CSI 2120 – Lab Task 2

Abdorrahim Bahrami

Lists in Scheme

Your task in this lab is to get yourself familiarized with lists 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 do some more practice on recursive functions before trying some functions that operate on lists. Write a recursive function that takes three numbers a, b, and d and returns the number of all numbers in the range $[a, b]$ that are not divisible by d. Note that a and b are inclusive.

Note that *modulo* is a function for computing the remainder of a division in Scheme.

```
(define (numNotDiv a b d)


)
```

Here are some test cases to test your function.

(numNotDiv 6 12 4) ==> 5

(numNotDiv 1 10 3) ==> 7

Task 2:

In this task, write a recursive function that takes a list and an integer number n and returns a list of all integers in the range $[1, n]$ which are missing from the given list.

(define (missing L x)

)

Here are some test cases to test your function.

(count '(2 4 6 1) 5) ==> ‘(5 3)

(count '(1 2 3 1 5 6 5 4 2) 7) ==> ‘(7)

Task 3 (Challenge): Write a function that takes a list of bills or coins in a country sorted in decreasing order and an integer number as the amount of money and returns a list that shows how we can change that money to the bills. Assume that the numbers are in a way that it is always possible to do the change.

(define (coin-change L)

)

Note that *quotient* is a function for computing the quotient of a division in Scheme.

Here are some test cases to test your function.

(coin-change '(100 50 20 10 5) 345) ==> ‘(3 0 2 0 1)

Note that this example shows bills in Canada, to exchange 345 dollars, we need 3 hundred-dollar bills, 2 twenty-dollar bills, and 1 five-dollar bill.

(coin-change ‘(16 3 1) 26) ==> ‘(1 3 1)