CSE 326: Data Structures
Assignment #2
October 9, 2009
revised: October 13
due: Monday, October 19, 11:15 a.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 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 will be the usual ones from page 74. You may choose whether to use the contiguous memory or linked memory implementation of your stack.

2. Implement a binary tree class. The methods in the binary tree class will include ones that (a) return the key at the given node, (b) return the left subtree of the given node, (c) return the right subtree of the given node, (d) test if the given 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 pointer.

3. Implement Algorithm 6.8 using your binary tree class. You may leave out the parts dealing with the "info" field; for our purposes the only information is the key. Because you will want to allow duplicate keys in your tree, you should also omit the lines

```
if Key(P) = K then
    Info(P) <- I
    return
```

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 using your stack class to avoid 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.

Turn in all your source files at

`https://catalysttools.washington.edu/collectit/dropbox/dcj3/7527`