Project Proposal:

Advanced Arithmetic Language

COMS W4115

UNI: jc3783
Jimin Choi

Overview & Motivation

To understand compiler/interpreter concepts learned from COMS W4115, I decided to implement an advanced arithmetic language. Unlike a basic 4 function calculator, this language will support advanced mathematical operations such as roots, powers, sequences, and summations, etc. This language will support basic programming language functionalities such as if-else statement blocks, loops, and variables. As I listen to more lectures, I will be able to have more concrete scope of this project.

I believe that implementing mathematical language is the best way to practice programming language concepts learned from class, as it will fit nicely to lexical analysis and parsing structures. Like other languages supporting mathematical operations, users will have ability to implement algorithms with this language. This language will be very useful for anyone, as it will provide lightweight Matlab like functionalities. Working on this project will give me valuable learning experiences.

As mandated by professor Edwards, I will implement all language features and compiler/interpreters in Ocaml. I found that learning Ocaml language itself is a very challenging work, and I couldn't get a handle of it at all until today. The syntax of functional language and using recursions for loops are too cryptic to me. These are my big concerns that are not being addressed.

Features

- Mathematical Operations (+,-,*,/,%, <, >, <=, >=, ==, sqrt, power, root, sum, avg, etc), trigonometric operators will not be supported.
- Variables, the scope can be either global or local
- If/else statement
- for/while loop
- Functions
- Variable data types – int, char, word(string), integer array (only number array is allowed as this is a mathematical language)

Syntax /Sample program

```
function int getMinimum: array data
        int minimum = 0;
        for t = 0 ; t < data.size ; t=t+1
                if minimum > data[t]
                   minimum = data[t];
                endif;
        endfor;
return minimum;
endfunction;
```

A function starts with a "function" keyword. Following data type specifies the return type of a function. "getMinimum" is a function name. The function arguments are followed by ':'. This is similar to Objective C's way of defining function arguments.

In this sample, the argument was an integer array. "data" is an array name. The next line shows the integer variable declaration. For loop syntax is similar to C type language's other than not having a parenthesis. Unlike C language, the end of function, if-else, and loop are indicated by endfunction, endif, and endfor respectively instead of using brackets. Any variable defined outside of any "function" blocks are considered as global variables. Each statement will end with semicolon.