CT331 Assignment 2

Functional​ ​Programming​ ​with​ ​Scheme

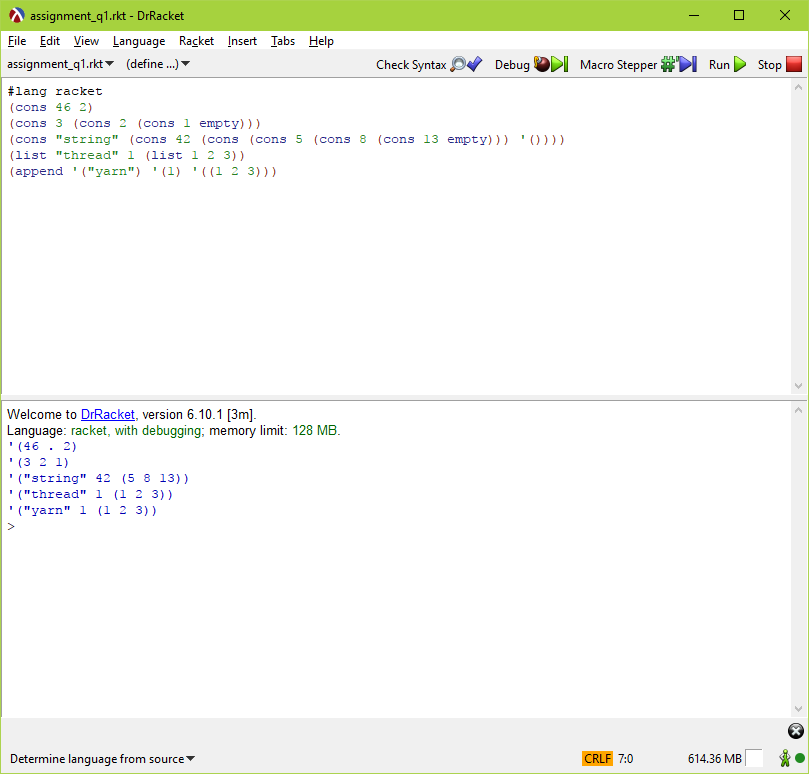
Andrew Reid East

16280042

C:\Users\deskase\AppData\Local\Microsoft\Windows\INetCache\Content.Word\fa-github-square.png <https://github.com/reideast/ct331_assignment2>

# Question 1

## Code Editor and Command Line Output



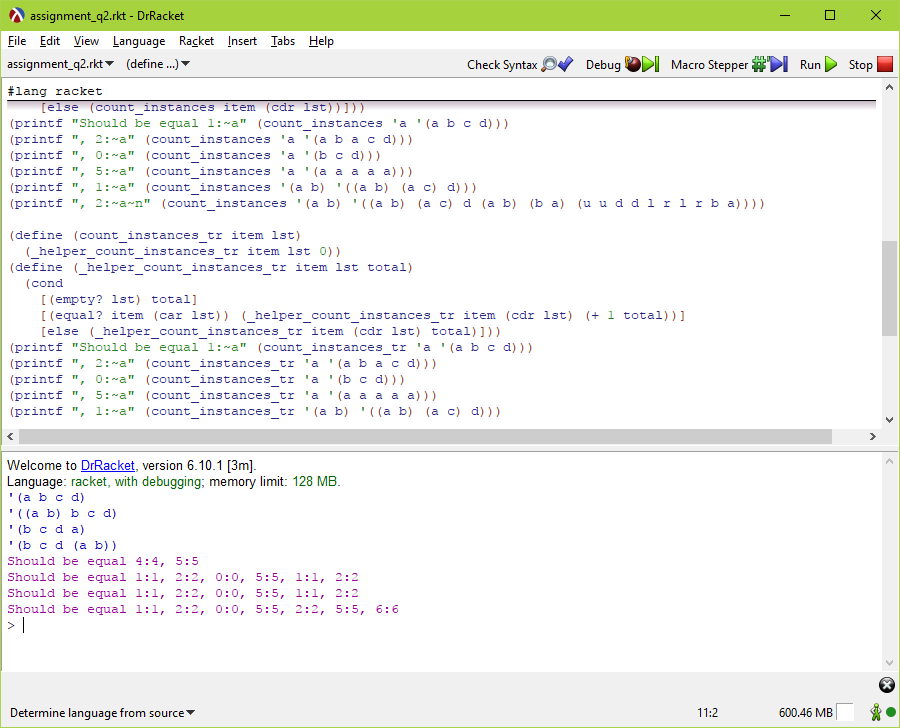
## Comments

The lists being made up of nested cons pairs seems similar to recursion, and I’m curious if the Racket interpreter evaluates using order-of-operation rules in one stack frame, or actually just fires off each in a recursive manner. Considering that Lisp itself is so recursive, I wonder how the compiler’s inner workings handle things.

Cons takes two s-expressions and combines then into a pair, which is in itself a single s-expression. The list function takes a series of zero or more s-expressions and combines them into a list, which is shown in the command line output as a single unit, but is really a shorthand for a more complex s-expression (a nested series of pairs, with the second item of the innermost pair being the empty value). The append function also builds a list, but each of its arguments must also be a list, and in contrast to the list function, each argument list is chained onto the previous, which makes one long list of all those elements. Finally, while the cons function always takes two values, both list and append can take any number of values.

# Question 2

## Code Editor and Command Line Output



# Question 3

## Code Editor and Command Line Output