SE 2A04 Fall 2002

Lab Exercise 5

Instructor: William M. Farmer

Assigned:	08 November 2002
Demonstration due:	15 November 2002
Lab report due:	22 November 2002

The purpose of this lab exercise is to implement in C an abstract data type (ADT) of vectors as a sorted linked list of records.

Step 1

Write a module in C named `test-vectors` that "black box" tests any module named `vectors` implementing the module interface specification (MIS) for a Vectors ADT given below. (Note: A module in C consists of two files, a `.h` header file and a `.c` code file.)

Step 2

Write a module in C named `vectors` that implements the MIS for a Vectors ADT given below. In addition, your implementation is required to:

(1) Represent a vector by a pointer to a record.

(2) Store the records as a linked list.

(3) The records in the linked list should be sorted lexicographically. This means, if r_1 and r_2 represent vectors (x_1, y_1) and (x_2, y_2), respectively, then r_1 comes before r_2 in the linked list iff $x_1 < x_2$ or ($x_1 = x_2$ and $y_1 < y_2$).

(4) For all $x, y \in$ `float`, store in the linked list at most one record that represents the vector (x, y).

(5) Except for the size of the type `float`, nothing in your implementation should limit the number of records that can be stored in the linked list.

Step 3

Construct a C program named `test-mine` that uses your `test-vectors` module to test your `vectors` module.

Step 4

During the lab session on November 15, demonstrate the `test-mine` program.

Step 5

Before or during the lab session on November 1, send a copy of your `vectors` module to your receiver partner, and get a copy of your sender partner's `vectors` module. Construct a C program named `test-partners` using your `test-vectors` module to test your sender partner's `vectors` module.

Step 6

Write a lab report that includes the following:

(1) A copy of the Lab Exercise 5 Marking Scheme (which is available on the course Web site) stapled to the front of your report.

(2) A copy of your `test-vectors` module and a brief explanation of its design.

(3) A copy of your `vectors` module and a brief explanation of its design.

(4) The results of the test of your `vectors` module.

(5) The results of the test of your sender partner's `vectors` module.

(6) A discussion of the test results and what you learned doing the lab exercise.

(7) A discussion of any problems you found with the MIS.

(8) 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 November 22.

Axiomatic Input/Output MIS for Vectors ADT:

- Imported modules: none required
- Interface:

```
INTERFACE vectors;
  TYPE vector;
  CONST zero: vector;
  PROCEDURE getvec(x,y: float): vector;
  PROCEDURE xval(v: vector): float;
  PROCEDURE yval(v: vector): float;
  PROCEDURE add(u,v: vector): vector;
  PROCEDURE mul(r: float; v: vector): vector;
END vectors.
```

- Exceptions: none required
- Axioms:

(1) $\texttt{xval}(\texttt{zero}) = 0.$

(2) $\texttt{yval}(\texttt{zero}) = 0.$

(3) $\forall\, x, y : \texttt{float}\ .\ \texttt{xval}(\texttt{getvec}(x, y)) = x$

(4) $\forall\, x, y : \texttt{float}\ .\ \texttt{yval}(\texttt{getvec}(x, y)) = y.$

(5) $\forall\, u, v : \texttt{vector}\ .\ \texttt{xval}(\texttt{add}(u, v)) = \texttt{xval}(u) + \texttt{xval}(v).$

(6) $\forall\, u, v : \texttt{vector}\ .\ \texttt{yval}(\texttt{add}(u, v)) = \texttt{yval}(u) + \texttt{yval}(v).$

(7) $\forall\, r : \texttt{float}\ .\ \forall\, v : \texttt{vector}\ .\ \texttt{xval}(\texttt{mul}(r, v)) = r * \texttt{xval}(v).$

(8) $\forall\, r : \texttt{float}\ .\ \forall\, v : \texttt{vector}\ .\ \texttt{yval}(\texttt{mul}(r, v)) = r * \texttt{yval}(v).$