CSE1729 - Introduction to Programming March 4, 2017

Laboratory Assignment 6

Objectives

- Work with pairs and lists

Activities

1. Define a SCHEME function to take a number, and return a pair with the number and its square.

Solution:

```
(define (square-pair i)
       (cons i (* i i)))
```

2. Define a SCHEME function `square-l` which will take any list of numbers as input, and returns a list with each of those numbers squared.

Solution:

```
(define (square-l list)
  (define (square-l-aux list squares-list)
    (if (null? list)
        squares-list
        (cons (* (car list) (car list))
              (square-l-aux (cdr list) squares-list))))
  (square-l-aux list '()))
```

3. Define a SCHEME function `range` that takes a pair (x, y) of natural numbers as input and returns a list of all integers between them, inclusive.
For example if $p = (0, 10)$ then

```
(range p) → '(0 1 2 3 4 5 6 7 8 9 10)
```

Solution:

```
(define (range p)
  (define (range-aux i list)
    (if (> (car p) i)
          list
          (range-aux (- i 1) (cons i list))))
   (range-aux (cdr p) null))
```

4. **Scalar-Vector Multiplication** Multiplying a vector by a scalar produces a vector where each component is the corresponding value in the original vector multiplied by the scalar quantity. For example:

$$a(x_1 \quad x_2 \quad x_3) = (ax_1 \quad ax_2 \quad ax_3)$$

Define a SCHEME function, named `sv-mult`, which takes a list and a value as parameters and performs scalar-vector multiplication on them.

Solution:

```
(define (sv-mult a x)
  (if (null? x)
      (list)
      (cons (* a (car x)) (sv-mult a (cdr x)))))
```

5. **Vector Addition** Adding two vectors produces a vector where each component is the sum of the corresponding values in the original vectors. For example:

$$(x_1 \quad x_2 \quad x_3) + (y_1 \quad y_2 \quad y_3) = (x_1 + y_1 \quad x_2 + y_2 \quad x_3 + y_3)$$

Define a SCHEME function, named `v-add`, which takes two lists and performs vector addition on them. *Note: **vector subtraction** is structured in the same way.*

Solution:

```
(define (v-add x y)
  (if (null? x)
      (list)
      (cons (+ (car x) (car y)) (v-add (cdr x) (cdr y)))))
```

6. The **dot product** of two lists of numbers $(x1\ x2\ x3)$ and $(y1\ y2\ y3)$ is

$$x1 * y1 + x2 * y2 + x3 * y3$$

Define a recursive SCHEME function `(dot x y)` that takes two lists of numbers as its inputs and returns the dot product of those two lists. Do not use the in-built `map` function. You can assume the two lists have the same length.

Solution:

```
(define (dot-prod x y)
  (if (null? x) 0
      (+ (* (car x) (car y)) (dot-prod (cdr x) (cdr y)))))
```

7. The **cross product** of two sets represented by lists of numbers $X = (x1\ x2\ x3)$ and $Y = (y1\ y2\ y3)$, denoted $X \times Y$, is the set (list) of all possible pairs of numbers where the first number in each pair is a member of the first set (list) and the second number in each pair is a member of the second set (list). In our example, $X \times Y$ is $((x1 . y1)(x1 . y2)(x1 . y3)(x2 . y1)(x2 . y2)(x2 . y3)(x3 . y1)(x3 . y2)(x3 . y3))$. Define a SCHEME function `(cross x y)` which takes two lists as parameters and returns the list of pairs representing the cross product of the items in the lists x and y.

Solution:

```
(define (cross x y)
  (define (pairup a l)
    (if (null? l)
        (list)
        (cons (cons a (car l)) (pairup a (cdr l)))))
  (if (null? x)
      (list)
      (append (pairup (car x) y) (cross (cdr x) y))))
```