LLVM aspect was a lot more complicated. If music grad school doesnt work out Id definitely like to do more with functional programming. I also liked that you need very little prior information coming in; starting the class I felt on the same page as everyone else. Lastly Id like to thank my teammates, Ive head a lot of horror stories and had a really great experience code grinding with these guys. To future teams, I would suggest 1) bonding with your team early on to feel a sense of connection/obligation to each other, and 2) considering including concentrators because we have hella more time on our hands.

7.1.3 Lakshmi Bodapati

I learned a lot about Git and pull requests and how easily git branches can become messy, unorganized and complicated tree of merge conflicts. I also learned how to solve those problems, maybe not in the Git recommended

way, but some way of getting a working master branch after merging in 4+ branches of conflicting code. I also learned a lot about dividing work and maximizing efficiency when all of us were working together and trying to knock out chunks of the project during our longer weekly meetings. The trickiest part was dividing up work such that everyone had something to do that wasn't blocked by something someone else had to do and every task was being worked on by someone who had some idea of what was necessary for successful completion. The coolest thing I learned was how to write a context-sensitive scanner. I learned the most OCaml while writing the polymorphic function definitions and call matching in the semant module. I also learned that OCaml is quite frosty and nice. For future teams, start early, own features not modules and don't be afraid to OCaml it up!

7.1.4 Husam Abdul-Kafi

I learned a lot about how coding a big group project with a lot of people with varying levels of time and preparedness. It was interesting to work around everyone's schedule and maximize the usage of our time. I love OCaml, and I have no problems with using it for this project. I did a past project where we compiled to a version of Java IR (I think it was called Jade?). That project didn't use anything like the OCAML LLVM Model in that we had to emit every line of code in the IR representation by hand. It was really interesting to compare the two types of IR representation (Java JVM is stack based) and

learn the differences.

Obviously, I'd recommend any future project to have a working version *at least* a week before the final demo so you can spend time optimizing your code and getting edge cases. Don't forget edge cases!!

8. Appendix

05/10/17 11:54:27 C:\Users\husam\OneDrive\PLT\music-mike\src\ast.ml

```
1 (* Abstract Syntax Tree and functions for printing it *)
2
3 type op = Add | FAdd | Sub | FSub | Mult | FMult | Div | FDiv
| Equal | Neq | Less | Leq | Greater | Geq | And | Or
4
5 type preop = Neg | Not | FNeg
6
7 type postop = Rhythmdot
8
9 type typ =
10   TUnit
11 | TInt
12 | TBool
13 | TFloat
14 | TString
15 | TPitch
16 | TChord
17 | TType of string
18 | TList of typ
19 | TFun of typ list * typ
20
21 type expr =
22   Literal of int
23 | FloatLit of float
24 | BoolLit of bool
25 | ID of string
26 | String of string
27 | Binop of expr * op * expr
28 | Preop of preop * expr
29 | Postop of expr * postop
30 | Assign of string * expr
31 | Call of expr * expr list
32 | If of expr * expr * expr
33 | Pitch of int * expr * int
34 | Chord of expr list
35 | Subset of expr * expr
36 | List of expr list
37 | PList of expr list
38 | ChordList of expr list (*PList --> "list of chords"*)
39 | RList of expr list
```

```

40  | Block of expr list
41  | Concat of expr * expr
42  | Noexpr
43  | Fun of string * string list * expr
44  | Unit
45
46 type aexpr =
47  | ALiteral of int * typ
48  | AFloatLit of float * typ
49  | ABoolLit of bool * typ
50  | AID of string * typ
51  | AString of string * typ
52  | ABinop of aexpr * op * aexpr * typ
53  | APReop of preop * aexpr * typ
54  | APStop of aexpr * postop * typ
55  | AAssign of string * aexpr * typ
56  | ACall of aexpr * aexpr list * typ
57  | AIIf of aexpr * aexpr * aexpr * typ
58  | ASubset of aexpr * aexpr * typ
59  | AList of aexpr list * typ
60  | APList of aexpr list * typ
61  | APitch of int * aexpr * int * typ
62  | AChord of aexpr list * typ
63  | AChordList of aexpr list * typ (*PList --> "list of
  chords")
64  | ARList of aexpr list * typ
65  | ABlock of aexpr list * typ
66  | AConcat of aexpr * aexpr * typ
67  | ANoexpr
68  | AFun of string * string list * aexpr * typ (* might be
  some issue with formals as string list *)
69  | AUUnit of typ
70
71 type program = expr list
72
73 type inferred_program = aexpr list
74
75 (* Pretty-printing functions *)
76
77 let string_of_op = function
78   Add -> "+"
79   | FAdd -> "+."

```

```

80     | Sub -> "_"
81     | FSub -> "-."
82     | Mult -> "*"
83     | FMult -> "*."
84     | Div -> "/"
85     | FDiv -> "/."
86     | Equal -> "==""
87     | Neq -> "!="
88     | Less -> "<"
89     | Leq -> "<="
90     | Greater -> ">"
91     | Geq -> ">="
92     | And -> "&&"
93     | Or -> "||"
94
95
96 let string_of_preop = function
97   Neg -> "_"
98   Not -> "!"
99   FNeg -> "_"
100  (*| OctaveUp -> "^"
101  | OctaveDown -> "v" *)
102
103 let string_of_postop = function
104  (*| Sharp -> "#"
105  | Flat -> "b" *)
106  | Rhythmdot -> "o"
107
108 let rec string_of_expr = function
109   Literal(l) -> string_of_int l
110   | FloatLit(f) -> string_of_float f
111   | BoolLit(true) -> "true"
112   | BoolLit(false) -> "false"
113   | ChordList(_) -> "Chordlist"
114   | ID(s) -> s
115   | String(s) -> s
116   | Binop(e1, o, e2) ->
117     string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
118   | Preop(o, e) -> string_of_preop o ^ string_of_expr e
119  (*| Postop(e, o) -> string_of_expr e ^ string_of_postop o *)
120   | Assign(v, e) -> "Assign(" ^ v ^ " = " ^ (string_of_expr e)

```

```

^ ")"
121 | Call(f, el) ->
122   string_of_expr f ^ "(" ^ String.concat ", " (List.map
string_of_expr el) ^ ")"
123 | If(e1, e2, e3) -> "if " ^ string_of_expr e1 ^ " then " ^
string_of_expr e2 ^ " else " ^ string_of_expr e3
124 | Subset(s, i) -> string_of_expr s ^ ".[[" ^ string_of_expr i
^ "]]"
125 | List(es) -> "[ " ^ String.concat " " (List.map
string_of_expr es) ^ " ]"
126
127 (* | ChordList(cs) ->
128   let string_of_chord ps =
129     let string_of_pitch (i1, e, i2) =
130       if i1 < 0 && i2 < 0 then
131         (String.make (abs i1) 'v') ^ (string_of_expr e) ^
(String.make (abs i2) 'b')
132       else
133         if i1 < 0 && i2 >= 0 then
134           (String.make (abs i1) 'v') ^ (string_of_expr e) ^
(String.make (abs i2) '#')
135       else
136         if i1 >= 0 && i2 < 0 then
137           (String.make (abs i1) '^') ^ (string_of_expr e) ^
(String.make (abs i2) 'b')
138       else
139         (String.make (abs i1) '^') ^ (string_of_expr e) ^
(String.make (abs i2) '#')
140       in
141         String.concat "|" (List.map string_of_pitch ps)
142       in
143         "p:[ " ^ String.concat " " (List.map string_of_chord cs) ^
" ]" *)
144
145
146 | RList(es) -> "r:[ " ^ String.concat " " (List.map
string_of_expr es) ^ " ]"
147 | Block(es) -> "{ " ^ String.concat " " (List.map
string_of_expr es) ^ " }"
148 | Concat(e1, e2) -> string_of_expr e1 ^ "@" ^ string_of_expr
e2
149 | Noexpr -> ""

```

```

150    | Unit -> "()"
151    | Fun(n, args, e) -> "Fun " ^ n ^ String.concat " " args ^ "
152      = " " ^ string_of_expr e
153      | _ -> "string_of_expr not implemented for your expression
154      yet."
155
156  let rec string_of_typ = function
157    TInt -> "[int]"
158    TBool -> "[bool]"
159    TFloat -> "[float]"
160    TString -> "[string]"
161    TPitch -> "[pitch]"
162    TUnit -> "[unit]"
163    TType(s) -> "[" ^ s ^ "]"
164    TFun(t1, t2) -> String.concat " " (List.map string_of_typ
165      t1) ^ " -> " ^ string_of_typ t2
166    TList(s) -> string_of_typ s ^ "list"
167
168  let rec string_of_aexpr = function
169    ALiteral(l, t)           -> string_of_int l ^ string_of_typ
170    t
171    | AFloatLit(f, t)        -> string_of_float f ^
172      string_of_typ t
173    | ABoolLit(true, t)       -> "true" ^ string_of_typ t
174    | ABoolLit(false, t)      -> "false" ^ string_of_typ t
175    | AID(s, t)              -> s ^ string_of_typ t
176    | AString(s, t)           -> s ^ string_of_typ t
177    | APitch(i1, ae, i2, t)   ->
178      let signs = (i1 >= 0, i2 >= 0) in
179      (match signs with
180        | false, false ->
181          (String.make (abs i1) 'v') ^ (string_of_aexpr ae) ^
182          (String.make (abs i2) 'b')
183        | false, true  ->
184          (String.make (abs i1) 'v') ^ (string_of_aexpr ae) ^
185          (String.make (abs i2) '#')
186        | true, false ->
187          (String.make (abs i1) '^') ^ (string_of_aexpr ae) ^
188          (String.make (abs i2) 'b')
189        | _ ->
190          (String.make (abs i1) '^') ^ (string_of_aexpr ae) ^
191          (String.make (abs i2) '#'))

