SE 2A04 Fall 2000

Lab Exercise 4

Instructor: William M. Farmer

Assigned: 24 October 2000
Lab report due: 7 November 2000

Step 1

Below is an interface for a module of tree functions. The module has 100 storage "bins" (numbered $1, 2, \ldots, 100$) so that it can store up to 100 different trees at a time. The definitions from Lab Exercise 2 are assumed. With your partner, produce one implementation of the module. No trees should be stored initially.

Interface:

1. $\mathsf{check\_bin} : \mathbf{Z} \to \mathbf{Z}$ is a unary function that, given x as input, returns 1 if $1 \leq x \leq 100$ and a pair tree is store in the xth bin; 0 if $1 \leq x \leq 100$ and an integer tree is store in the xth bin; -1 if $1 \leq x \leq 100$ and no tree is stored is in the xth bin; and -2 if not $1 \leq x \leq 100$.

2. $\mathsf{display\_tree} : \mathbf{Z} \to \mathsf{strings}$ is a unary function that, given x as input, returns a string representation of the tree stored in the xth bin.

3. $\mathsf{store\_integer} : \mathbf{Z} \to \mathbf{Z}$ is a unary function that, given an integer x as input, stores x in some unused bin with number y and then returns y.

4. $\mathsf{store\_pair} : \mathbf{Z} \times \mathbf{Z} \to \mathbf{Z}$ is a binary function that, given x and y as input, stores in some unused bin with number z the pair of the trees stores in the xth and yth bins and then returns z.

5. $\mathsf{left} : \mathbf{Z} \to \mathbf{Z}$ is a unary function that, given x as input, returns the number of a bin holding the left component of the tree stored in the xth bin.

6. $\mathsf{right} : \mathbf{Z} \to \mathbf{Z}$ is a unary function that, given x as input, returns the number of a bin holding the right component of the tree stored in the xth bin.

7. $\mathsf{height} : \mathbf{Z} \to \mathbf{Z}$ is a unary function that, given x as input, returns the height of the tree stored in the xth bin.

8. $\mathsf{size} : \mathbf{Z} \to \mathbf{Z}$ is a unary function that, given x as input, returns the number of pairs in the tree stored in the xth bin that are distinct with respect to equality between trees.

Step 2

As a separate program, write a user interface program that allows users to store up to 100 different trees at a time and to inquire about the characteristics of the stored trees. The user interface program must accomplish its task by using the tree module described above. It should inform the user in a friendly way of any errors in input.

Step 3

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

Step 4

During the lab session on 31 October 2000, exchange your tree module implementation with that of another team chosen by the instructor or TAs. Test it separately in the same way as in Step 3. Then combine your 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 7 November 2000.

Step 6

During the lab session on 7 November 2000, each team will be required to demonstrate its full program to one of the TAs.