Younwoo Park
2022310853
June 11, 2025

Programming Language SWE3006_41
Dept. of System Management Engineering
Sungkyunkwan University

Assignment 6B: Function Programming Syntax

Q1.

Q1. What does the following Scheme function do?

```
(define (y s lis)
        (cond
          ((null? lis) '() )
       ((equal? s (car lis)) lis)
          (else (y s (cdr lis)))
))
```

The scheme function first finds the first occurrence of `s` in `lis`, and once it does, it returns the sub-list (rest of the list) starting from that element. If an occurrence is not found, an empty list is returned, as per the third line.

Q2.

Q2. What does the following Scheme function do?

```
(define (x lis)
  (cond
    ((null? lis) 0)
    ((not (list? (car lis)))
      (cond
        ((eq? (car lis) #f) (x (cdr lis)))
        (else (+ 1 (x (cdr lis))))))
    (else (+ (x (car lis)) (x (cdr lis))))))
```

It counts the number of non-false elements, which includes but are not limited to #t, numbers, and symbols. It recursively traverses the list, including any nested lists, and counts all occurrences where the element is not #f.

Q3.

The function can be modified as follows:

```
(define (countX x aList)
  (cond
    ((null? aList) 0)
    ((equal? (car aList) x) (+ 1 (countX x (cdr aList))))
    (else (countX x (cdr aList)))))
```

By adding another parameter `x`, we can count the number of element x in a simple list.

Q3. Change the following function

The following function returns the number of zeros in a given simple list of numbers

```
(define (countZero aList)
 ;; assume that aList is a list of numbers
 (cond ( (null? aList) 0 )
       ( (eq? (car aList) 0) (+ 1 (countZero (cdr aList))))
       ( #t (countZero (cdr aList)))
       ) )
```

Change this function to make it return the number of element x in simple list.

For example: number of **1** in the list '(1 2 5 4 1 3 1) is 3

For example: number of **4** in the list '(4 2 4 4 4 3 4 6 7) is 5

Q4.

The scheme function that returns the min and max number of a list is as follows:

```
(define (findMinMax aList)
  (let loop ((aList (cdr aList))
             (minimum (car aList))
             (maximum (car aList)))
    (if (null? aList)
        (list maximum minimum)
        (loop (cdr aList)
              (min minimum (car aList))
              (max maximum (car aList))))))
```

Q4. Write the following function

- Write a Scheme function that takes a simple list of numbers as a parameter and returns a list with the largest and smallest numbers in the input list.
- For example: the list '(1 2 5 4 1 3 1) returns '(5 1)

Q5.

The scheme function that returns a list with the given element deleted is as follows:

```
(define (deleteElem e aList)
  (cond
    ((null? aList) '())
    ((eq? (car aList) e)
     (deleteElem e (cdr aList)))
    (else
     (cons (car aList)
           (deleteElem e (cdr aList))))))
```

Q5. Write the following function

- Write a Scheme function that takes a list and an atom as parameters and returns a list identical to its parameter list except with all instances of the given atom deleted.
- Assume the function name is deleteatom, then, you can use the function as follows:

```
> (deleteatom '2 '(2 3 1 4 5 6 -1 2 7))
'(3 1 4 5 6 -1 7)
> (deleteall 'a '(a r f t a r c d a e))
'(r f t r c d e)
```