```

```

183
184   | ABinop(e1, o, e2, t)    -> string_of_aexpr e1 ^ " " ^
185   string_of_op o ^
186   string_of_typ t
186   | APreop(o, e, t) -> string_of_preop o ^ string_of_aexpr e ^
187   string_of_typ t
187   | APostop(e, o, t) -> string_of_aexpr e ^ string_of_postop o
188   ^ string_of_typ t
188   | AAssign(v, e, t) -> "Assign(" ^ v ^ " = " ^
189   (string_of_aexpr e) ^ ")" ^ string_of_typ t
189   | AFun(id, f, e, t) -> "Function:" ^ "\n" ^ id ^ " " ^
190   String.concat " " f ^ " -> " ^ string_of_aexpr e ^ "\n Type:"
190   ^ string_of_typ t
190   | ACall(f, el, t) ->
191   "Call:" ^ "[" ^ string_of_aexpr f ^ "]" ^ "(" ^
191   String.concat ", " (List.map string_of_aexpr el) ^ ")" ^ ^
191   string_of_typ t
192   | AIf(e1, e2, e3, t) -> "if " ^ string_of_aexpr e1 ^ " then"
192   " ^ string_of_aexpr
193   e2 ^ " else " ^ string_of_aexpr e3 ^ string_of_typ t
194   | ASubset(s, i, t) -> string_of_aexpr s ^ ".[" ^
194   string_of_aexpr i ^ "]" ^ string_of_typ t
195   | AList(es, t) -> begin match t with
196   | TList(TPitch) -> "CHORD"
197   | TList(TList(TPitch)) -> "CHORDLIST"
198   | _ -> "[" ^ String.concat " " (List.map string_of_aexpr
198   es) ^ "]" ^ string_of_typ t
199   end
200   | APList(es, t) -> "p:[ " ^ String.concat " " (List.map
200   string_of_aexpr es) ^ "]" ^ string_of_typ t
201   | ABlock(es, t) -> "{ " ^ String.concat " " (List.map
201   string_of_aexpr es) ^ " }" ^ string_of_typ t
202   | AConcat(e1, e2, t) -> string_of_aexpr e1 ^ "@" ^
202   string_of_aexpr e2 ^ string_of_typ t
203   | ANoexpr -> ""
204   | AUunit(t) -> "(" ^ string_of_typ t
205   | _ -> "[string_of_aexpr not implemented.]"
206
207 let string_of_program (exprs) =
208   "EXPRESSIONS: " ^ String.concat "\n" (List.map
208   string_of_expr exprs) ^ "\n"

```

```
209
210 let string_of_inferred (aexprs) =
211   "INFERRED EXPRS: " ^ String.concat "\n" (List.map
212     string_of_aexpr aexprs) ^
213   "\n"
```

05/10/17 11:54:44 C:\Users\husam\OneDrive\PLT\music-mike\src\codegen.ml

```
1 (*
2 Code generation: translate takes a semantically checked AST
3 and
4 produces LLVM IR
5
6 Detailed documentation on the OCaml LLVM library:
7
8 http://llvm.moe/
9 http://llvm.moe/ocaml/
10 *)
11
12 module L = Llvm
13 module A = Ast
14 module I = Infer
15
16 module StringMap = Map.Make(String)
17
18 let first (a,_,_) = a;;
19 let second (_,a,_) = a;;
20 let third (_,_ ,a) = a;;
21
22 let main_vars:(string, L.llvalue) Hashtbl.t = Hashtbl.create
23 100
23 let function_defs:(string, (L.llvalue * A.aexpr)) Hashtbl.t =
24 Hashtbl.create 100
25
25 (* , functions, structs *)
26 let translate (exprs) =
27   let context = L.global_context () in
28
29   let names:(string, L.llvalue) Hashtbl.t = Hashtbl.create 10
30   in
31     let the_module = L.create_module context "MusicMike"
32
32     and i32_t    = L.i32_type           context (* integer *)
33     and i8_t     = L.i8_type            context (* char? *)
34     and i1_t     = L.i1_type            context (* boole *)
35     and float_t  = L.double_type       context (* float *)
36     and void_t   = L.void_type         context in (* void *)
37     let i8p_t    = L.pointer_type i8_t  in
38
38   (* char
pointer-string*)
```

```

38  let i32p_t = L.pointer_type i32_t in          (* int* *)
39  let i32pp_t = L.pointer_type i32p_t in          (* int** *)
40  let i32ppp_t = L.pointer_type i32pp_t in         (* int*** *)
41  let floatp_t = L.pointer_type float_t in        (* float* *)
42
43  (* int list struct *)
44  let list_t = L.named_struct_type context "list_struct" in
45  L.struct_set_body list_t [| i32_t ; i32p_t |] true;
46  let listp_t = L.pointer_type list_t in
47
48  (* int * list struct *)
49  let chord_struct = L.named_struct_type context
"chord_struct" in
50  L.struct_set_body chord_struct [| i32_t ; i32pp_t |] true;
51  let chord_structp = L.pointer_type chord_struct in
52
53  (* int ** list struct *)
54  let chordlist_struct = L.named_struct_type context
"chordlist_struct" in
55  L.struct_set_body chordlist_struct [| i32_t ; chord_structp
|] true;
56  let chordlist_structp = L.pointer_type chordlist_struct in
57
58  (* float list struct *)
59  let list_t_f = L.named_struct_type context "list_struct_f"
in
60  L.struct_set_body list_t_f [| i32_t ; floatp_t |] true;
61  let listp_t_f = L.pointer_type list_t_f in
62
63  let ltype_of_typ = function
64    A.TInt      -> i32_t
65    | A.TBool     -> i1_t
66    | A.TList(A.TInt)  -> listp_t
67    | A.TList(A.TFloat) -> listp_t_f
68    | A.TList(A.TList(A.TPitch)) -> chordlist_structp
69    (* | A.TVoid     -> void_t *)
70    | A.TFloat    -> floatp_t
71    | A TString   -> i8p_t
72    | A.TUnit     -> void_t
73    | t -> raise (Failure (A.string_of_typ(t) ^ "Shouldn't be
here")) in
74
75  let stype_of_typ = function
76    A.TList(A.TInt) -> (i32_t, list_t)

```

```

77    | A.TList(A.TFloat) -> (float_t, list_t_f)
78    | _ -> raise (Failure "No Struct of this type") in
79
80
81 (* Declare printf(), which the print built-in function will
82   call *)
82  let printf_t = L.var_arg_function_type i32_t [|  

83    L.pointer_type i8_t |] in
83  let printf_func = Ldeclare_function "printf" printf_t
83  the_module in
84
85  let default_fun = L.define_function "main" (L.function_type
85  (ltype_of_typ A.TInt) [||]) the_module in
86  let builder = L.builder_at_end context (L.entry_block
86  default_fun) in
87
88  let int_format_str = L.build_global_stringptr "%d\n" "fmt"
88  builder in
89  let str_format = L.build_global_stringptr "%s\n" "str"
89  builder in
90  let float_format = L.build_global_stringptr "%f\n" "flt"
90  builder in
91  let char_no_line = L.build_global_stringptr "%c" "str"
91  builder in
92  let int_no_line = L.build_global_stringptr "%d" "fmt"
92  builder in
93  let float_no_line = L.build_global_stringptr "%f" "fmt"
93  builder in
94
95 (* Declare the built-in synth() function *)
96  let synth_t = L.function_type i32_t [|i32ppp_t ; i32_t ;
96  i32p_t; i32_t; i32p_t; i32_t; floatp_t; i32pp_t; i32_t; i8p_t
96  |] in
97  let synth_func = Ldeclare_function "synth" synth_t
97  the_module in
98
99  let strcat_t = L.function_type i8p_t [|i8p_t; i8p_t|] in
100  let strcat_func = Ldeclare_function "strcat" strcat_t
100  the_module in
101
102 (*Declare the build-in make_midi() function*)
103  let make_midi_t = L.function_type i32_t [|i8p_t; i8p_t|] in
104  let make_midi_func = Ldeclare_function "make_midi"
104  make_midi_t the_module in

```

```

105      (* get length of struct *)
106      let get_length (struct_obj, sub_builder) = (*
107        print_endline(L.string_of_lltype (L.type_of struct_obj)); *)
108        (* get pointer to length in the struct (at position 0,0) *)
109        let pointer = L.build_in_bounds_gep struct_obj [| L.const_int i32_t 0; L.const_int i32_t 0 |] "length"
110        sub_builder in
111          (* load that pointer to the length *)
112          L.build_load pointer "size" sub_builder
113    in
114    let get_list (struct_obj, sub_builder) =
115      (* get pointer to the int* in the struct (at position 0,1) *)
116      let list_pointer = L.build_in_bounds_gep struct_obj [| L.const_int i32_t 0; L.const_int i32_t 1 |] "cur_list_ptr"
117      sub_builder in
118        (* load that pointer - now act_list is the pointer to the
119          head of the list *)
120        L.build_load list_pointer "cur_list" sub_builder
121    in
122
123    (* s_list is llvalue, application is function taking element
124      of list, index, and builder *)
125    let map s_list application =
126      (* get pointer to length in the struct (at position 0,0) *)
127      let pointer = L.build_in_bounds_gep s_list [| L.const_int i32_t 0; L.const_int i32_t 0 |] "length" builder in
128        (* load that pointer to the length *)
129        let length = L.build_load pointer "size" builder in
130        (* get pointer to the int* in the struct (at position 0,1) *)
131        let list_pointer = L.build_in_bounds_gep s_list [| L.const_int i32_t 0; L.const_int i32_t 1 |] "cur_list_ptr"
132        builder in
133        (* load that pointer - now act_list is the pointer to the
134          head of the list *)
135        let act_list = L.build_load list_pointer "cur_list"
136        builder in
137        (* allocate a pointer to an int (on the stack) *)
138        let cur_index_ptr = L.build_alloca i32_t "cur_index_ptr"
139        builder in

```

```

133      (* store a 0 in that location *)
134      let cur_index = L.build_store (L.const_int i32_t 0)
135      cur_index_ptr builder in
136          (* we are creating blocks, so we need the function
137             we are currently in *)
138          let cur_fun = L.block_parent (L.insertion_block
139             builder) in
140              (* create the block that's supposed to have
141                 cur_index < length
142                 "the conditional block" ==> pred_bb *)
143              let pred_bb = L.append_block context "while"
144              cur_fun in
145                  ignore (L.build_br pred_bb builder);
146
147                  (* create the block of the body - basically
148                     printf act_list[cur_index] *)
149                  let body_bb = L.append_block context "while_body"
150                  cur_fun in
151                      (* body_builder is the builder in the "while
152                         body" *)
153                      let body_builder = L.builder_at_end context
154                      body_bb in
155
156                      (* DO THE WORK ON THE ACTUAL ELEMENTS OF THE LIST
157                         HERE *)
158                          (* loads the value in cur_index_ptr *)
159                          let cur_idx_in_body = L.build_load
160                          cur_index_ptr "cur_indexplz" body_builder in
161                              (* get a pointer into the list at the index
162                                 with the value just loaded *)
163                              let ptr_to_idx = L.build_in_bounds_gep
164                              act_list [| cur_idx_in_body |] "cur_val" body_builder in
165                                  (* load the value at that pointer (aka value
166                                     of act_list[cur_index]) *)
167                                      let val_idx = L.build_load ptr_to_idx
168                                      "val_idx" body_builder in
169                                          (* apply function onto element*)
170                                          ignore(application val_idx cur_idx_in_body
171                                              body_builder);
172                                              (* print_endline("line 169");*)
173                                              (* END WORK HERE *)
174
175                      (* loads the value in cur_index_ptr *)

