SE 2A04 Fall 2000

Lab Exercise 5

Instructor: William M. Farmer

Assigned: 7 November 2000
Lab report due: 21 November 2000

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

Stack 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, and $\mathsf{Element}$ denote a set of abstract elements. The set $\mathsf{Stack(Element)}$ of *stacks of elements* is defined inductively by the following statements:

1. $\mathsf{bottom} \in \mathsf{Stack(Element)}$.
2. If $e \in \mathsf{Element}$ and $s \in \mathsf{Stack(Element)}$, then the ordered pair $(e, s) \in \mathsf{Stack(Element)}$.

The interface is composed of the following functions:

1. $\mathsf{stack\_get\_bottom} : \to \mathsf{Stack(Element)}$ is a nullary function that returns bottom.
2. $\mathsf{stack\_empty\_q} : \mathsf{Stack(Element)} \to \mathsf{Boolean}$ is a unary function such that, given $s \in \mathsf{Stack(Element)}$ as input, returns true iff $s = \mathsf{bottom}$.
3. $\mathsf{stack\_push} : \mathsf{Element} \times \mathsf{Stack(Element)} \to \mathsf{Stack(Element)}$ is a binary function such that, given $e \in \mathsf{Element}$ and $s \in \mathsf{Stack(Element)}$ as input, returns (e, s).
4. $\mathsf{stack\_pop} : \mathsf{Stack(Element)} \to \mathsf{Stack(Element)}$ is a unary function such that, given $s \in \mathsf{Stack(Element)}$ as input, returns s' if $s = (e, s')$ for some $e \in \mathsf{Element}$ and $s' \in \mathsf{Stack(Element)}$ and is undefined if $s = \mathsf{bottom}$.
5. $\mathsf{stack\_top} : \mathsf{Stack(Element)} \to \mathsf{Element}$ is a unary function such that, given $s \in \mathsf{Stack(Element)}$ as input, returns e if $s = (e, s')$ for some $e \in \mathsf{Element}$ and $s' \in \mathsf{Stack(Element)}$ and is undefined if $s = \mathsf{bottom}$.

6. $\mathsf{stack\_height} : \mathsf{Stack(Element)} \to \mathsf{Natural}$ is a unary function such that, given $s \in \mathsf{Stack(Element)}$ as input, returns $1 + \mathsf{height}(\mathsf{pop}(s))$ if if $s = (e, s')$ for some $e \in \mathsf{Element}$ and $s' \in \mathsf{Stack(Element)}$ and returns 0 if $s = \mathsf{bottom}$.

Step 1

Implement the stack module in C using the tree module from Lab Exercise 4 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. bottom is represented by the integer tree -1, and the other members of Stack(Element) are represented by trees (as implemented in the tree module) created from -1 and the natural numbers.
5. Up to 100 different stacks can be stored at a time.

Step 2

As a separate program, write a user interface program that allows users to store up to 100 different stacks at a time and to inquire about the characteristics of the stored stacks. The user interface program must accomplish its task by using the stack module described above. That is, the user interface should have access to the interface functions of the stack module but not to the interface functions of the tree module. The user interface program should inform the user in a friendly way of any errors in input.

Step 3

Test the implementation of the stack module and the user interface program separately until you are satisfied that they are correct. Then combine them and your tree module to produce a full program that can be demonstrated. Test the full program.

Step 4

During the lab session on 14 November 2000, exchange your tree module implementation (from Lab Exercise 4) with that of another team chosen by the instructor or TAs. Then combine your stack module and user interface program with it to produce a full program. Test the full program in the same way as in Step 3.

Step 5

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 each team member's log book.

The final lab report is due at the beginning of the lab session on 21 November 2000.

Step 6

During the lab session on 21 November 2000, each team will be required to demonstrate its full program to one of the TAs.