CSE 1729 - Introduction to Principles of Programming | October 9, 2018

Laboratory Assignment 6

Objectives

- Work with lists and a couple of interesting small problems.

Activities

In this lab, you will

- write some functions that manipulate lists, and
- write a couple of "interesting" functions that do not use lists.

1. Write the following functions on lists.
   (a) Write a Scheme function `(count-positives lst)` that counts the number of positive numbers in a list of numbers. See examples below.
   (b) Write a Scheme function `(multiply-list lst)` that evaluates to the product of all the elements in a list of numbers.
   (c) Write a Scheme function `(consecutive-ints a b)` that evaluates to a list of numbers from a to b, where a and b are integers; if $a > b$ the result is the empty list `'()`.
   (d) In similar fashion to (c), write a Scheme function `(consecutive-squares a b)` that evaluates to the list of perfect squares from a^2 to b^2. a and b should be integers; if $a > b$ the result should be the empty list `'()`.

```
; some examples:
> (count-positives (list 1 -23 0 -11 3 1002))
3
> (count-positives '())
0
> (pro-list '(1 2 3 4 5))
120

> (consecutive-ints 3 9)
(3 4 5 6 7 8 9)

> (consecutive-squares 1 10)
(1 4 9 16 25 36 49 64 81 100)
> (consecutive-squares -4 6)
(16 9 4 1 0 1 4 16 25 36)
> (consecutive-squares 4 -6)
()
```

2. Write a Scheme function `(count-if f lst)` that returns a number equal to the number of elements of list `lst` for which `(f element)` is true. The elements of the list could be anything.

3. A couple of interesting functions.

   (a) We can define a number of "interesting" subsequences of the Natural numbers (that is, the positive integers $1, 2, 3, \ldots$), for example the prime numbers, the positive even numbers, the powers of 2, and so forth.

   Write a Scheme function `(nth-filtered f n)`, where `f` is a function of one variable and `n` is a natural number, which evaluates to the n^{th} natural number such that `f` applied to that number is `#t`. Here are two examples:

   ```
   > (nth-filtered even? 1)
   2
   > (nth-filtered prime? 10)
   29
   ```

   It may be helpful to define a helper function; if you do, it should be local to `nth-filtered`.

   (b) Consider the following strategy for computing an approximation to the minimum value that a smooth function f takes between the numbers a and b. If $|b - a| < \frac{1}{10000}$, the minimum value is close to $f(a)$ so this value is returned. Otherwise, let m be halfway between a and b and (recursively) compute the minimum that f takes between a and m and the minimum f takes between m and b; the smaller of these two values is returned.

   Produce a Scheme function `(min-value f a b)` that implements this idea. (You can assume that `f` is a function, `a` and `b` are numbers, and `a` is less than `b`. For full credit, calls to `(min-value f a b)` should generate no more than two recursive calls to `min-value` (one to compute the minimum value between `a` and the midpoint, and one to compute the minimum value between the midpoint and `b`.