```

```

162           let cur_index_val = L.build_load cur_index_ptr
163           "cur_index" body_builder in
164             (* add 1 to the value *)
165             let new_idx = L.build_add cur_index_val
166             (L.const_int i32_t 1) "new_idx" body_builder in
167               (* store the new value in the pointer *)
168               ignore(L.build_store new_idx cur_index_ptr
169 body_builder);
170               ignore(L.build_br pred_bb body_builder);
171
172           (* the builder at the "check if cur_index <
173             length" *)
174           let pred_builder = L.builder_at_end context
175             pred_bb in
176             let cur_index_val2 = L.build_load cur_index_ptr
177             "cur_index2" pred_builder in
178               let bool_val = L.build_icmp L.Icmp.Ne length
179               cur_index_val2 "pred" pred_builder in
180
181               let merge_bb = L.append_block context "merge"
182             cur_fun in
183               ignore(L.build_cond_br bool_val body_bb merge_bb
184 pred_builder);
185               L.position_at_end merge_bb builder;
186               in
187
188           let rec expr builder = function
189             A.ALiteral(i, _) -> L.const_int i32_t i
190             | A.AFloatLit(f, _) -> L.const_float float_t f
191             | A.ABoolLit(b, _) -> L.const_int i1_t (if b then 1 else
192 0)
193             | A.ANoexpr -> L.const_int i32_t 0
194             | A.AID(s, _) -> L.build_load (try Hashtbl.find main_vars
195 s with Not_found -> raise(Failure(s ^ " Not Found"))) s
196 builder
197             | A.AString(s, _) -> L.build_global_stringptr s "" builder
198             | A.ABinop (e1, op, e2, _) ->
199               let e1' = expr builder e1
200               and e2' = expr builder e2 in
201                 (match op with
202                   A.Add      -> L.build_add
203                   | A.Sub      -> L.build_sub
204                   | A.Mult     -> L.build_mul
205                   | A.Div      -> L.build_sdiv

```

```

194     | A.FAdd    -> L.build_fadd
195     | A.FSub    -> L.build_fsub
196     | A.FMult   -> L.build_fmul
197     | A.FDiv    -> L.build_fdiv
198     | A.And     -> L.build_and
199     | A.Or      -> L.build_or
200     | A.Equal   -> L.build_icmp L.Icmp.Eq
201     | A.Neq     -> L.build_icmp L.Icmp.Ne
202     | A.Less    -> L.build_icmp L.Icmp.Slt
203     | A.Leq     -> L.build_icmp L.Icmp.Sle
204     | A.Greater -> L.build_icmp L.Icmp.Sgt
205     | A.Geq     -> L.build_icmp L.Icmp.Sge
206   ) e1' e2' "tmp" builder
207
208   | A.APitch(preop, e, postop, _) ->
209     (* allocates single pitch *)
210     let arr_pitch_malloc = L.build_array_malloc i32_t
211     (L.const_int i32_t 3) "array" builder in
212     (* prefield *)
213     let prefield_pointer=L.build_gep arr_pitch_malloc [| 
214     (L.const_int i32_t 0)|] "prefield_elem" builder in
215     let el'=L.const_int i32_t preop in
216     ignore(L.build_store el' prefield_pointer builder);
217
218     (*scale degree *)
219     let sd_pointer=L.build_gep arr_pitch_malloc [| 
220     (L.const_int i32_t 1)|] "scaledegreer_elem" builder in
221     let el'=expr builder e in
222     ignore(L.build_store el' sd_pointer builder);
223
224     (*posfield*)
225     let postfield_pointer=L.build_gep arr_pitch_malloc [| 
226     (L.const_int i32_t 2)|] "postfield_elem" builder in
227     let el'=L.const_int i32_t postop in
228     ignore(L.build_store el' postfield_pointer builder);
229     arr_pitch_malloc
230
231   | A.AList(es, t) -> ( match t with
232     | TList(TList(TPitch)) ->
233       (*allocate struct to hold chordlist and length,
234       struct_c {i32_t, struct_b*} *)
235       let cl_struct=L.build_malloc chordlist_struct
236       "cl_struct" builder in

```

```

232          (* builds pointer for length field in struct to
233             hold length of chordlist *)
234             let c_len_pointer = L.build_in_bounds_gep cl_struct
235               [| L.const_int i32_t 0; L.const_int i32_t 0 |] "length"
236             builder in
237               ignore(L.build_store (L.const_int i32_t
238                 (List.length es)) c_len_pointer builder);
239               (* malloc's an array of type struct_b {i32_t,
240                 i32_t** } *)
241               let arr_malloc = L.build_array_malloc chord_struct
242                 (L.const_int i32_t (List.length es)) "chord_pointer_array"
243                 builder in
244                   (* makes chord struct- length + content *)
245                   (* iterates thru es and builds each chord *)
246
247                   let iter_thru_chord index chord =
248                     let cl2c_pointer = L.build_gep arr_malloc [| |
249                       (L.const_int i32_t index)|] "pointer_chord_elem_list" builder
250                     in
251                     let e' = expr builder chord in
252                     let e_val = L.build_load e' "actual_chord_struct"
253                     builder in
254                     ignore(L.build_store e_val cl2c_pointer
255                     builder);
256                     in
257                     (* iterates through chords with iter_thru_chord
258                     *)
259                     List.iteri iter_thru_chord es;
260                     (* make pointer to chord array in s list *)
261                     let cl_pointer_arr = L.build_in_bounds_gep cl_struct
262                       [| L.const_int i32_t 0; L.const_int i32_t 1 |]
263                       "struct_cl_pointer" builder in
264                         (* fill arr_malloc into pointer to chord list struct
265                         *)
266                         ignore(L.build_store arr_malloc cl_pointer_arr
267                           builder); cl_struct
268
269             | TList(TPitch) -> (*A List of Pitches (aka things
270               separated by | ) *)
271               (* allocates the chord list *)
272               let c_struct=L.build_malloc chord_struct
273                 "chord_struct" builder in
274                   let c_len_pointer = L.build_in_bounds_gep
275                     c_struct [| L.const_int i32_t 0; L.const_int i32_t 0 |]

```

```

"length" builder in
257      ignore(L.build_store (L.const_int i32_t
(List.length es)) c_len_pointer builder);
258      let arr_chord_malloc = L.build_array_malloc i32p_t
(L.const_int i32_t (List.length es)) "arr_pitch" builder in
259      (* ties c_struct to cl_struct *)
260      let deal_with_pitch index el =
261      (*assigns a pointer to the pitch *)
262      let pitch_pointer = L.build_gep arr_chord_malloc
[| (L.const_int i32_t index)|] "pitch_pointer_elem" builder in
263      let e' = expr builder el in
264      (* let e_val = L.build_load e'
"actual_pitch_struct" builder in *)
265      ignore(L.build_store e' pitch_pointer builder);
266      in
267      (* iterates through pitches with deal_with_pitch
*)
268      List.iteri deal_with_pitch es;
269      let c_pointer_arr = L.build_in_bounds_gep c_struct
[| L.const_int i32_t 0; L.const_int i32_t 1 |]
"struct_c_pointer" builder in
270      ignore(L.build_store arr_chord_malloc
c_pointer_arr builder); c_struct
271      | _ ->
272      let s_list = L.build_alloca (snd (stype_of_typ t))
"array_struct" builder in
273      let pointer = L.build_in_bounds_gep s_list [| L.const_int i32_t 0; L.const_int i32_t 0 |] "length" builder
in
274      ignore(L.build_store (L.const_int i32_t
(List.length es)) pointer builder);
275      let arr_alloc = L.build_array_alloca (fst
(stype_of_typ t)) (L.const_int i32_t (List.length es)) "array"
builder
276      in
277      let deal_with_element index e =
278      let pointer = L.build_gep arr_alloc [| L.const_int i32_t index|] "elem" builder in
279      let e' = expr builder e in
280      ignore(L.build_store e' pointer builder)
281      in List.iteri deal_with_element es;
282      let pointer_arr = L.build_in_bounds_gep s_list
[| L.const_int i32_t 0; L.const_int i32_t 1 |] "actual_list"
builder in

```

```

283             ignore(L.build_store arr_alloc pointer_arr
284         builder); s_list
285     )
286
287     | A.ASubset(e1, e2, _) ->
288         let s_list = expr builder e1 in
289         let index = expr builder e2 in
290         let pointer = L.build_in_bounds_gep s_list [| L.const_int i32_t 0; L.const_int i32_t 0 |] "length" builder
291         in
292             let length = L.build_load pointer "size" builder in
293                 let list_pointer = L.build_in_bounds_gep s_list [| L.const_int i32_t 0; L.const_int i32_t 1 |] "cur_list_ptr"
294                     builder in
295                         let act_list = L.build_load list_pointer "cur_list"
296                             builder in
297                             let pointer_to_element = L.build_gep act_list [| index
298                                 |] "pointer_to_element" builder in
299                                 L.build_load pointer_to_element "tmp" builder
300
301             | A.ABlock(es, t) -> ( match es with
302                 e::e1::rest -> ignore(expr builder e); expr builder
303                 (A.ABlock(e1::rest, t))
304                 | [e] -> expr builder e)
305             | A.APreop(op, e, _) ->
306                 let e' = expr builder e in
307                 (match op with
308                     A.Neg      -> L.build_neg
309                     | A.Not     -> L.build_not
310                     ) e' "tmp" builder
311             (*
312                 | A.AMuPreop(op, e, _) ->
313                     (* given pointer to pitch array, memory operations for
314                     adding or subtracting to position 0 of pitch element *)
315                     (* index is needed so map will accept it *)
316                     let interior_operation index pitch builder1 =
317                         let prefld_pointer=L.build_gep pitch [| (L.const_int
318                             i32_t 0)|] "prefield_elem" builder1 in
319                         let cur_prefld = L.build_load prefld_pointer
320                             "cur_prefld" builder1 in
321                             (match op with
322                                 AАОup ->
323                                     let new_prefld = L.build_add cur_prefld

```

```

(L.const_int i32_t 1) builder1
316      A.A0down ->
317          let new_prefield = L.build_sub cur_prefield
318          (L.const_int i32_t 1) builder1
319      ) in
320      ignore(L.build_store new_prefield prefield_pointer
321      builder); in
322      (* match different things mupreops could be applied to,
323      there are 3 *)
324      (match e with
325      | A.APitch ->
326          let e' = expr builder e in
327              interior_operation e' builder
328      | A.AChord ->
329          let e' = expr builder e in
330              map (get_list e' builder) interior_operation
331
332      | A.AChordlist ->
333          let e' = expr builder e in
334      )
335
336      *)
337      | A.AAssign (s, e, t) -> let e' = expr builder e in
338          let var = try Hashtbl.find main_vars s
339              with Not_found ->
340                  let local_var = L.build_alloca
341          (ltype_of_typ t) s builder in
342              Hashtbl.add main_vars s local_var;
343          local_var in
344              ignore (L.build_store e' var builder); e'
345      (* | A.ACcall (A.AID("Map", _), act, _) ->
346          let func = expr builder (List.hd act) in
347          let lst = expr builder (List.hd (List.tl act)) in
348          let wrapper f=f in
349          map lst (wrapper func); L.const_int i32_t 1
350      *)
351      | A.ACcall (A.AID("Printint", _), [e], _) ->
352          L.build_call printf_func [| int_format_str ; (expr
353          builder e) |] "printf" builder
354      | A.ACcall (A.AID("Printstr", _), [e], _) ->

