Practice 01:

Implementation of binary tree and performing operations on it such as insertion, finding an element.

Code:

**package** leet;

**class** Node {

**int** data;

Node left;

Node right;

Node(**int** data) {

**this**.data = data;

right = **null**;

left = **null**;

}

};

**package** leet;

**class** BinaryTreeEg {

Node root;

BinaryTreeEg(**int** data){

root = **new** Node(data);

}

BinaryTreeEg(){

root = **null**;

}

// function for printing the tree

**void** printpostorder(Node node){

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

**return**;

}

System.***out***.print(node.data + " ");

printpostorder(node.left);

printpostorder(node.right);

}

// function for inserting the element in the tree

**private** Node addRecursive(Node node,**int** data){

**if**(node == **null**)

**return** **new** Node(data);

**if** (node.data > data) {

node.left = addRecursive(node.left, data);

}**else** **if** (node.data < data) {

node.right = addRecursive(node.right, data);

}**else**{

**return** node;

}

**return** node;

}

**void** add(**int** data){

root = addRecursive(root, data);

}

**void** printpostorder(){

printpostorder(root);

}

////////////////////////////////////////////////////////

// function for finding the element whether it exist or not

**private** **boolean** containsNodeRecursive(Node current, **int** data) {

**if** (current == **null**) {

System.***out***.println("DATA NOT FOUND");

**return** **false**;

}

**if** (data == current.data) {

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

**return** **true**;

}

**return** data < current.data ? containsNodeRecursive(current.left, data) : containsNodeRecursive(current.right, data);

}

**public** **boolean** containsNode(**int** value) {

**return** containsNodeRecursive(root, value);

}

/////////////////////////////////////////////////////////

//main code

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

BinaryTreeEg tree = **new** BinaryTreeEg();

//adding elements

tree.add(6);

tree.add(4);

tree.add(8);

tree.add(3);

tree.add(5);

tree.add(7);

tree.add(9);

//printing the elements

tree.printpostorder();

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

///////////////////////////////////////////////////////////

//deleting the elements

tree.containsNode(6);

tree.containsNode(4);

tree.containsNode(1);

}

}