CS 115 Fall 2012

Assignment 6

Due at 9:00 AM on Wednesday, November 14

Assignment Guidelines:

- The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.
- Do not copy the purpose directly from the assignment description. The purpose should be written in your own words and include reference to the parameter names of your functions.
- Do not send any code files by email to your instructors or tutors. It will not be accepted by course staff as an assignment submission.
- Read the course Web page for more information on assignment policies and how to organize and submit your work.
- Download the interface file from the course Web page.
- Follow the instructions in the style guide (see the updated version posted on the course web page). Specifically, your solutions should be placed in files a06qY.rkt, where Y is a value from 1 to 4.
- For full marks, it is not sufficient to have a correct program. Be sure to follow all the steps of the design recipe, including the definition of constants and helper functions where appropriate.
- Test data for all questions will always meet the stated assumptions for consumed values.
- Read each question carefully for restrictions.
- Use only material from Modules 1 through 7 in your solution. Where possible use list abbreviations.

Language level: Beginning Student with List Abbreviations.
Coverage: Modules 6 and 7.

Useful structure definition:

```
;; An association list (al) is either
;;   empty or
;;   (cons (list k v) alst), where
;;        k is a number (the key),
;;        v is a symbol (the value), and
;;        alst is an association list.
```

Question 1.

A necklace consists of a series of knots separated by a fixed number of beads in between. The knots are of different colours (`'red`, `'blue`, and `'green`) and the beads are of different shapes (`'round`, `'square`, `'oval`). A necklace can have different types of knots but the same type of beads. Each necklace should consist of at least one knot.

Examples of necklaces are `(list 'red)`, `(list 'red (list 'square 'square) 'blue)` and `(list 'green (list 'oval) 'green (list 'oval) 'green)`.

Write a Scheme function `create-necklace` that consumes a **non-empty** list of symbols (`'red`, `'blue`, and `'green`), `knots`, and a **non-empty** list of symbols (`'round`, `'square`, `'oval`), `beads`, and produces a list consisting of symbols and lists of symbols that represents a necklace; with any pair of consecutive symbols from `knots` separated by the list of symbols from `beads` except the case when there is only one knot, in this case the function produces a list consisting of only the one symbol in `knots`. For example:

```
(create-necklace (list 'red) (list 'square 'square 'square) ) =>
                                                    (list 'red)

(create-necklace (list 'red 'blue) (list 'oval 'oval) ) =>
                                (list 'red (list 'oval 'oval) 'blue)

(create-necklace (list 'red 'blue 'green) (list 'round)) =>
                  (list 'red (list 'round) 'blue (list 'round) 'green)
```

Question 2.

In a multiple-choice exam, a student is required to get at least a certain mark to pass. The correct answers for the exam are represented by the symbols `'A`, `'B`, `'C` and `'D`. A student's answers are also represented by the same set of symbols.

Write a Scheme function `student-pass?` that consumes two **non-empty** lists of letter symbols of the same length, `corr-ans` (list of correct answers) and `stud-ans` (list of student's answers) and a natural number, `pass-mark`, and produces `true` if there are at least `pass-mark` correct matches. A correct match means symbols in the ith index in both lists are the same; that is the student got the answer correct. Note that if `pass-mark` is 0, `student-pass?` will produce `true`; that is the student will always pass. You can assume that the pass mark for the exams will not be more than the number of questions given. Also assume that the lengths of both `corr-ans` and `stud-ans` are equal.

For example:

```
(student-pass? ( list 'A 'A 'C 'D 'B 'D )
               ( list 'A 'C 'C 'D 'B 'D ) 4 )   => true

(student-pass? ( list 'C 'B 'A 'B )
               ( list 'D 'B 'A 'B ) 3 )   => true

(student-pass? ( list 'D 'A 'D 'D 'C )
               ( list 'D 'B 'C 'B 'A ) 3 )   => false
```

Question 3.

One of the common operations of an **association list** is the `insert` operation, which inserts a new entry (key and value pair) into a specified position in an association list.

Write a Scheme function `insert`, that consumes an association list, `alst`, a new entry, `new` and a position, `index` (non-negative) and inserts `new` into `alst` at position `index`. The first element of the association list is at position 0. You can assume that the provided index will not be greater than the length of the association list.

For example:

```
(insert empty (list 3 'v) 0 ) => (list (list 3 'v))

(insert (list (list 2 'a)) (list 5 'c) 1 ) =>
                                    (list (list 2 'a)(list 5 'c)

(insert (list (list 2 'a)(list 3 'v)) (list 5 'c) 1 ) =>
                           (list (list 2 'a)(list 5 'c)(list 3 'v))
```

Question 4.

Write a Scheme function `sublist?` that consumes two lists of numbers, `lst1` and `lst2`, and produces `true` if `lst1` is a sublist of `lst2` (i.e. `lst2` contains `lst1`) and `false` otherwise. The list `lst1` is a sublist of list `lst2` if the elements of `lst1` appear consecutively in `lst2` with no other elements intervening. If `lst1` is empty `sublist?` should produce true.

For example:

```
(sublist? (list 1 2 3) (list 2 2 3 1 2 3 4)) => true

(sublist? (list 2 3 3) (list 1 2 3 4 5 3)) => false

(sublist? (list 1 2 3) (list 1 2 6 1 2 3)) => true
```