CS 284D: Homework Assignment Bonus
Due: May 13rd, 11:55pm

1 Assignment Policies

Collaboration and IP Policy. Do homework individually. Do ask questions if anything is not clear. You can collaborate to understand the material. Do not solve the problem with others. Do not use any solution on the Internet. Do not exchange code with other students. **Any solution found online or in another submission will be penalized.**

Late Policy. No late submissions will be allowed without consent from the instructor. For urgent or unusual circumstances e-mail the instructor.

2 Assignment

Data Structure Programing Assignment 2: Binary Search Tree In this programing assignment you are to construct and perform transversals on a binary search tree. As in the first assignment your program will read from standard-in (the input will be only doubles) and outpurt to standard-out.

A binary search tree has the property that for each node (not a leaf) the doubled value stored at the node is greater then or equal to the value of its left child and less then the value of the right child. See Data Structures and Algorithms in Java, page 382. For a given input sequence the tree is unique. Below are some example:

Input: 1, 2, 3

Binary Search Tree:

```
1
 \
  2
   \
    3
```

Input: 2, 1, 3

Binary Search Tree:

```
    2
  /   \
1       3
```

Input 3, 4, 1, 2
Binary Search Tree:

```
   3
  / \
1     4
 \
   2
```

You probably have guessed correctly that the first number in the sequence is always the root of the tree, and likewise the last number is always a leaf. The following is the complete algorithm (modified from TreeSearch Algorithm, page 382):

```
Algorithm: insertValue(double k, BinaryTree T)
        // inserts k into Binary Tree
        node parent = findParent(k, T.root())
        if k  < =  parent.value then {
                     if  prarent left child = null
                          then make parent left child with value k
                     else return error
                                  // or you can findParent(k, parent left child)
                                  // but then you would use a while-loop
          }
        else {
                     if  prarent right child = null
                          then make parent right child with value k
                     else return error
        }
    // end algorithm
```

The algorithm above uses the recursive algorithm below to find the parent of the new node.

```
    Recursvie Algorithm: findParent (double k, node v)

        // base case
        if ( k  < =  v.value and  T.leftChild(v) = null ) then  return v
        elseif ( k >  v.value and T.rightChild(v) = null ) then  return v
        else { // recursive cases
             if k  < =  v.value then  findParent (k, T.leftChild(v))
            else findParent (k, T.rightChild(v))
        }

    // end recursive algorithm
```

Note that in these algorithms the binary tree does not use empty externals rather externals contian values.

The program will:

Print out a tree representation on it side and then print all the values of the nodes ascending order You are to write a print routine that prints the expression tree on its left side. For each node, you should print the double value. The text for a node should be indented 4 times the depth of the node (e.g., a node at level 0, the root, would not be indented, a node at depth 3 would be indented 12 spaces).

As an example for the input: 5, 2, 6, 1, 3, 9

The output of you program is:

```
        9
    6
5
        3
    2
        1
```

Inorder: 1, 2, 3, 5, 6, 9

Note that the tree is on it side and the inorder sequence is outputted after the tree representation. I recommend using java's StreamTokenizer class, the input will contian only one sequence of numbers and you are to print out only one tree.

The binary search tree can be implemented as a link tree or an array tree, but you can not assume a maximum size or use the java.util Vector class. I recommend using the interfaces and classes provided in the book. The code is online at http://datastructures.net/

After you have debugged your program put all the classes in a single file called BinarySearchTree.java. BinarySearchTree should be the name of the only public class in the file. Have fun growing trees and remember to submit a compiling program at least.

Acknowledgement We are thankful for MTU CS2321 for the exam question.