CS 2SC3 and SE 2S03

McMaster University, Fall 2009

Assignment 2

Instructor: William M. Farmer

Revised: 6 October 2009

Files due: 16 October 2009

1 Overview

The purpose of this programming exercise is to learn the following:

1. How finite sequences can be represented by lists.
2. How finite sequences can be represented by functions over the integers.
3. How to write procedures using loops and recursion.
4. How to write procedures that take functions as input.

2 Background

A *vector* is a mathematical entity that has direction and magnitude. A vector can be identified with a point in Euclidean space. A point in 2-dimensional Euclidean space can be represented with *Cartesian coordinates* as a pair $V = (a, b)$ of real numbers where a is the x-coordinate and b is the y-coordinate of the point, respectively. (A point in 2-dimensional Euclidean space could also be represented in other ways such as with *polar coordinates*.)

Suppose $V = (a, b)$ and $V' = (a', b')$ are two vectors represented by points in 2-dimensional Euclidean space. V is the *zero vector* if $a = b = 0$. The *negation* of V is the vector $(-a, -b)$. The *magnitude* of V is $\sqrt{a^2 + b^2}$. The *sum* of V and V' is the vector $(a + a', b + b')$. The *distance* between V and V' is the magnitude of the sum of V and the negation of V'.

Suppose $S = a_0, a_1, \ldots, a_n$ is a finite sequence of values. S can be represented in OCaml as the list $[a_0; a_1; \ldots; a_n]$. S can also be represented in OCaml as a function

```
function (i : int) -> b_i
```

with the integer n such that $b_0 = a_0, b_1 = a_1, \ldots, b_n = a_n$.

3 Requirements

3.1 Program Requirements

Write an OCaml program that includes:

1. The type definition

   ```
   type vector = float * float ;;
   ```

2. A variable named `vec_zero` of type `vector` that is bound to the zero vector.

3. A function named `vec_neg` of type

   ```
   vector -> vector
   ```

   that maps a vector to its negation.

4. A function named `vec_mag` of type

   ```
   vector -> float
   ```

   that maps a vector to its magnitude.

5. A function named `vec_add` of type

   ```
   vector -> vector -> vector
   ```

   that maps two vectors to their sum.

6. A function named `romulus_iter` of type

   ```
   vector list -> vector -> vector
   ```

   such that

   `romulus_iter` x v

   is the first member of the list x of vectors that is closest to the vector v. If $x =$ `[]`, the value of the function application is the zero vector. `romulus_iter` must be implemented using a for loop.

7. A function named `romulus_rec` of type

   ```
   vector list -> vector -> vector
   ```

   such that

   `romulus_rec` x v

   is the first member of the list x of vectors that is closest to the vector v. If $x =$ `[]`, the value of the function application is the zero vector. `romulus_rec` must be implemented using recursion.

8. A function named `remus_iter` of type

   ```
   (int -> vector) -> int -> vector -> vector
   ```

   such that

   `remus_iter` $f\ n\ v$

   is the first member of the finite sequence $f(o), \ldots, f(n)$ of vectors that is closest to the vector v. If $n < 0$, the value of the function application is the zero vector. `remus_iter` must be implemented using a for loop.

9. A function named `remus_rec` of type

   ```
   (int -> vector) -> int -> vector -> vector
   ```

   such that

   `remus_rec` $f\ n\ v$

   is the first member of the finite sequence $f(0), \ldots, f(n)$ of vectors that is closest to the vector v. If $n < 0$, the value of the function application is the zero vector. `remus_rec` must be implemented using recursion.

10. Code that tests the implementation of the components described above on a representative set of inputs. When the program is executed, it prints out the test results. (Note that the program should not ask the user for input.)

3.2 Submission Requirements

Put your program in a file named `prog2.ml`, and put a copy of your log book in a file named `log2.txt`. (Make sure that the files are named exactly as specified. Case matters!) Put your name and MacID at the top of each of these files. Create a directory named `assign2`. Put the files `prog2.ml` and `log2.txt` into this directory. Using subversion, import this directory into your directory in the course subversion repository at

```
https://websvn.mcmaster.ca/se2s03
```

Your files must be submitted no later than **10:30 a.m. on Friday, October 16, 2009.**

4 Marking Scheme

This assignment is worth 100 points allocated as follows:

1. **Objective** (checked automatically by software)

(a)	Program file is present	**____/10 pts.**
(b)	Program compiles	**____/10 pts.**
(c)	Program runs	**____/10 pts.**
(d)	Program prints test results	**____/10 pts.**
(e)	Program passes objective tests	**____/20 pts.**

2. **Subjective** (assessed by TAs)

(a)	Program satisfies the requirements	**____/20 pts.**
(b)	Choice of test inputs	**____/10 pts.**
(c)	Quality of print out of test results	**____/10 pts.**
(d)	Style (comments only)	

3. **Penalties**

(a)	Missing or substandard log book	**____/-10 pts.**

Notes:

1. A program that is submitted late will receive 0 points.
2. Your program must compile and execute correctly on mills to receive full marks.
3. Your program must be your own work.