Racket Interpreter – CPSC 3740 Final Project

Nathan Tipper & Vincent Cote

April 1st, 2019

For this project, we were tasked with creating a Racket interpreter using the racket language. To use the interpreter, the user calls startEval accompanied by a Racket program. This report will highlight the organization of the interpreter, data structures used, bugs or issues we encountered, and discuss how we tested our program.

# Organization

To implement our Racket interpreter we decided to use a list to keep track of the variables, in order to use this list we needed to add a couple helper functions. These helper functions are addBinding and findBinding, we also made the startEval function call myEval with the given program as well as this initial list of bindings. The “brain” of this interpreter is the myEval function, this function handles the parsing of the given program and calls the appropriate function based on what the program is asking for. To handle the lambda functions we also created a helper function to extract the variables and add them to our list.

# Data Structures

In order to keep track of our variables we decided to use a list in a similar fashion as a stack, adding new variables as needed. We decided to add the variables to the beginning of the list so that when searching through the list the first result we get is the one in scope. To search through the list we are simply using a linear search iterating from the beginning of the list to the end.

# Bugs and Limitations

# Testing