COIS-2020: Data Structures and Algorithms

Lab 7: Binary Search Tree

In this lab, you will be given a start-up code for a BST class. Your task is to complete some of the methods. Particularly, you will implement traversal methods (pre-order, post-order, in-order, and breadth-first traversals, and FindSmallest methods.

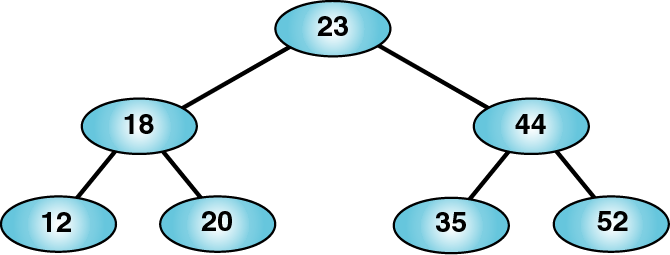
There are two source files attached to this lab: BST.cs (the BST class), and “program.cs” (the main driver). Check the files and complete the required methods: PreOrder, PostOrder, InOrder, BreadthFirst, and FindSmallest.

One Traverse method is given to you. The first 3 methods are basically the same three lines of code, just in different orders.

|  |
| --- |
| Here you are to show your three traversal methods: PreOrder, PostOrder, and InOrder  public string inOrder(Node root)  {  if (root == null)  return "";  string s = "";  /\* first recursively move on to on **left** child \*/  inOrder(root.left);  /\* then get the **data** of node \*/  s += root.value + " ";  /\* finally recursively move on to **right** child \*/  inOrder(root.right);  return s;//return string  }//end inOrder  public string preOrder(Node root)  {  if (root == null)  return "";  string s = "";  /\* first get the **data** of node \*/  s += root.value + " ";  /\* then recursively move on to **left** child \*/  preOrder(root.left);  /\* finally recursively move on to **right** child \*/  preOrder(root.right);  return s; //return string  }//end preOrder  public string postOrder(Node root)  {  if (root == null)  return "";  string s = "";  /\* first recursively move on to **left** child \*/  postOrder(root.left);  /\* then recursively move on to **right** child \*/  postOrder(root.right);  /\* finally get the **data** of node \*/  s += root.value + " ";  return s; //return string  }//end postOrder |

For the Breadth First algorithm, the logic is given in the source file. Before implementation, trace how it would process each iteration of the while loop for the tree to get an idea. For each step, show the printed value and the state of the queue at the bottom of the loop.

For example: using the following tree



**loop iteration 1**  
printed: 23  
queue:18, 44

**loop iteration 2**

…

|  |
| --- |
| Here you are to show your three traversal methods: breadthFirst(Node root)  public string breadthFirst(Node root)  {  //create a queue  //add the root to the queue  //immedietly  string n = ""; //Create a string  Queue<Node> Q = new Queue<Node>(); //create a queue  Q.Enqueue(root); //append the root to the queue  Node node;  while (Q.Count != 0)//while loop  {  node = Q.Dequeue(); //remove the head of the queue  n += Convert.ToString(node.value);//add value of the head node to the string  //add its children(if any) to the queue  if (root.left != null)  Q.Enqueue(root.left); //add left child if it exists  if (root.right != null)  Q.Enqueue(root.right); //add right child if it exists  }  return n;  }//end breadthFirst |

Implement a simple “public node FindSmallest”, recursively, that well, finds the smallest element in the tree (the very leftmost element in the tree). You should print the value of this node in the main.

|  |
| --- |
| Here you are to show your FindSmallest() method.  //findSmallest  //Make this work.  public int findSmallest(Node root)  {  Node current = root;  //getting the left most node in the bst (which is the smallest value)  while(current.left != null)  {  current = current.left;  }  return current.value;  } |

Text

Description automatically generated

Save this file as “YOUR\_NAME.doc” and upload it along with your BST source file onto the Lab submission link