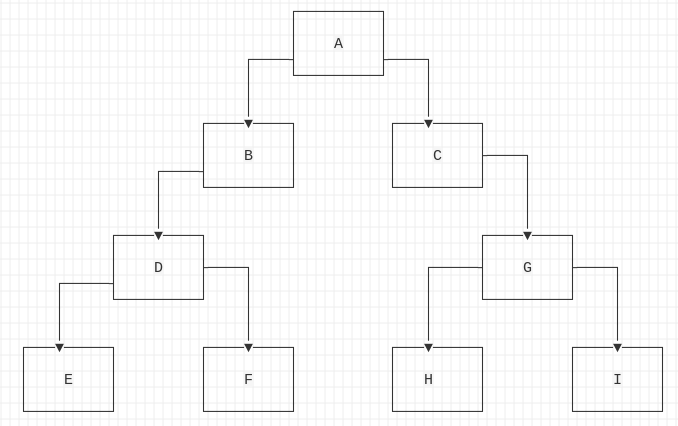
## Directions

You will have until the end of day (11:59pm) to complete this portion of the quiz. You may use notes, textbooks, labs, the [Java API](https://docs.oracle.com/javase/8/docs/api/), and class activities as resources for this. **You may not use other people's code (students in this class or others.)** If you do, your quiz may be invalidated and you may be given a 0 for the quiz, as well as other possible results listed in the [CS Department Cheating Guidelines](http://www.cs.unh.edu/~cs416/docs/cheating.pdf).

## Part 1 - Tree Building

In the TreeNode class, add a static method buildTree that creates and uses TreeNode objects to create a tree with the values shown below. (A is the root node, B is the root's left child, C is the root's right child, D is the left child of B with children E and F, and G is the right child of C with children H and I.) You do not need to write a method that could make different kinds of trees. Just manually create the nodes and connect them to make this tree. The method should return the TreeNode that is the root of the tree.



### 

### Part 1 - Starter Code

| public class TreeNode{  String data;  TreeNode left;  TreeNode right;  public TreeNode( String data ){  this.data = data;  }  public static TreeNode buildTree() {  // YOUR CODE HERE  }  } |
| --- |

## Part 2 - BST Minimum

In the BST class, add a non-static method min which takes no parameters and returns the String that is the minimum data stored in the tree. You can add helper methods if you would like, but you are not required to do so.

### 

### 

### 

### 

### 

### 

### Continued on next page

### Part 2 - Starter Code

| public class BST{  Node root;  public BST(){ }  public void add( String data ){  Node n = new Node( data );  if( root == null ){  root = n;  return;  }  Node curr = root;  boolean added = false;  while( !added ){  int c = curr.data.compareTo( data );  if( c > 0 ){  if( curr.left == null ){  curr.left = n;  added = true;  n.parent = curr;  }  else{  curr = curr.left;  }  }  else{  if( curr.right == null ){  curr.right = n;  added = true;  n.parent = curr;  }  else{  curr = curr.right;  }  }  }  }  public class Node{  String data;  Node left;  Node right;  Node parent;  public Node( String data ){  this.data = data;  }  }  // YOUR CODE HERE  } |
| --- |