CSE 332: Data Abstractions
Assignment #2
October 6, 2010
updated October 13, 6:45 a.m.
due: Friday, October 15, 2:15 p.m.

The purpose of this programming assignment is to give you some experience in how to implement stacks, how to implement binary trees, and how stacks are used by compilers to implement recursion. In summary, you will be writing a sorting algorithm that works as follows: you will insert all the integer keys to be sorted into an initially empty binary search tree T, and then traverse T to retrieve the keys in sorted order. Instead of implementing the traversal recursively, you will use a stack.

1. Implement a stack class. The methods in the stack class should include push, pop, top, isEmpty, and a constructor to make a new empty stack. You may choose whether to use the linked list or array implementation of your stack. Do not use Java's Stack or LinkedList library. If you choose the array implementation, be sure that you allocate space for at least 1000 stack entries. Your stack class does not have to be designed to store generics: it is fine to tailor it to the type you will need in step 4 below.

2. Implement a binary tree class. The methods in the binary tree class will include ones that (a) return the key at the root, (b) return the left subtree of the root, (c) return the right subtree of the root, (d) test if tree is empty, (e) construct an empty tree, and (f) any other method that you find absolutely necessary. Each node in your implementation should have fields for a key and left and right subtrees, but should not contain a parent reference.

3. Implement the algorithm of Figure 4.22 using your binary tree class. Because you will want to allow duplicate keys in your tree (there may be duplicates in the list of keys to be sorted), you must do something different than lines 19 and 20 of Figure 4.22. You do not have to implement other binary search tree methods that are not needed for this assignment.

4. What type of tree traversal do you need to implement to retrieve the keys from your binary search tree in sorted order? Implement it without recursion, instead using your stack class to simulate the recursion. Here are questions to think about that will help you work out the use of the stack: What type of object needs to be stored on the stack? Exactly which object needs to be pushed when you are about to start the traversal of the left subtree? When do you pop the stack and what do you do with the result that is popped? What do you do when the stack is empty?

5. Your program should be called `SortInts` and will take two filenames as arguments. The first one is the input file. If it does not already exist, your program should print a warning and exit. The second one is the output file. It should be created if it does not exist and overwritten if it does. If there are not exactly two arguments, print a warning and exit. Otherwise read the integers to be sorted from the input file and write the sorted integers into the output file. The input and output files should each contain one integer per line, with nothing else in the file. We should be able to invoke your program from the command line via the command

   ```
   java SortInts infile.txt outfile.txt
   ```

Turn in all your source files at

https://catalyst.uw.edu/collectit/dropbox/trobison/11681