CSCI 275

Lab 02: Lists and Recursion
Due Thursday February 20 at 11:59 PM

The goals for this assignment are increasing your skills with

- Writing Scheme functions
- Using recursion with flat lists
- Using lists to structure data

You can use the solution to any exercise as a helper function in subsequent exercises, and you can also write stand--alone helper functions..

Part 1 - Lists as structures

1. Write a function **merge** that merges two sorted lists of numbers onto one sorted list, the way MergeSort works.
    - (merge '(1 4 5) '(2 3 4 6)) returns (1 2 3 4 4 5 6)

2. Write a **sort** function for lists of numbers. Don't get the idea from (1) that you should do MergeSort. Try InsertionSort.
    - (sort '(5 1 8 3 7)) returns (1 3 5 7 8)

3. Write function **(contains-sublist? sublist biglist)** that determines if a list contains a particular sublist:.
    - (contains-sublist? '(2 3 4) '(1 2 3 4 5)) returns #t
    - (contains-sublist? '(2 3 4) '(1 2 5 3 4)) returns #f

4. Write function **(rember-sublist sublist biglist)** that removes the first occurrence of the sublist from the given list
    - (rember-sublist '(2 3 4) '(1 2 3 4 5)) returns (1 5)
    - (rember-sublist ‘(2 3 4) ‘(1 2 5 3 4)) returns (1 2 5 3 4)

Part 2 - Association Lists

Flat lists aren't a very good way to store data. An obvious improvement is to have the list consist of elements, each of which is itself a list, representing the data for one individual. For

example, we might make a phone-book as a list of name, phone number pairs. Such a structure is called an association list. For example,

(define phone-book
'((barbara 775-1234) (luke 774-2839) (nick 775-0912) (valerie 775-9043)))

5. Write function **(phone-number person phone-book)** that returns the phone number of the given person, or the atom **'disconnected** if there is no entry for the person.
   With the phone book defined above (phone-number 'nick phone-book) returns 775-0912.

6. Write funtion **(person phone-number phone-book)**that returns the name of the person with the given phone number.
   So (person '775-0912 phone-book) returns nick.

Part 3 – Other Structured Lists

7. Write function **(deepen lyst)** that wraps a pair of parentheses around each top-level element in lyst. For example:

   - (deepen ‘(a b c)) returns ‘((a) (b) (c))
   - (deepen ‘(a (b (c d)) e)) returns ‘((a) ((b (c d))) (e))

8. Write function **(evalBin binVec)** that takes a binary vector, a flat list of 1s and 0s, and finds the base-10 value of the number represented by this vector. For example,

   - (evalBin ‘(1 0 1 1)) returns 11
   - (evalBin ‘(1 1 0) returns 6

   Hint: if you read the digits from left to right, at each step you double the previous value and add the new digit. For example, with ‘(1 0 1 1) start with value 0. Double it and add 1 to get the value 1 for ‘(1). Double that and add 0 to get the value 2 for ‘(1 0). Double that and add 1 to get value 5 for ‘(1 0 1). Double that and add 1 to get 11 for ‘(1 0 1 1).

9. Write function **(sub old new lat)** that replaces each instance of atom old in lat with atom new. For example:(sub ‘a ‘x ‘(a b r a c a d a b r a) returns ‘(x b r x c x d x b r x)

10. Write function **(subs oldList newList lat)** that takes a list of atoms to swap out and a list of atom to replace them with and performs these changes on lat. You can assume that oldList and newList have the same length. For example:

    - (subs ‘(b) ‘(m) ‘(b o b)) returns ‘(m o m)
    - (subs ‘(b o) ‘(m u) (‘b o b)) returns ‘(m u m)
    - (subs ‘(a b c) ‘(x y z) ‘(a b r a c a d a b r a)) returns ‘(x y r x z x d x y r x)