SE 2AA4 Winter 2007

Software Design Exercise 3

Instructor: William M. Farmer

Assigned:	9 February 2007
Files due:	2 March 2007
Lab report due:	9 March 2007
Revised:	8 February 2007

The purpose of this software design exercise is to write a C program, consisting of three modules, that creates, uses, and tests an abstract data type (ADT) of vectors stored as a sorted linked list of records.

Background

A *vector* is a mathematical entity that has direction and magnitude. A two-dimensional vector is represented by a pair $V = (a, b)$ of real numbers where a and b are the x- and y-coordinates of V. Given a real number r and two vectors $V_1 = (a_1, b_1)$ and $V_2 = (a_2, b_2)$, the *scalar multiple* of r and V_1 is the vector $rV_1 = (ra_1, rb_1)$ and the *sum* of V_1 and V_2 is the vector $V_1 + V_2 = (a_1 + a_2, b_1 + b_2)$.

Step 1

Write a first module that creates a basic ADT of vectors. It should consist of a C code file named `VectorAdt.c` and a C header file named `VectorAdt.h` that contains the following interface components:

- A type declaration that defines the type `Vector`.
- Two selectors

  ```
  float get_x(Vector v);
  ```

  and

  ```
  float get_y(Vector v);
  ```

  that return the x and y coordinates of `v`, respectively.
- Two constructors

  ```
  Vector i_vector();
  ```

  and

  ```
  Vector j_vector();
  ```

  that return the units vectors in the x and y directions, respectively.

- A constructor

  ```
  Vector mul(float r, Vector v);
  ```

  that returns the scalar multiple of `r` and `v`.

- A constructor

  ```
  Vector add(Vector u, Vector v);
  ```

  that returns the sum of `u` and `v`.

The implementation of your module is required to:

1. Represent a vector as a pointer to a record (structure).
2. Store the records as a linked list.
3. The records in the linked list should be sorted lexicographically. This means, if V_1 and V_2 represent vectors (a_1, b_1) and (a_2, b_2), respectively, then V_1 comes before V_2 in the linked list iff $a_1 < a_2$ or ($a_1 = a_2$ and $b_1 < b_2$).
4. For all a, b of type `float`, store in the linked list at most one record that represents the vector (a, b). This means that, if one of interface procedures is required to return a vector (a, b) that has not already been created, the procedure should create the vector and insert it into linked list at the correct point. Otherwise, the vector should be found in the linked list.
5. Nothing in your implementation should limit the number of records that can be stored in the linked list.

Step 2

Write a second module that creates a definitional extension of the first module. It should consist of a C code file named `VectorAdtPlus.c` and a C header file named `VectorAdtPlus.h` that includes the interface of the first module plus the following components:

- A selector

  ```
  float get_magnitude(Vector v);
  ```

  that returns the length of `v`.

- A selector

  ```
  float get_angle(Vector v);
  ```

  that returns the angle from the x-axis to `v` measured in radians.

- A constructor

  ```
  Vector zero_vector();
  ```

  that returns the zero vector.

- A predicate

  ```
  bool is_orthogonal(Vector u, Vector v);
  ```

  that returns true iff `u` is orthogonal to `v`.

Step 3

Write a third module specification that "black box" tests the services provided by the second module. (Note that the services of the second module includes the services of the first module.) It should consist of a C code file named `VectorTest.c` and a C header file named `VectorTest.h`. Record the results of using this module to test the second module.

Step 4

Submit the six files `VectorAdt.c`, `VectorAdt.h`, `VectorAdtPlus.c`, `VectorAdtPlus.h`, `VectorTest.c`, and `VectorTest.h`. E-mail the `VectorAdt.c` and `VectorAdt.h` files to your assigned partner (your partner will be posted on the course web site on Thursday, March 1, 2007). Your partner will likewise e-mail his or her `VectorAdt.c` and `VectorAdt.h` files to you. You must finish this step no later than 23:59 on Friday, March 2, 2007.

Step 5

After you have received your partner's `VectorAdt.c` and `VectorAdt.h` files, replace your `VectorAdt.c` and `VectorAdt.h` files with your partner's. Do not make any modifications to any of the code. Run your test module and record the results.

Step 6

Write a report that includes the following:

1. Your name and MAC ID.
2. Your `VectorAdt.c`, `VectorAdt.h`, `VectorAdtPlus.c`, `VectorAdtPlus.h`, `VectorTest.c`, and `VectorTest.h` files.
3. Your partner's `VectorAdt.c` and `VectorAdt.h` files.
4. The results of the test of your second module.

5. The results of the test of your second module using your partner's first module.

6. A discussion of the test results and what you learned doing the exercise. List any problems you found with (1) your program, (2) your partner's `VectorAdt.c` and `VectorAdt.h` files, and (3) the specification of the three module specifications. Discuss how these problems could have been avoided.

7. A copy of the part of your log book relevant to this lab exercise.

The lab report is due no later than the beginning of the tutorial session on Friday, March 9.

Notes:

1. Please put your name and MAC ID at the top of each of your source files.

2. Your program must work on birkhoff when compiled by `gcc`.

3. If your partner fails to provide you with his or her `VectorAdt.c` and `VectorAdt.h` files by the deadline, please tell the instructor via e-mail as soon as possible.