SE 2A04 Fall 2000

Lab Exercise 6

Instructor: William M. Farmer

Assigned: 21 November 2000
Lab report due: 4 December 2000

Unlike the previous five lab exercises you will work on this lab by yourself.

Consider the following specification of an interface for a list module that can be used to create and access lists of elements.

List Module Interface Specification

Let $\mathsf{Boolean}$ denote the set $\{\mathsf{true}, \mathsf{false}\}$ of truth values, $\mathsf{Natural}$ denote the set of the natural numbers, $\mathsf{Element}$ denote a set of abstract elements, and $\mathsf{List(Element)}$ denote the set of finite sequences of elements. Let

$$[e_1, e_2, \ldots, e_n]$$

denote the member of $\mathsf{List(Element)}$ consisting of the ordered elements

$$e_1, e_2, \ldots, e_n$$

where $n \geq 0$.

The interface is composed of the following functions:

1. $\mathsf{list\_member} : \mathsf{Element} \times \mathsf{List(Element)} \rightarrow \mathsf{Boolean}$ is a binary function such that, given $e \in \mathsf{Element}$ and $[e_1, \ldots, e_n] \in \mathsf{List(Element)}$ as input, returns true iff $e = e_1 \vee \cdots \vee e = e_n$.

2. $\mathsf{list\_nil} : \rightarrow \mathsf{List(Element)}$ is a nullary function that returns the empty list $[\,]$.

3. $\mathsf{list\_construct} : \mathsf{Element} \times \mathsf{List(Element)} : \rightarrow \mathsf{List(Element)}$ is a binary function such that, given $e \in \mathsf{Element}$ and $[e_1, \ldots, e_n] \in \mathsf{List(Element)}$ as input, returns $[e, e_1, \ldots, e_n]$.

4. $\mathsf{list\_head} : \mathsf{List(Element)} \rightarrow \mathsf{Element}$ is a unary function such that, given $[e_1, \ldots, e_n] \in \mathsf{List(Element)}$ as input, returns e_1 if $n \neq 0$ and is undefined if $n = 0$.

5. $\mathsf{list\_tail} : \mathsf{List(Element)} \to \mathsf{List(Element)}$ is a unary function such that, given $[e_1, \ldots, e_n] \in \mathsf{List(Element)}$ as input, returns $[e_2, \ldots, e_n]$ if $n \neq 0$ and is undefined if $n = 0$.

6. $\mathsf{list\_take} : \mathsf{Natural} \times \mathsf{List(Element)} \to \mathsf{List(Element)}$ is a binary function such that, given $m \in \mathsf{Natural}$ and $[e_1, \ldots, e_n] \in \mathsf{List(Element)}$ as input, returns $[e_1, \ldots, e_m]$ if $m < n$ and $[e_1, \ldots, e_n]$ if $m \geq n$.

7. $\mathsf{list\_drop} : \mathsf{Natural} \times \mathsf{List(Element)} \to \mathsf{List(Element)}$ is a binary function such that, given $m \in \mathsf{Natural}$ and $[e_1, \ldots, e_n] \in \mathsf{List(Element)}$ as input, returns $[e_{m+1}, \ldots, e_n]$ if $m < n$ and $[\,]$ if $m \geq n$.

8. $\mathsf{list\_append} : \mathsf{List(Element)} \times \mathsf{List(Element)} \to \mathsf{List(Element)}$ is a binary function such that, given $[e_1, \ldots, e_n], [e'_1, \ldots, e'_{n'}] \in \mathsf{List(Element)}$ as input, returns $[e_1, \ldots, e_n, e'_1, \ldots, e'_{n'}]$.

9. $\mathsf{list\_length} : \mathsf{List(Element)} \to \mathsf{Natural}$ is a unary function such that, given $[e_1, \ldots, e_n] \in \mathsf{List(Element)}$ as input, returns n.

Step 1

Specify List(Element) as an abstract data type.

Step 2

Implement the list module in C using the stack module from Lab Exercise 5 as follows:

1. Boolean is represented by the set $\{0, 1\}$ of values of type `int` in C.

2. Element is defined to be Natural.

3. Natural is represented by the set of natural numbers of type `int` in C.

4. List(Element) is represented by `int`. A member of `int` is a number of a bin that either is empty or "holds" a list represented as a stack. The empty list is represented as the bottom stack.

5. Up to 100 different lists can be stored at a time.

No user interface program is required.

Step 3

Test the implementation of the list module.

Step 4

Write a lab report that includes the following:

1. A description and analysis of your work and the results obtained. In particular, analyze the difficulties you encountered and explain how they could have been avoided.

2. Listings of your source code and documentation. Be prepared to provide copies of your source code files if they are requested. Your full program must work correctly on the Sun workstations in the laboratory.

3. A description of how you tested your programs including what test data you used.

4. A copy of your log book.

The final lab report is due at the beginning of class on 4 December 2000.