King Saud University
College of Computer and Information Sciences
Computer Science Department

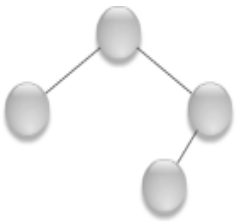
CSC 212 **First Semester 1439-1440**

Tutorial #8

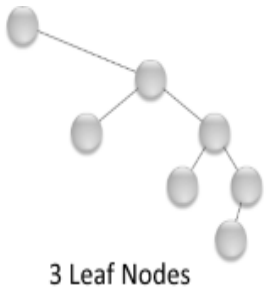
Problem 1

Write the method ***countLeafNodes*** part of the Binary Tree ADT. The method should return the number of leaf nodes in the tree.

Method: public int countLeafNodes()
Example:



2 Left Nodes



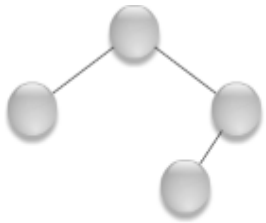
3 Leaf Nodes

```
public int countLeafNodes() {
     return countLeafNodesRec(root);
}

private int countLeafNodesRec(BTNode<T> pointer) {
     if (pointer == null)
          return 0;
     if (pointer.left == null && pointer.right == null)
          return 1;
     return countLeafNodesRec(pointer.left) +
     countLeafNodesRec(pointer.right);
}
```

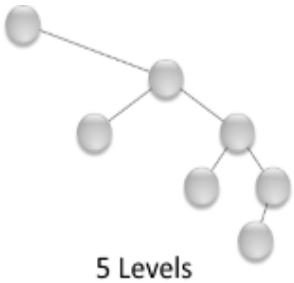
Problem 2

Write the method ***getHeight*** part of the Binary Tree ADT. It should return the height of the tree. The height of the tree is the longest path from the root to a leaf node.

Method: public int getHeight()
Example:



3 Levels



5 Levels

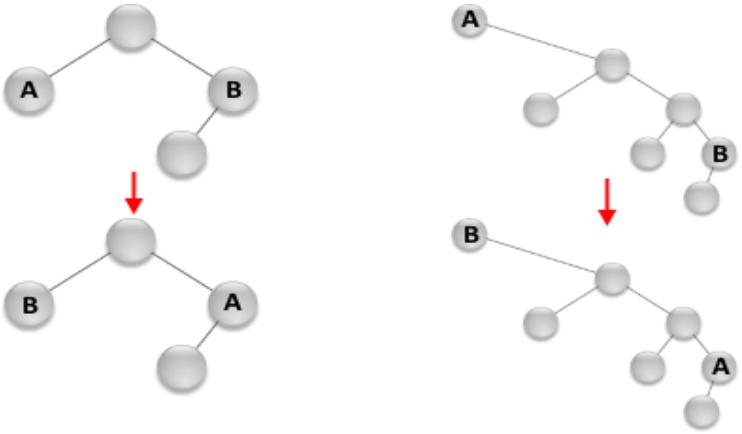
```
public int getHeight() {
     return getHeightRec(root);
}

private int getHeightRec(BTNode<T> pointer) {
     if (pointer == null)
          return 0;
     int leftHight = getHeightRec(pointer.left);
     int rightHight = getHeightRec(pointer.right);
     return leftHight > rightHight ? leftHight + 1 : rightHight + 1;
}
```

Problem 3

Write the static method ***swapMost*** (user of Binary Tree ADT) that takes a Binary Tree ***bTree*** and swaps the data of the left most node with the right most node.

Method: public static <T> void swapMost(BinaryTree<T> bTree)

Example:




```
public static <T> void swapMost(BinaryTree<T> bTree) {
     if (bTree.empty())
          return;
     bTree.find(Relative.Root);
     while (bTree.find(Relative.LeftChild));
     T mostLeft = bTree.retrieve();
     bTree.find(Relative.Root);
     while (bTree.find(Relative.RightChild));
     T mostRight = bTree.retrieve();
     bTree.update(mostLeft);
     bTree.find(Relative.Root);
     while (bTree.find(Relative.LeftChild));
     bTree.update(mostRight);
}
```