Justin Petluk

Ryan Ozar

Chris Perlette

Ryan Bauert

**Project Summary:**

For this assignment we created a rudimentary programmable calculator. The calculator can evaluate basic arithmetic expressions following standard order of operations (2+3\*4 evaluates to 14, not to 20 as it would if simply evaluated left to right). It also allows you to define variables and assign them values either with an input statement, an arithmetic evaluation and other variables (myVar = 3+3 sets myVar to 6, myVar2 = myVar + 2 sets myVar2 to 8) . It also lets you define your own functions and then call them should you wish to execute. It will also evaluate Boolean expressions as per the rules of those expressions.

**Difficulties:**

The primary difficulty came in storing data and executing functions. Racket does not have any nice to use data structures like vectors to store an ever expanding list of variables in, and so what we ended up doing is storing all the data in a single list and using a step size of 3 to allow us to have the variable name, type, and value stored next to each other to make it easier to access it all. We debated using parallel lists, but this is what we settled on. Another major difficulty was creating a parser to evaluate equations with.

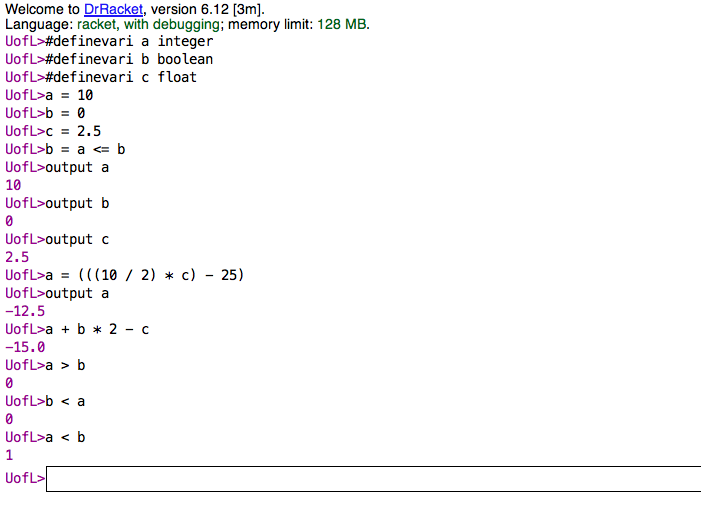
For the main functions like #definevari we simply used cond statements and if’s to check the function, however this didn’t work very well for arithmetic evaluation and so we decided to use a lexical parser to make the cascading equations work. Another problem we had was trying to create our own parser. We wanted to create our own parser but proved to be much to complicated given our time frame. We ended up going with the included racket parser library, and while it did work, it introduced several limitations. Due to a relatively low amount of time to learn the library we didn’t, and still don’t, fully understand how it works (mostly for more complex functions). This made it so that while we were able to evaluate both arithmetic and Boolean equations, we were unable to make the evaluations work with variables of any kind outside of assignment.

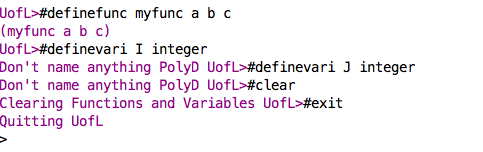
**Key Data Structures:**

The only major data structures we used were two lists for storing the user defined variables and functions. These were names globalVariStack and globalFuncStack respectively. The lists contained both the names of the variables and functions and the required data for both of those things.

**Limitations:**

Currently we do not have conditional (if) statements and for loops working.

**Input Statements: **

****