```

```

353     L.build_call printf_func [| str_format; (expr builder
354     e) |] "printf" builder
355     | A.ACall (A.AID("Printfloat", _), [e], _) ->
356         L.build_call printf_func [| float_format; (expr builder
357     e) |] "printf" builder
358     | A.ACall (A.AID("Printlist", _), [e], _) ->
359         let printfun value index builder = L.build_call
360         printf_func [|int_no_line ; value |] "printf" builder in
361             let s_list= expr builder e in
362                 map s_list printfun; L.const_int i32_t 1
363             | A.ACall(A.AID("Prinrlist", _), [e], _) ->
364                 let printfun value index builder = L.build_call
365                 printf_func [|float_no_line ; value |] "printf" builder in
366                     let s_list= expr builder e in
367                         map s_list printfun; L.const_int i32_t 1
368
369             | A.ACall(A.AID("Make_midi", _), [e1; e2], _) ->
370                 (* ignore(L.build_call printf_func [| str_format; (expr
371                 builder e1) |] "printf" builder);
372                     L.build_call printf_func [| str_format; (expr builder
373                 e2) |] "printf" builder
374 *)   let new_str = L.build_array_malloc i8_t (L.const_int
375                 i32_t 1000) "new_string" builder in
376                 let ptr_to_first = L.build_in_bounds_gep new_str [|  

377                     L.const_int i32_t 0 |] "first" builder in
378                     ignore(L.build_store (L.const_int i8_t 0) ptr_to_first
379                         builder);
380                     ignore(L.build_call strcat_func [| new_str; (expr
381                         builder e1) |] "put_first" builder);
382                     ignore(L.build_call strcat_func [| new_str; (expr
383                         builder e2) |] "put_first" builder);
384                     new_str
385
386             (* assumed order of acutals: pitchlist, rhythmlist,
387             modelist, start note, channel num *)
388             | A.ACall (A.AID("Synth", _), act, _) ->
389                 (*extract the actuals *)
390                 let clist = expr builder (List.hd act) in
391                 let clist_len = get_length (clist, builder) in
392                 let rlist = expr builder (List.hd (List.tl act)) in

```

```

384      let act_rlist = get_list(rlist, builder) in
385      let modelist = expr builder (List.hd (List.tl (List.tl
386      act))) in
387      let act_modelist = get_list(modelist, builder) in
388      let mode_len = get_length(modelist, builder) in
389      let start_pitch = expr builder (List.hd (List.tl (List.tl
390      (List.tl act)))) in
391      let channel = expr builder
392      (List.hd(List.tl(List.tl(List.tl(List.tl act))))) in
393      (*build the nessesary structures to pass into c function -
394      plist as non-struct int***, list of chord lengths, return-arr
395      *)
396
397      (*malloced structure that contains lengths of chords *)
398      let chord_lengths = L.build_array_malloc i32_t clist_len
399      "return_arr" builder in
400      (*malloced structure that normalized pitch array (no
401      octaves or accidnetals) will be built into. This is passed
402      into C synth function *)
403      let clear_cl_list = L.build_array_malloc i32p_t
404      clist_len "return_arr" builder in
405      (*building non-struct chord : Note that this refers to
406      both the normal builder and the builder inside the while loop
407      (builder1)*)
408      let passed_cl_list =L.build_array_malloc i32pp_t clist_len
409      "norm_arr" builder in
410      let chord_func value1 index builder1=
411      (* for chord_lengths *)
412      let pointer_to_ret_elem = L.build_in_bounds_gep
413      passed_cl_list [| index |] "cur_val" builder1 in
414      let new_elem_list = L.build_extractvalue value1 1
415      "stuff" builder1 in
416      let chord_len_pointer = L.build_in_bounds_gep
417      chord_lengths [| index |] "len" builder1 in
418      let new_elem_len = L.build_extractvalue value1 0
419      "oldlen" builder1 in
420      let clear_list_pointer = L.build_in_bounds_gep
421      clear_cl_list [| index |] "len" builder1 in
422      let new_clear_arr = L.build_array_malloc i32_t
423      new_elem_len "clear_cl_list_elem" builder1 in
424      ignore(L.build_store new_clear_arr
425      clear_list_pointer builder1);
426      ignore(L.build_store new_elem_len
427      chord_len_pointer builder1);

```

```

408         ignore(L.build_store new_elem_list
409         pointer_to_ret_elem builder1);
410             in
411
412         map clist (* (get_list(clist, builder)) *) chord_func;
413
414
415         (* build buffer *)
416         let buff = L.build_array_malloc i8_t (L.const_int i32_t
417         1000) "synth-buffer" builder in
418             (* call synth *)
419             ignore(L.build_call synth_func [| (* int ***
420             *)passed_cl_list; (* int *)clist_len;
421             (* int ***)chord_lengths; (* int *) start_pitch; (*
422             int * *)act_modelist;
423             (* int *)mode_len; (* double * *)act_rlist; (* int
424             ** *)clear_cl_list; (* int *)channel; (* char * *) buff |]
425             "synth" builder);
426             buff
427
428
429
430         | A.ACall (A.AID(s, _), act, _) ->
431             let (fdef, fdecl) = Hashtbl.find function_defs s in
432                 let actuals = List.rev (List.map (expr builder)
433                 (List.rev act)) in
434                     let result = (match fdecl with
435                         A.AFun(f, _, _, _) -> f
436                         | _ -> raise(Failure "second problem with call") ) in
437                         L.build_call fdef (Array.of_list actuals) result
438                         builder
439
440         | A.AIf(e1, e2, e3, _) ->
441             let bool_val = expr builder e1 in
442                 let cur_fun = L.block_parent (L.insertion_block
443                 builder) in
444                     let merge_bb = L.append_block context "merge"
445                     cur_fun in
446
447                         let then_bb = L.append_block context "then" cur_fun
448                         in
449                             ignore(expr (L.builder_at_end context then_bb) e2);
450                             ignore(L.build_br merge_bb (L.builder_at_end context

```

```

        then_bb));
440
441         let else_bb = L.append_block context "else" cur_fun
442     in
443         ignore(expr (L.builder_at_end context else_bb) e3);
444         ignore(L.build_br merge_bb (L.builder_at_end context
445             else_bb));
446         ignore(L.build_cond_br bool_val then_bb else_bb
447             builder);
448         L.position_at_end merge_bb builder;
449         L.const_int i32_t 1
450
451     | A.AFun(fid, arg_list, e, A.TFun(arg_types, ret_type)) ->
452
453         let formal_types = Array.of_list (List.map ltype_of_typ
454             arg_types) in
455         let ftype = L.function_type (ltype_of_typ ret_type)
456             formal_types in
457         let the_function = L.define_function fid ftype
458             the_module in
459
460         Hashtbl.add function_defs fid
461             (the_function, A.AFun(fid, arg_list, e,
462                 A.TFun(arg_types, ret_type)));
463         let builder2 = L.builder_at_end context (L.entry_block
464             the_function) in
465         let alloc_local (s, t, p) =
466             L.set_value_name s p;
467         let local_var = L.build_alloca (ltype_of_typ t) s
468             builder2 in
469             ignore(Hashtbl.add main_vars s local_var);
470             ignore(L.build_store p local_var builder2)
471         in
472         let rec iter3 (f, l1, l2, l3) =
473             (match (l1, l2, l3) with
474                 (hd1::rest1, hd2::rest2, hd3::rest3) -> f(hd1, hd2,
475                     hd3); ignore(iter3(f, rest1, rest2, rest3)))
476                 | ([] , [], []) -> ()
477                 | _ -> print_endline "ERROR LINE 491";
478             ) in
479             iter3 (alloc_local, arg_list, arg_types,
480                 (Array.to_list((L.params the_function))));
481             let ret_val = expr builder2 e in

```

```
470         L.build_ret ret_val builder2
471
472         | _ -> L.const_int i32_t 1
473         in
474         let exprbuilder builder e = ignore(expr builder e);
475         builder
476         in
477         let builder = List.fold_left exprbuilder builder
478         (List.rev(exprs))
479         in
480         ignore (L.build_ret (L.const_int i32_t 0) builder);
481         the_module
```

05/10/17 11:55:00 C:\Users\husam\OneDrive\PLT\music-mike\src\infer.ml

```
1  open Ast
2  open Lib
3
4  module StringMap = Map.Make(String)
5  module StringSet = Set.Make(String)
6  type environment = typ StringMap.t
7  type constraints = (typ * typ) list
8
9  let letter = ref (Char.code 'a');
10 let new_type () = let c1 = !letter in
11   incr letter; TType(Char.escaped (Char.chr c1))
12 ;;
13
14 let kws = ["if"; "then"; "else"; "true"; "false"; "def"]
15 ;;
16
17 let keywords =
18   List.fold_left (fun set x -> StringSet.add x set)
19   StringSet.empty kws
20 ;;
21 let rec annotate_expr exp env : (aexpr * environment) =
22   match exp with
23   | Unit      -> AUnit(TUnit), env
24   | Literal(n) -> ALiteral(n, TInt), env
25   | FloatLit(n) -> AFLOATLit(n, TFloat), env
26   | BoolLit(n) -> ABoolLit(n, TBool), env
27   | String(n)  -> AString(n, TString), env
28   | ID(n)       -> if StringMap.mem n env then
29                     AID(n, StringMap.find n env), env
30                     else raise(Failure(n ^ " Not Found"))
31   | Binop(e1, op, e2) ->
32     let ae1, _ = annotate_expr e1 env
33     and ae2, _ = annotate_expr e2 env
34     and ntyp = new_type () in
35     ABinop(ae1, op, ae2, ntyp), env
36   | Preop(preop, e) ->
37     let ae, _ = annotate_expr e env
38     and ntyp = new_type () in
39     APREOP(preop, ae, ntyp), env
40   | Postop(e, postop) ->
```

```

41      let ae, _ = annotate_expr e env
42      and ntyp = new_type () in
43      APostop(ae, postop, ntyp), env
44  | Assign(name, e) ->
45      if StringMap.mem name env then
46          raise (Failure "Reassignment")
47      else if StringSet.mem name keywords then
48          raise (Failure "Redefining keyword")
49      else let ntyp = new_type () in
50      let nenv = StringMap.add name ntyp env in
51      let ae, _ = annotate_expr e nenv in
52      AAssign(name, ae, ntyp), nenv
53  | List(e_list) ->
54      let ae_list = List.map (fun e -> fst (annotate_expr e
55      env)) e_list in
56      AList(ae_list, TList(new_type ())), env
56  | RList(e_list) ->
57      let ae_list = List.map (fun e -> fst (annotate_expr e
58      env)) e_list in
59      ALst(ae_list, TList(TFloat)), env
59  | Pitch(i1, e, i2) ->
60      let ae, _ = annotate_expr e env in
61      APitch(i1, ae, i2, TPitch), env
62  | Block(e_list) ->
63      let ae_list, nenv = ListLabels.fold_left ~init: ([] , env)
63      e_list
64          ~f: (fun (ae_list, env) e -> let ae, env =
64          annotate_expr e env in (ae::ae_list, env))
65          in ABlock(ae_list, new_type ()), nenv
66  | Chord(e_list) ->
67      let ae_list = List.map (fun e -> fst (annotate_expr e
68      env)) e_list in
69      ALst(ae_list, TList(TPitch)), env
69  | ChordList(e_list) ->
70      let ae_list = List.map (fun e -> fst (annotate_expr e
71      env)) e_list in
72      AList(ae_list, TList(TList(TPitch))), env
72  | Call(func, args) ->
73      let a_func, _ = annotate_expr func env in
74      let a_args = List.map (fun arg -> fst (annotate_expr arg
75      env)) args in
76          ACall(a_func, a_args, new_type ()), env
76  | If(pred, e1, e2) ->

