CSI 2120 – Lab Task 1

Abdorrahim Bahrami

Introduction to Scheme

Your first task in this lab is to get yourself familiarized with Scheme and do some simple programming in it. Make sure you have DrRacket installed, and you are familiar with the interpreter and the coding parts. If you need help, ask your TA to help you with this.

Then, you should do the following programming tasks. Each programming task in the lab is a design based on the subjects you learned during lectures. There are test samples that you can use to test your program. If you have questions, ask your TAs.

Important Note: Using ChatGPT during the lab time is strictly forbidden, if your TA catch you using it, you will receive 0 for all lab tasks immediately. During lab time, you can only use online references mentioned in class.

Task 1:

In this task, you are going to try some evaluations using the scheme interpreter. First, try some simple evaluations on your own to warm up. Then, try to evaluate the following expressions in scheme.

(7 + 13 * 22) – 51 / 64 * (19 – 45 / (32 + 11))

Remember that this is an infix notation, and you have to write it in prefix notation in Scheme. Be careful about the priorities of the operators.

As we discussed in class, we don't have a global variable in Scheme because of its side effects. But we do have a global constant. Note that constants do not have side effects because their values cannot be changed.

We can define a constant as follows

(define PI 3.1415)

Try it in Scheme and then evaluate the following trigonometric formulas using the definition you just did for some angles such as $\pi, \frac{\pi}{4}, \frac{\pi}{2}$, etc. Note that *sin* and *cos* are built-in functions in Scheme for computing Sin and Cos. Evaluate each side of the equality for some angles and see if they give you the same result.

$Sin(x)^2 + Cos(x)^2 = 1$

$Sin(2x) = 2Sin(x)Cos(x)$

$Cos(2x) = Cos(x)^2 - Sin(x)^2$

Task 2:

As we mentioned in class, we need to write a lot of recursive functions in Scheme. In this task, you should write two classic examples of recursive functions in Scheme, which are factorial and power. It is better if you use if-then-else function of Scheme.

(define (fact n)

)

Here are some test cases to test your function.

(fact 5) ==> 120

(fact 7) ==> 5040

(define (power x y)

)

Here are some test cases to test your function.

(power 4 5) ==> 1024

(power 3 6) ==> 729

Task 3 (Challenge): Using the factorial function above, try to write a lambda expression that calculates $x! + y!$.

(define factSum

Here are some test cases to test your lambda expression.

(factSum 2 3) ==> 8

(factSum 5 6) ==> 840