Tutorial Problem Set 1

CS 152 – Abstractions and Paradigms in Programming

Date: 1 Feb 2017 Submission Date: 10th Feb 2017

1. The function cprod takes a list of lists and computes its Cartesian product:

   ```
   (cprod ((1 2 3) (4) (5 6))) = ((1 4 5) (1 4 6) (2 4 5)
                                  (2 4 6) (3 4 5) (3 4 6))
   ```

   Define cprod.

2. Imagine the following game: You are given a path that consists of white and black squares. You start on the leftmost square (which we'll call square 1) and your goal is to move off the right end of the path in the least number of moves. However, the rules stipulate that:

   - If you are on a white square, you can move either 1 or 2 squares to the right.
   - If you are on a black square, you can move either 1 or 4 squares to the right.

   Write a function fewest_moves that takes a path representated as a list of 0's and 1's (1 is white and 0 is black) and computes the minimum number of moves. As an example:

   (fewest-moves '(1 0 1 1 1 0 1 1 0 1 0 1 1 0 0 1 1 1)) returns 6 and is obtained by stepping on the squares at positions 1, 2, 6, 10, 11 and 15.

3. Can you write a tail-recursive version of reverse?

4. Run-length encoding (RLE) is a very simple form of data compression in which runs of data (that is, sequences of the same data value) are stored as a single data value and count. As an example:

   ```
   (rle '(1 1 1 3 3 2 5 5 5 4 4)) = ((1 3) (3 2) (2 1) (5 3) (4 2))
   ```

   Define rle

5. Write a function called summands which takes as input a positive integer n, and produces a list containing all ways of writing n as a sum of positive integers. Example:

   ```
   (summands 1)  = ((1))
   (summands 2)  = ((1 1) (2))
   (summands 3)  = ((1 1 1) (1 2) (2 1) (3))
   (summands 4)  = ((1 1 1 1) (1 1 2) (1 2 1) (1 3) (2 1 1) (2 2)
                    (3 1) (4))
   (summands 5)  = ((1 1 1 1 1) (1 1 1 2) (1 1 2 1) (1 1 3) (1 2 1 1)
                    (1 2 2) (1 3 1) (1 4) (2 1 1 1) (2 1 2) (2 2 1)
                    (2 3) (3 1 1) (3 2) (4 1)  (5))
   ```

6. An n-bit Gray code is a sequence of n-bit strings constructed according to certain rules. For example,

```
(gc 1) =  ((0) (1))
(gc 2) =  ((0 0) (0 1) (1 1) (1 0))
(gc 3) =  ((0 0 0) (0 0 1) (0 1 1) (0 1 0)
            (1 1 0) (1 1 1) (1 0 1) (1 0 0))
```

7. The solution of a Sudoku puzzle is correct if (a) Each of its rows is filled with digits 1 to 9 with each digit occurring once (b) Each of its columns is filled with digits 1 to 9 with each digit occurring once, and (c) each of its 3×3 squares is filled with digits 1 to 9 with each digit occurring once.

   Assume that a solution of a Sudoku puzzle is represented as list of lists, where each of the inner lists is a row of the solution. Write a function `check-c` that returns `#t` if the solution is correct with respect to condition (c) and `#f` otherwise.

8. Write a function `(lcs l1 l2)` to find a longest common subsequence of two lists. For example the following call to `lcs`:

   ```
   (lcs ‘(1 2 3 2 4 1) ‘(2 4 3 1 2 1))
   ```
   returns `(2 3 2 1)`.

9. Write a function called shuffle, which will produce all interleavings of two lists xs1 and xs2. As an example:

   ```
   (shuffle ’(1 2) ’(3 5 4)) =
       ((1 2 3 5 4) (1 3 2 5 4) (1 3 5 2 4) (1 3 5 4 2) (3 1 2 5 4)
        (3 1 5 2 4) (3 1 5 4 2) (3 5 1 2 4) (3 5 1 4 2) (3 5 4 1 2))
   ```

   The elements of shuffled list need not be in the same order as in the example. Explain briefly how your function works.