```

```

77      let apred, _ = annotate_expr pred env
78      and e1, _ = annotate_expr e1 env
79      and e2, _ = annotate_expr e2 env in
80      AIf(apred, e1, e2, new_type()), env
81 | Fun(name, args, e) ->
82   if StringMap.mem name predefined
83   then raise (Failure "Cannot redefine stdlib function.");
84   let args_t = List.map (fun f -> new_type()) args
85   and ret_t = new_type() in
86   let fun_t = TFun(args_t, ret_t) in
87   let a_args = List.combine args args_t in
88   let nenv = List.fold_left
89     (fun e (id, t) ->
90      if StringMap.mem id e
91      then raise (Failure "Variable already defined")
92      else StringMap.add id t e) env a_args
93   in
94   if StringMap.mem name env then
95     raise (Failure "Redefining function")
96   else let nenv = StringMap.add name fun_t nenv in
97   let ae, _ = annotate_expr e nenv in
98   AFun(name, args, ae, fun_t), nenv
99 | Subset(var, e) ->
100  let avar, _ = annotate_expr var env
101  and ae, _ = annotate_expr e env
102  and t = new_type() in
103  let typ = t in
104  ASubset(avar, ae, t), env
105 | _ -> AUnit(TUnit), env
106;;
107
108 let type_of ae =
109   match ae with
110   | AUnit(t)          -> t
111   | ALiteral(_,t)    -> t
112   | AFloatLit(_,t)   -> t
113   | AString(_,t)     -> t
114   | ABoolLit(_,t)    -> t
115   | AID(_,t)         -> t
116   | ABinop(_,_,_,t) -> t
117   | APreop(_,_,t)    -> t
118   | APostop(_,_,t)  -> t
119   | AAssign(_,_,t)   -> t

```

```

120  | ACall(_, _, t)      -> t
121  | ABlock(_, t)       -> t
122  | AFun(_, _, _, t)  -> t
123  | AList(_, t)        -> t
124  | AIf(_, _, _, t)   -> t
125  | ASubset(_, _, t)   -> t
126  | APitch(_, _, _, t) -> t
127  | _                   -> print_string "[Missed a type in
type_of]"; TUnit
128 ;;
129
130
131 let rec collect_expr ae : constraints =
132   match ae with
133   | ALiteral(_)          -> []
134   | ABoolLit(_)           -> []
135   | AFloatLit(_)          -> []
136   | AString(_)            -> []
137   | AUnit(_)              -> []
138   | AID(_)                -> []
139   | ABinop(ae1, op, ae2, t) ->
140     let t1 = type_of ae1
141     and t2 = type_of ae2 in
142     let con = match op with
143       | Add | Sub | Mult | Div -> [(t1, TInt); (t2, TInt); (t,
TInt)]
144       | FAdd | FSub | FMult | FDiv -> [(t1, TFloat); (t2,
TFloat); (t, TFloat)]
145       | Neq | Equal | Greater | Less | Geq | Leq -> [(t1, t2);
(t, TBool)]
146       | And | Or -> [(t1, TBool); (t2, TBool); (t, TBool)]
147     in
148     (collect_expr ae1) @ (collect_expr ae2) @ con
149
150   | AAssign(_, ae, t) -> (collect_expr ae) @ [(t, type_of ae)]
151   | ABlock(ae_list, t) ->
152     let ret = List.hd (List.rev ae_list) in
153     let ret_t = type_of ret in
154     (List.flatten (List.map collect_expr ae_list)) @ [(t,
ret_t)]
155   | AList(ae_list, t)   ->
156     let list_t = match t with
157       | TList(s) -> s

```

```

158      | _ -> raise (Failure "Unreachable state in List
literal") in
159      let con = List.map (fun aexpr -> (list_t, type_of aexpr))
ae_list in
160      (List.flatten (List.map collect_expr ae_list)) @ con
161      | APitch(i1, ae, i2, t) -> [(type_of ae, TInt)]
162      | AIf(pred, ae1, ae2, t) ->
163          let pt = type_of pred and t1 = type_of ae1 and t2 =
type_of ae2 in
164          let con = [(pt, TBool); (t1, t2); (t, t1)] in
165          (collect_expr pred) @ (collect_expr ae1) @ (collect_expr
ae2) @ con
166
167      | AFun(_, _, ae, t) -> begin match t with
168          | TFun(_, ret_t) -> (collect_expr ae) @ [(type_of ae,
ret_t)]
169          | _ -> raise (Failure "Unreachable state in Function
literal") end
170
171      | ASubset(v, e, typ) ->
172          let vt = (match v with
173              | AID(_) -> type_of v
174              | _ -> raise (Failure "Unreachable state in Subset")) )
in
175          let s = match vt with
176              | TList(t) -> [(t, typ)]
177              | TType(t) -> [(vt, TList(typ))]
178              | _ -> raise (Failure "Subset can only be applied to
lists")
179          in
180          (collect_expr e) @ [(type_of e, TInt)] @ s
181
182
183      | ACall(name, args, t) ->
184          let fnt = (match name with
185              | AID(_) -> type_of name
186              | _ -> raise (Failure "Unreachable state in Call")) ) in
187          let s = match fnt with
188              | TFun(args_t, ret_t) ->
begin
189                  let l1 = List.length args and l2 = List.length
args_t in
190                  if l1 <> l2

```

```

192         then raise (Failure "Mismatched argument count")
193     else let args_c = List.map2 (fun ft at -> (ft,
194 type_of at)) args_t args in
195         args_c @ [(t, ret_t)]
196     end
197     | TType(_) -> [(fnt, TFun(List.map type_of args, t))]
198     | _ -> print_endline (string_of_typ fnt); raise (Failure
199 "Mismatched type")
200     in
201     (collect_expr name) @ (List.flatten (List.map collect_expr
202 args)) @ s
203
204 | e -> raise (Failure ("collect_expr can't handle your expr
205 yet" ^ string_of_aexpr e))
206 ;;
207
208 let rec substitute u x t =
209   match t with
210   | TUnit | TInt | TBool | TFloat | TPitch | TString -> t
211   | TType(c) -> if c = x then u else t
212   | TFun(t1, t2) -> TFun(List.map (substitute u x) t1,
213 substitute u x t2)
214   | TList(t) -> TList(substitute u x t)
215 ;;
216
217 let apply subs t =
218   List.fold_right (fun (x, u) t -> substitute u x t) subs t
219 ;;
220
221 let rec unify constraints =
222   match constraints with
223   | [] -> []
224   | (x, y) :: tl ->
225     let t2 = unify tl in
226     let t1 = unify_one (apply t2 x) (apply t2 y) in
227     t1 @ t2
228
229 and unify_one t1 t2 =
230   match t1, t2 with
231   | TInt, TInt | TBool, TBool | TString, TString
232   | TUnit, TUnit | TFloat, TFloat | TPitch, TPitch -> []
233   | TType(x), z | z, TType(x) -> [(x, z)] (* Not completely
234 correct *)

```

```

229   | TList(t1), TList(t2) -> unify_one t1 t2
230   | TFun(u, v), TFun(x, y) ->
231     let l1 = List.length u and l2 = List.length x in
232     if l1 = l2 then unify ((List.combine u x) @ [(v, y)]) (*
233       Double check if
234         args are correct *)
235     else raise (Failure "Mismatched Argument Count")
236   | _ -> raise (Failure "Mismatched types")
237 ;;
238 let rec apply_expr subs ae =
239   match ae with
240   | AUnit(t)           -> AUnit(apply subs t)
241   | ALiteral(value, t) -> ALiteral(value, apply subs t)
242   | AFLOATLit(value, t) -> AFLOATLit(value, apply subs t)
243   | ABoolLit(value, t)  -> ABoolLit(value, apply subs t)
244   | AString(value, t)   -> AString(value, apply subs t)
245   | ABinop(ae1, op, ae2, t) ->
246     ABinop(apply_expr subs ae1, op, apply_expr subs ae2,
247             apply subs t)
247   | AID(name, t)        -> AID(name, apply subs t)
248   | APitch(a, ae, b, t) ->
249     APitch(a, apply_expr subs ae, b, apply subs t)
250   | AAssign(name, ae, t) ->
251     AAssign(name, apply_expr subs ae, apply subs t)
252   | AList(ae_list, t)    ->
253     AList(List.map (apply_expr subs) ae_list, apply subs t)
254   | AFun(name, frmls, ae, t) ->
255     AFun(name, frmls, apply_expr subs ae, apply subs t)
256   | AIf(pred, ae1, ae2, t) ->
257     AIf(apply_expr subs pred, apply_expr subs ae1,
258          apply_expr subs ae2, apply subs t)
259   | ACall(fname, args, t) ->
260     ACall(apply_expr subs fname,
261           List.map (apply_expr subs) args, apply subs t)
262   | ABlock(ae_list, t)    ->
263     ABlock(List.map (apply_expr subs) ae_list, apply subs t)
264   | ASubset(var, i, t)    ->
265     ASubset(apply_expr subs var, apply_expr subs i, apply
266             subs t)
266   | e -> raise (Failure ("No apply_expr for AEXPR:" ^
267                         string_of_aexpr e))
267 ;;

