//Liang Fang

//Emplid: 23134567

//MAC286

//Take home exam

Question 3:

//Use the program from question 2 and display all the leaf nodes and their parents and lowest common ancestor.

**public** **class** TreeNode {

**int** value;

TreeNode right;

TreeNode left;

TreeNode parent;

**public** TreeNode(**int** value){

**this**.value = value;

right = **null**;

left = **null**;

parent = **null**;

}

**public** **boolean** isLeaf() {

**return** right == **null** && left == **null**;

}

}

**import** java.util.\*;

**public** **class** BinarySearchTree {

TreeNode root;

**public** **void** insert(**int** value) {

TreeNode newNode = **new** TreeNode(value);

**if**(root == **null**) {

root = newNode;

}

**else** {

TreeNode curr = root;

TreeNode parentNode = **null**;

**while**(curr!= **null**) {

parentNode = curr;

**if**(curr.value > value) {

curr = curr.left;

**if**(curr == **null**) {

parentNode.left = newNode;

newNode.parent = parentNode;

}

}

**else** {

curr = curr.right;

**if**(curr == **null**) {

parentNode.right = newNode;

newNode.parent = parentNode;

}

}

}

}

}

**public** **void** printTree(**int** height) {

**if**(root == **null**)

**return**;

Queue<TreeNode> q = **new** LinkedList<TreeNode>();

TreeNode curr = root;

q.add(curr);

**int** nodeCount = 0;

**int** level = 0;

**while**(q.size()>0 && level <=height) {

TreeNode node = q.remove();

nodeCount++;

printSpace(height\*(height -level));

**if**(node != **null**) {

System.***out***.print(node.value);

printSpace(height);

q.add(node.left);

q.add(node.right);

}

**else** {

System.***out***.print("\*");

}

**if**(nodeCount %2 ==0)

printSpace(height);

**if**(nodeCount == (**int**)Math.*pow*(2, level)) {

nodeCount = 0;

System.***out***.println("");

level++;

}

}

}

**private** **void** printSpace(**int** number) {

**while** (number > 0) {

System.***out***.print(" ");

number --;

}

}

**public** **void** preOrder(TreeNode curr){

**if**(curr == **null**) **return**;

**if**(curr.isLeaf()) {

System.***out***.println("Leaf Node:" + curr.value + ", Parent Node: "+ curr.parent.value+", Lowest Common Ancestor: " + curr.parent.value);

//lowest common ancestor of leaf node and parent node is always the parent node.

}

preOrder(curr.left);

preOrder(curr.right);

}

**public** **void** printLeafNode() {

preOrder(root);

}

}

**import** java.util.\*;

**public** **class** BinarySearchTreeApp {

**public** **static** **void** main(String[] args) {

**int** height;

String nodeString;

**int** intValue;

System.***out***.println("Enter the height of the tree: ");

Scanner scan = **new** Scanner(System.***in***);

height = scan.nextInt();

BinarySearchTree bst = **new** BinarySearchTree();

System.***out***.println("Enter those node values: ");

scan = **new** Scanner(System.***in***);

nodeString = scan.nextLine();

String[] nodeArr = nodeString.split(",");

**for**(**int** i = 0; i < nodeArr.length; i++) {

bst.insert(Integer.*parseInt*(nodeArr[i]));

}

System.***out***.println("The tree is: ");

bst.printTree(height);

bst.printLeafNode();

}

}



