CSI 2120 – Lab Task 3

Abdorrahim Bahrami

Let instructions and string

Your task in this lab is to get yourself familiarized with Let instructions and Strings in Scheme and do some simple recursive programming. 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, we want to use letrec to define a function that returns the first element in a list that is unique. If the given list is empty, the function should return 0.

(define (first-unique L)

)

Here are some test cases to test your function.

(first-unique ‘(18 22 17 19 21 18 17)) ==> 22

(min-element ‘(7 2 2 1 8 1)) ==> 7

Task 2:

In this task, write a function that takes a string and returns the reverse of the given string. Try to do it without a helper function and just with local definitions using let. You can use built-in functions of Scheme. Even if you use a helper function, you need to use at least one form of Let(let, let*, or letrec).

(define (reverse-string str)

)

Here are some test cases to test your function.

(palindrome “rahim”) ==> “mihar”

(palindrome “extension") ==> “noisnetxe”

Task 3 (Challenge): Write a function that takes a string and a positive number and returns a list of all substrings of the given string, whose length is the given number. You need to create a loop using the named-let structure you learned in class.

(define (all-substrings str1 n)

)

Here are some test cases to test your function.

(all-substrings “Rahim” 2) ==> ‘(“Ra” “ah” “hi” “im”)

(all-substrings “Green” 5) ==> ‘(“Green”)