```

```

268
269 let infer expr env flag =
270   let aexpr, nenv = annotate_expr expr env in
271   let constraints =
272     if flag then
273       print_endline ("AEXPR: " ^ string_of_aexpr aexpr);
274       collect_expr aexpr in
275   let subs =
276     List.iter (fun (a,b) ->
277       if flag then
278         print_endline
279         ("CONSTRAINTS: " ^ string_of_typ a ^ " " ^ string_of_typ b)) constraints;
280     unify constraints in
281   let inferred_expr =
282     List.iter (fun (a,b) -> if flag then
283       print_endline ("SUBS: " ^ a ^ " " ^ string_of_typ b)) subs;
284     apply_expr subs aexpr in
285     if flag then
286       print_endline("FINAL: " ^ string_of_aexpr
287 inferred_expr);
287   inferred_expr, nenv
288 ;;
289
290 let typecheck program flag : (aexpr list) =
291   let env = Lib.predefined in
292   let inferred_program, _ = ListLabels.fold_left (List.rev
293 program)
294   ~init: ([] , env)
295
296   ~f: (
297     fun (acc, env) expr ->
298       let inferred_expr, env = infer expr env flag in
299       let inferred_expr, env = match inferred_expr with
300       | AAssign(name, _, t) ->
301           let env = StringMap.add name t env in
302           inferred_expr, env
303       | AFun(name, _, _, t) ->
304           let env = StringMap.add name t env in
305           inferred_expr, env
306       | _ -> inferred_expr, env in

```

```
307           (inferred_expr :: acc, env)
308       )
309   in (* List.rev *) inferred_program
310 ;;
312
```

05/10/17 11:55:14 C:\Users\husam\OneDrive\PLT\music-mike\src\lib.ml

```
1  open Ast
2
3  module StringMap = Map.Make(String)
4
5  let stdlib =
6    ("Printint", TFun([TInt], TString));
7    ("Printstr", TFun([TString], TString));
8    ("Printfloat", TFun([TFloat], TString));
9    ("Printlist", TFun([TList(TInt)], TString));
10   ("Prinrlist", TFun([TList(TFloat)], TString));
11   ("Synth", TFun( [TList(TList(TPitch)); TList(TFloat);
12     TList(TInt); TInt; TInt], TString));
13   ("Make_midi", TFun( [TString; TString], TUnit));
14   ("Merge", TFun( [TString; TString] , TString));
15 ;;
16 let predefined =
17   List.fold_left (fun env (id, t) -> StringMap.add id t env)
18   StringMap.empty stdlib
19 ;;
```

05/10/17 11:55:28 C:\Users\husam\OneDrive\PLT\music-mike\src\make_midi.c

```
1 #include <unistd.h>
2
3 int make_midi(char * buffer, char * name){
4     exec1("./testCFugueLib", "./testCFugueLib", buffer, name,
5           (char *)0);
6     return 0;
7 }
```

05/10/17 11:55:40 C:\Users\husam\OneDrive\PLT\music-mike\src\Makefile

```
1 # Make sure ocamlbuild can find opam-managed packages: first
2 run
3 #
4 # eval `opam config env`
5
6 # Easiest way to build: using ocamlbuild, which in turn uses
7 ocamlfind
8
9 musicmike.native :
10    ocamlbuild -ocamlyacc "ocamlyacc -v" -use-ocamlfind -pkgs
11    llvm,llvm.analysis -cflags -w,+a-4 \
12        musicmike.native
13
14 # "make clean" removes all generated files
15 .PHONY : clean
16 clean :
17    ocamlbuild -no-log -clean
18    rm -rf testall.log *.diff musicmike scanner.ml parser.ml
19    parser.mli
20    rm -rf synth make_midi
21    rm -rf *.cmx *.cmi *.cmo *.cmx *.o *.s *.ll *.out *.exe
22    *.output
23
24 parser:
25     ocamlyacc -v parser.mly
26
27 scanner:
28     ocamllex scanner.mll
29
30 frontend:
31     ocamllex scanner.mll
32     ocamlc -c ast.ml
33     ocamlyacc -v parser.mly
34     ocamlc -c parser.mli
35     ocamlc -c lib.ml
36
37
38 # More detailed: build using ocamlc/ocamlopt + ocamlfind to
```

```

locate LLVM
37
38 OBJS = ast.cmx codegen.cmx infer.cmx lib.cmx parser.cmx
      scanner.cmx semant.cmx musicmike.cmx
39
40 musicmike : $(OBJS)
41     ocamlfind ocamlopt -linkpkg -package llvm -package
      llvm.analysis $(OBJS) -o musicmike
42
43 scanner.ml : scanner.mll
44     ocamllex scanner.mll
45
46 parser.ml parser.mli : parser.mly
47     ocamllyacc parser.mly
48
49 %.cmo : %.ml
50     ocamlc -c $<
51
52 %.cmi : %.mli
53     ocamlc -c $<
54
55 %.cmx : %.ml
56     ocamlfind ocamlopt -c -package llvm $<
57
58 # Synth from microC
59 synth : synth.c
60         cc -o synth -DBUILD_TEST synth.c
61 make_midi : make_midi.c
62         cc -o make_midi -DBUILD_TEST make_midi.c
63 ### Generated by "ocamldep *.ml *.mli" after building
      scanner.ml and parser.ml
64 ast.cmo :
65 ast.cmx :
66 codegen.cmo : ast.cmo
67 codegen.cmx : ast.cmx
68 musicmike.cmo : semant.cmo scanner.cmo parser.cmi codegen.cmo
      ast.cmo infer.cmo
69 musicmike.cmx : semant.cmx scanner.cmx parser.cmx codegen.cmx
      ast.cmx infer.cmx
70 parser.cmo : ast.cmo parser.cmi
71 parser.cmx : ast.cmx parser.cmi
72 scanner.cmo : parser.cmi
73 scanner.cmx : parser.cmx

```

```
74 semant.cmo : ast.cmo
75 semant.cmx : ast.cmx
76 infer.cmo : ast.cmo lib.cmo
77 infer.cmx : ast.cmx lib.cmx
78 lib.cmo: ast.cmo
79 lib.cmx: ast.cmx
80 parser.cmi : ast.cmo
81
```

05/10/17 11:55:53 C:\Users\husam\OneDrive\PLT\music-mike\src\musicmike.ml

```
1 (* Top-level of the MusicMike compiler: scan & parse the input,
2    check the resulting AST, generate LLVM IR, and dump the
3    module *)
4 type action = Ast | LLVM_IR | Sast | Compile | Semant
5
6 let _ =
7   let action = if Array.length Sys.argv > 1 then
8     List.assoc Sys.argv.(1) [("-t", Sast);(* Print the AST
9     only *)
10      (" -a", Ast);
11      (" -l", LLVM_IR); (* Generate LLVM, don't
12      check *)
13      (" -c", Compile);
14      (" -s", Semant)] (* Generate, check LLVM IR *)
15   else Compile in
16 let lexbuf = Lexing.from_channel stdin in
17 let ast = Parser.program Scanner.next_token lexbuf in
18 let sast =
19   match action with
20   | Ast -> []
21   | Sast -> Infer.typecheck ast true
22   | Semant -> Semant.check (Infer.typecheck ast true)
23   | _ -> Semant.check (Infer.typecheck ast false) in
24
25   match action with
26   | Ast -> print_string (Ast.string_of_program ast)
27   | Sast -> print_string (Ast.string_of_inferred sast)
28   | LLVM_IR -> print_string (Llvm.string_of_llmodule
29     (Codegen.translate sast))
30   | Semant -> print_string ("SEMANT DEBUGGING :" ^
31     Ast.string_of_inferred (List.rev sast))
32   | Compile -> let m = Codegen.translate sast in
33     Llvm_analysis.assert_valid_module m;
34     print_string (Llvm.string_of_llmodule m)
```

05/10/17 11:56:04 C:\Users\husam\OneDrive\PLT\music-mike\src\parser.mly

```
1 %{
2     open Ast
3 %}
4
5 %token SEMI LPAREN RPAREN LBRACE RBRACE COMMA LBRACKET
RBRACKET PLBRACKET RLBRACKET LTUPLE RTUPLE
6 %token OUP ODOWN FLAT OCTOTHORPE RHYTHMDOT DOTLBRACKET BAR
7
8 %token PLUS MINUS TIMES DIVIDE ASSIGN NOT FPLUS FMINUS FTIMES
FDIVIDE CONCAT
9 %token EQ NEQ LT LEQ GT GEQ TRUE FALSE AND OR
10 %token IF THEN ELSE WHILE INT BOOL VOID
11 %token TYP DEF FOR
12 %token <int> LITERAL
13 %token <float> FLITERAL
14 %token <string> STRING
15 %token <string> ID FID
16 %token EOF
17
18 %right ASSIGN
19 %right call
20 %right IF
21 %left OR
22 %left AND
23 %left EQ NEQ
24 %nonassoc LT GT LEQ GEQ
25 %left PLUS MINUS FPLUS FMINUS
26 %left TIMES DIVIDE FTIMES FDIVIDE
27 %left OUP ODOWN FLAT OCTOTHORPE RHYTHMDOT
28 %right RBRACKET
29 %left LBRACKET
30 %left CONCAT
31 %right NOT NEG
32
33 %start program
34 %type <Ast.program> program
35
36 %%
37
38 /* "A program consists of a list of statements, aka `stmts`"*/
39
40 program:
```

```

41
42     stmts EOF           { $1 }
43
44
45
46 /* "stmts is a tuple with the first field being a list of
   expressions (expr),
47 the second field being a list of function declarations
   (fdecl), and the
48 third field being a list of type declaratiosn (tdecl)" */
49
50 stmts:
51             { [] }
52 | stmts expr SEMI { $2 :: $1 }
53
54
55 /* "A function declaration `fdecl` consists of
56      a keyword 'Def'
57
58      a Function Identifier `FID` - string w/ first letter
      capitalized
59      a list of formals `formals_list`
60      a body which consists of an `expr` expression"*/
61
62 fdecl:
63     DEF FID formals_list ASSIGN expr { Fun($2, List.rev($3),
   $5) }
64
65 /* "expressions always return a value and consists of:
66      literals-basic types
67      binop-binary operator
68      unop-unary operators
69      primaries-miscelaneous pool (list type, assignment, etc.
   */
70
71 expr:
72     literals { $1 }
73 | binop    { $1 }
74 | unop    { $1 }
75 | primaries { $1 }
76 | fdecl    { $1 }
77 | LPAREN expr RPAREN { $2 }
78
79 literals:

```

```

80      LITERAL          { Literal($1) }
81  | FLITERAL          { FloatLit($1) }
82  | TRUE              { BoolLit(true) }
83  | FALSE             { BoolLit(false) }
84  | LPAREN RPAREN    { Unit }
85  | ID                { ID($1) }
86  | STRING            { String($1) }

87
88
89
90 binop:
91  | expr PLUS         expr { Binop($1, Add,      $3) }
92  | expr MINUS        expr { Binop($1, Sub,      $3) }
93  | expr TIMES         expr { Binop($1, Mult,     $3) }
94  | expr DIVIDE        expr { Binop($1, Div,      $3) }
95  | expr FPLUS          expr { Binop($1, FAdd,    $3) }
96  | expr FMINUS         expr { Binop($1, FSub,    $3) }
97  | expr FTIMES         expr { Binop($1, FMult,   $3) }
98  | expr FDIVIDE        expr { Binop($1, FDiv,    $3) }
99  | expr EQ              expr { Binop($1, Equal,   $3) }
100 | expr NEQ             expr { Binop($1, Neq,    $3) }
101 | expr LT              expr { Binop($1, Less,    $3) }
102 | expr LEQ             expr { Binop($1, Leq,    $3) }
103 | expr GT              expr { Binop($1, Greater, $3) }
104 | expr GEQ             expr { Binop($1, Geq,    $3) }
105 | expr AND             expr { Binop($1, And,    $3) }
106 | expr OR              expr { Binop($1, Or,     $3) }

107
108 unop:
109 /*| MINUS expr %prec NEG { Preop(Neg, $2) } */
110 | NOT expr           { Preop(Not, $2) }

111
112
113
114 primaries:
115  /* "Block of expressions" */
116
117  LBRACE semi_list RBRACE { Block($2) }
118  /* "Calling a function" */
119  | FID LPAREN actuals_list RPAREN { Call(ID($1),
List.rev($3)) }
120  /* "Assigning a value to an variable" */
121  | assign               { $1 }
122  /* "list of expressions of same type (enforced in

```

```

semant.ml)" */
123 | LBRACKET expr_list RBRACKET { List(List.rev($2)) }
124 /* "list of chords" */
125 | PLBRACKET pxpr_list RBRACKET { ChordList(List.rev($2)) }
126 /*"list of rhythms"*/
127 | RLBRACKET expr_list RBRACKET { RList(List.rev($2)) }
128 /* "tuple of expressions with different types (enforced
in semant.ml)" */
129 /* | LTUPLE expr_list RTUPLE { Tuple($2) } */
130 /* "concatanating 2 lists (enforced in semant.ml)" */
131 | expr CONCAT expr { Concat($1, $3) }
132 /* "If, then else "*/
133 | IF expr THEN expr ELSE expr
134 %prec IF
135 { If($2, $4, $6) }
136 /* "getting an element from a list/tuple/pitchlist" */
137 | ID DOTLBRACKET expr RBRACKET { Subset(ID($1), $3) }
138
139
140
141 /* "Assigning a value to an variable"*/
142 assign:
143 ID ASSIGN expr { Assign($1, $3) }
144
145 /*" List of assingments (a=b) used in type declaration " */
146
147 assign_list:
148
149 | assign { [$1] }
150 | assign_list assign { $2 :: $1 }
151
152
153 /* "List of whitespace separated expressions used in
   -Lists
   -Tuples" */
154
155 expr_list:
156
157 /*nothing*/ { [] }
158 | expr_list expr { $2 :: $1 }
159
160
161
162 /* "List of semicolon separated expressions used in block" */
163
164 semi_list:

```

```

165     expr SEMI           { [ $1 ] }
166   | semi_list expr SEMI { $2 :: $1 }
167
168
169
170 /* "List of formal arguments used in function declaration"
 */
171
172 formals_list:
173   /*nothing*/          { [ ] }
174   | formals_list ID    { $2 :: $1 }
175
176
177 /* "List of actual arguments used in function calls" */
178
179 actuals_list:
180   /*nothing*/          { [ ] }
181   | actuals_list expr  { $2 :: $1 }
182
183
184 /* "List of whitespace separated chords(simultaneous pitches)
185 used in Plist (pitch list) "*/
186
187 pxpr_list:
188   chord                { [ Chord($1) ] }
189   | pxpr_list chord    { Chord($2) :: $1 }
190
191
192 /* "List of simultaneous pitches" */
193
194
195 chord:
196   pitch                { [ $1 ] }
197   | chord BAR pitch    { $3 :: $1 }
198
199
200 /*p:[3|5|6 3  ^^3#|9bb]*/
201
202
203 /* "Tuple consisting of 3 fields:
204    prefld-a list of ints representing '^' and 'v' as '1'
205    and '-1'
206    an int representing scale degree inputed by user
207    postfld-a list of ints representing '#' and 'b' as '1'
208    */

```

```

and '-1' */
207
208 pitch:
209     prefield expr postfield { Pitch($1, $2, $3) }
210
211
212 /*"a list of ints representing '^' and 'v' as '1' and '-1'
   respectively" */
213
214 prefield:
215 /* nothing */           { 0 }
216 | prefield OUP          { $1+1 }
217 | prefield ODOWN         { $1-1 }
218
219
220 /* "a list of ints representing '#' and 'b' as '1' and '-1'
   *//
221
222 postfield:
223 /*nothing*/
224 | postfield OCTOTHORPE  { $1+1 }
225 | postfield FLAT         { $1-1 }
226

```

05/10/17 11:56:30 C:\Users\husam\OneDrive\PLT\music-mike\src\semant.ml

```
1 (* Semantic checking for the MicroC compiler *)
2
3 open Ast
4 open Infer
5 module StringMap = Map.Make(String)
6
7 (* Hack for polymorphism compilation: take each polymorphic
   function and check when it is called.
8 * Create a new Function Aexpr for each time it is called, with
   the specific type of the actuals. *)
9
10 let check (aexprs: aexpr list) =
11     let is_poly ae = match ae with
12         | AFun(_, _, _, TFun(f_t, r_t)) ->
13             let poly t = match t with
14                 | TType(_) -> true
15                 | _ -> false
16             in
17             List.exists poly f_t
18         | _ -> false
19     in
20     let poly = List.filter is_poly aexprs in
21     let getname ae = match ae with
22         | AFun(fn, _, _, _) -> fn
23         | _ -> raise (Failure "What the hell you doin")
24     in
25     let poly_fnames = List.map getname poly in
26     let rec is_call ae = match ae with
27         | [] -> []
28         | AAssign(_, ie, _)::rest -> (match ie with
29             | ACall(AID(fn, t), args, rt) -> let name = fn in
30                 if List.mem name poly_fnames then
31                     ACall(AID(fn, t), args, rt)::(is_call rest)
32                 else (is_call rest)
33                 | _ -> is_call rest)
34         | ACall(AID(fn, t), args, rt)::rest -> let name = fn in
35             if List.mem name poly_fnames then
36                 ACall(AID(fn, t), args, rt)::(is_call rest)
37             else (is_call rest)
38         | x::rest -> is_call rest
39     in
```

```

40  let polycalls = is_call aexprs in
41  let rec matching x lst =
42    match lst
43    with [] -> []
44    | ACall(AID(x,t), a, b)::rest -> ACall(AID(x,t), a, b)::(matching x rest)
45    | y::rest -> matching x rest
46    in
47    let poly t = match t with
48      | TType(_) -> true
49      | _ -> false
50    in
51  let rec iterAexprs alist =
52    match alist with
53    [] -> []
54    | AFun(fn, a, b, TFun(ts, ret_t))::rest ->
55      if List.exists poly ts then
56        let callmatches = matching fn polycalls in
57        let typelist cm =
58          (match cm with
59            ACall(_, c, _) -> List.map Infer.type_of c
60            | _ -> [])
61        in
62        let calltofun cm1 = AFun(fn, a, b, TFun(typelist cm1,
63          Infer.type_of cm1)) in
64          (List.map calltofun callmatches)@(iterAexprs rest)
65        else
66          AFun(fn, a, b, TFun(ts, ret_t))::(iterAexprs rest)
67        | w::rest -> w::(iterAexprs rest)
68  in iterAexprs aexprs;;

```

05/10/17 11:56:40 C:\Users\husam\OneDrive\PLT\music-mike\src\synth.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <ctype.h>
4 #include <string.h>
5 #include <unistd.h>
6 #include <errno.h>
7
8
9 int ***fold_lists ( int ***chord_list, int cl_length, int
10 chord_lengths[],
11 int start_pitch, int *modelist, int mode_length){
12 //fprintf(stderr,"%s\n", "entering chord list");
13 //map the mode to absolute pitches (0 corresponds to first
14 //scale degree)
15 int i=0;
16 while (i<mode_length){
17     modelist[i]=modelist[i]+start_pitch;
18     //fprintf(stderr,"%s %d\n", "new mode value = ",
19     modelist[i]);
20     i+=1;
21 }
22 //runs off assumption that malloced chunks are not contiguous
23 int j=0;
24 while (j<cl_length){
25     int ** chord= chord_list[j];
26     int i=0;
27     //fprintf(stderr,"%s %d\n", "chord number", j);
28     while (i<chord_lengths[j]){
29         //fprintf(stderr,"%s %d\n", "into while loop", i);
30         //fprintf(stderr,"%p\n", chord);
31         int * pitch= chord[i];
32         //fprintf(stderr,"%s %d\n", "pitch number", i);
33         //fprintf(stderr,"%s\n", "int *pitch=chord[i];");
34         //new pitch to be added
35         int transformed_pitch;
36         int p=pitch[1];
37         if (p==0){
38             transformed_pitch=0;
39         }
```

```

38     else{
39         int oup=(p-1)/mode_length;
40         if (oup>0){
41             //add 'overflow' to octaveup
42             pitch[0]+=oup;
43         }
44         transformed_pitch=modelist[(p-1)%mode_length];
45     }
46     pitch[1]=transformed_pitch;
47
48     i++;
49 }
50     j++;
51 }
52 //fprintf(stderr,"%s\n", "after while loop");
53 return chord_list;
54 }
55
56 //takes normalized chord list and spits out list with actual
57 //pitches
58 int ** apply_accidentals (int ***chordlist, int cl_len, int
59 *chord_lengths, int mode_length, int **return_arr, int * mode)
60 {
61     int octave=mode[mode_length-1]-mode[0];
62
63     int j=0;
64     while (j<cl_len){
65         int ** chord= chordlist[j];
66         int i=0;
67         while (i<chord_lengths[j]){
68             int * pitch= chord[i];
69
70             if (pitch[1]!=0){
71                 //add or subtract octaves
72                 int octave_shift=pitch[0]*octave;
73                 pitch[1]=pitch[1]+octave_shift;
74                 //add or subtract accidentals
75                 pitch[1]=pitch[1]+pitch[2];
76             }
77             (return_arr[j])[i]=pitch[1];
78         }
79         i++;
80     }

```

```

77             j++;
78         }
79
80     return return_arr;
81 }
82
83
84
85 //string generator, takes rhythm list plus absolute pitch list
86 //and turns into strings that can be plopped in Cfugue
86 //gonna mix in some c++ lets see if it crashes :/
87 int strgen (char * buff, double * rhythmlist, int **
88 corrected_chordlist, int cl_len, int * chord_lengths, int
89 channel){
90     //first add channel to beginning
91     char v[3];
92     strcpy(v, "V");
93     strcat(buff, v);
94     char channel_buff[3];
95     sprintf(channel_buff, 3, "%d", channel);
96     strcat(buff, channel_buff);
97     char space[2];
98     strcpy(space, " ");
99     strcat(buff, space);
100    int j=0;
101    while (j<cl_len){
102        //take note_len and convert into string
103        double note_len= rhythmlist[j];
104        //fprintf(stderr, "note_len: %.2f\n", note_len);
105        char snote_len[10];
106        memset(snote_len, '\0', sizeof(snote_len));
107        sprintf(snote_len, 10, "% .2f", note_len);
108        //initialize chord
109        int * chord= corrected_chordlist[j];
110        int i=0;
111        while (i<chord_lengths[j]){
112            int pitch= chord[i];
113            //convert pitch into string
114            char pitchstring[3];
115            sprintf(pitchstring, 3, "%d", pitch);
116            //buffer all the goddamn symbols
117            char lbracket[2];

```

```

116         strcpy(lbracket, "[");
117         char rbracket[3];
118         strcpy(rbracket, "]/");
119         char plus[2];
120         strcpy(plus, "+ ");
121         char space[2];
122         strcpy(space, " ");
123         char rest[3];
124         strcpy(rest, "R/");
125         //make acutal pitchstring
126         if (pitch==0){
127             strcat(buff, rest);
128         }
129         else{
130             strcat(buff, lbracket);
131             strcat(buff, pitchstring);
132             strcat(buff, rbracket);
133         }
134             strcat(buff, snote_len);
135
136         if (i<chord_lengths[j]-1){
137             strcat(buff, plus);
138             }
139         else{
140             strcat(buff, space);
141             }
142         i++;
143         }
144         j++;
145     }
146
147     return 0;
148
149
150 }
151
152
153
154 //synth- imitates behavior of main(), compared at end
155
156 int synth(int *** chordlist, int len_chordlist, int *
chord_lengths,

```

```

157     int start_pitch, int * modelist, int mode_length, double
158     *rhythmlist,
159     int **pure_chord_arr, int channel, char * buff){
160     //fprintf(stderr,"%s\n", "in synth");
161     int *** new_chordlist = (int ***) malloc(len_chordlist *
162     sizeof(int **));
163     int j1=0;
164     while (j1<len_chordlist){
165         //fprintf(stderr, "%d\n", j1);
166         //fprintf(stderr, "%s\n", "LINE 162");
167         int ** chord= chordlist[j1]; //old
168         int ** new_chord = (int **)
169             malloc(chord_lengths[j1]*sizeof(int *));
170         new_chordlist[j1]=new_chord;//stuff in
171         int i=0;
172         while (i<chord_lengths[j1]){
173             //fprintf(stderr, "%s\n", "LINE 167");
174             int *pitch= chord[i];//old
175             int *new_pitch = (int *) malloc(3*sizeof(int));
176             new_chord[i]=new_pitch;
177             new_pitch[0] = pitch[0];
178             new_pitch[1] = pitch[1];
179             new_pitch[2] = pitch[2];
180             //fprintf(stderr, "%s\n", "LINE 173");
181             //fprintf(stderr, "%s\n", "LINE 175");
182             i++;
183         }
184         j1++;
185     }
186     int *new_modelist = (int *) malloc(mode_length *
187     sizeof(int *));
188     int j2 = 0;
189     while(j2 < mode_length) {
190         new_modelist[j2] = modelist[j2];
191         j2++;
192     }
193     // int i = 0;
194     // while (i<len_chordlist){

```

```

195 //         int **temp=(int
196 //             **)(malloc(chord_lengths[i]*sizeof(int *)));
197 //             fprintf(stderr,"%s\n", "fuck pointers");
198 //             new_chordlist[i]=temp;
199 //             fprintf(stderr,"%s\n", "pointers are cooeilo");
200 //             int j=0;
201 //             while (j<chord_lengths[i]){
202 //                 fprintf(stderr,"%s %d\n", "r = ", r);
203 //                 int* temp2=(int *) malloc(3*sizeof(int));
204 //                 temp[j] = temp2;
205 //                 // (chordlist[i])[j]=temp2;
206 //                 int* pitch=(chordlist[i])[j];
207 //                 pitch[1]=r;
208 //                 pitch[2]=c;
209 //                 r++;
210 //                 j++;
211 //             }
212 //         }
213
214
215
216 // int j=0;
217 // while (j<len_chordlist){
218 //     int ** chord= chordlist[j];
219 //     fprintf(stderr,"%p\n", chord);
220 //     int i=0;
221 //     while (i<chord_lengths[j]){
222 //         int * pitch= chord[i];
223 //         fprintf(stderr,"\t%p\n", pitch);
224 //         fprintf(stderr,"\t\t%d\n", pitch[0]);
225 //         fprintf(stderr,"\t\t%d\n", pitch[1]);
226 //         fprintf(stderr,"\t\t%d\n", pitch[2]);
227
228 //         i++;
229 //     }
230 //     j++;
231 // }
232 //modifies chordlist so mode is normalize to absolute
233 // value of notes. If range goes above octave, adds to prefield
234     int ***new_list = fold_lists(new_chordlist, len_chordlist,
chord_lengths, start_pitch, new_modelist, mode_length);

```

```

234     //fprintf(stderr,"%s\n", "AFTER NEW LIST");
235     int j=0;
236     while (j<len_chordlist){
237         int ** chord= new_list[j];
238         //fprintf(stderr,"%p\n", chord);
239         int i=0;
240         while (i<chord_lengths[j]){
241             int * pitch= chord[i];
242                 //fprintf(stderr,"\t%p\n", pitch);
243                 //fprintf(stderr,"\t\t%d\n", pitch[0]);
244                 //fprintf(stderr,"\t\t%d\n", pitch[1]);
245                 //fprintf(stderr,"\t\t%d\n", pitch[2]);
246
247                 i++;
248             }
249             j++;
250         }
251         //copies new_list into pure_chord_list to incorporate
252         octaves and accidentals (yes, I know a new int ** is redundant
253         but atm just want to see if works
254         //fprintf(stderr,"%s\n", "after new_list");
255         int **correct_pitches=apply_accidentals(new_list,
256         len_chordlist, chord_lengths, mode_length, pure_chord_arr,
257         new_modelist);
258         //fprintf(stderr,"%d\n", chord_lengths[0]);
259         //takes rhythm list and turns into string that can be fed
260         // into CFugue
261         memset(buff, '\0', 900);
262         strgen (buff, rhythmlist, correct_pitches,
263         len_chordlist, chord_lengths, channel );
264         fprintf(stderr,"buff %s\n", buff);
265
266
267 // //tester
268 // int main(){
269

```

```

270
271
272 //      //variablesle we need
273 //      int ***chordlist;
274 //      int cl_len=4;
275 //      int chord_lengths[4]={2,2,2,2};
276 //      int start_pitch=10;
277 //      int modelist[4]={1,3,5,7};
278 //      int mode_length=4;
279 //      double rhythmlist[]={1, 1.5, 0.25, 0.33};
280 //      //build chordlist
281 //      chordlist=(int ***)malloc(4 * sizeof(int **));
282 //      int r=0; //pitch literal value
283 //      int c=-1; //accidental value
284 //      int i=0; //number of chords
285 //      while (i<4){
286 //          int **temp=(int **)malloc(2*sizeof(int *));
287 //          fprintf(stderr,"%s\n", "fuck pointers");
288 //          chordlist[i]=temp;
289 //          fprintf(stderr,"%s\n", "pointers are cooeilo");
290 //          int j=0;
291 //          while (j<2){
292 //              fprintf(stderr,"%s %d\n", "r = ", r);
293 //              int* temp=(int *) malloc(3*sizeof(int));
294 //              (chordlist[i])[j]=temp;
295 //              int* pitch=(chordlist[i])[j];
296 //              pitch[1]=r;
297 //              pitch[2]=c;
298 //              r++;
299 //              j++;
300 //          }
301 //          i++;
302 //      }
303
304
305 //      fprintf(stderr,"%s\n", "chord list was created");
306 //      //
307 //      int temp1[2];
308 //      int temp2[2];
309 //      int temp3[2];
310 //      int temp4[2];
311 //      int *modarr[4]={ temp1, temp2, temp3, temp4};

```

```

312
313 //      //testing synth
314 //          synth(chordlist, cl_len, chord_lengths,
315 //          start_pitch, modelist, mode_length, rhythmlist, modarr );
315
316 //      //let's see...
317 //      int ***new_list = fold_lists(chordlist, cl_len,
318 //          chord_lengths, start_pitch, modelist, mode_length);
318 //      //print it!
319 //      int j=0;
320 //      while (j<cl_len){
321 //          int ** chord= new_list[j];
322 //          fprintf(stderr,"%p\n", chord);
323 //          int i=0;
324 //          while (i<chord_lengths[j]){
325 //              int * pitch= chord[i];
326 //              fprintf(stderr,"\t%p\n", pitch);
327 //              fprintf(stderr,"\t\t%d\n", pitch[0]);
328 //              fprintf(stderr,"\t\t%d\n", pitch[1]);
329 //              fprintf(stderr,"\t\t%d\n", pitch[2]);
330
331 //                  i++;
332 //          }
333 //          j++;
334 //      }
335
336 //      int **correct_pitches=apply_accidentals(new_list,
337 //          cl_len, chord_lengths, mode_length, modarr);
337
338
339 //      j=0;
340 //      while (j<cl_len){
341 //          int * chord= correct_pitches[j];
342 //          fprintf(stderr,"%p\n", chord);
343 //          int i=0;
344 //          while (i<chord_lengths[j]){
345 //              int pitch= chord[i];
346 //              fprintf(stderr,"\t%d\n", pitch);
347 //              i++;
348 //          }
349 //          j++;
350 //      }

```

```
351
352
353 // //testing adding rhythm to the whole shebang
354 //     char buff[14* 8+1];
355 //     buff[0]='\0';
356 //     strgen (buff, rhythmlist, correct_pitches, cl_len,
357 //             chord_lengths );
358 //         fprintf(stderr,"%s\n", buff);
359
360 // //testing aggregate synth funtion
361
362 // return 0;
363 // }
```