**Lab 10:**

**Question 01:**

class node{

int d,h;

node l,r;

node() {

h=1;

}

}

class bst\_balance{

public node insert(node root,int d){

if(root==null){

node nno=new node();

nno.d=d;

root=nno;

return root; }

else if(root.d>d){

root.l=insert(root.l,d); }

else{

root.r=insert(root.r,d); }

return root;

}

public void inorder(node root){

if(root==null){

return; }

inorder(root.l);

System.out.print(root.d+" ");

inorder(root.r);

}

public int height(node root){

if(root==null){

return 0; }

int lh=height(root.l);

int rh=height(root.r);

return 1+Max(lh,rh);

}

public int balance(node root){

if (root==null){

return 0; }

return height(root.l)-height(root.r);

}

public int Max(int x, int y){

if(x>y){

return x; }

else{

return y; }

}

public boolean c\_balance(node root){

int bal=balance(root);

if(bal<=1&&bal>=-1){

return true; }

return false;

}

}

public class Main{

public static void main(String[] args) {

bst\_balance a=new bst\_balance();

node root=null;

root=a.insert(root,10);

a.insert(root,20);

a.insert(root,30);

a.insert(root,40);

a.insert(root,50);

a.insert(root,45);

a.inorder(root);

System.out.println();

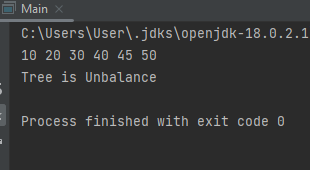
if(a.c\_balance(root)==true){

System.out.println("Tree is Balance"); }

else{

System.out.println("Tree is Unbalance"); }

} }



**Question 02:**

public class Main{

class node{

int d,h;

node l,r;

node(int d){

l=null;

r=null;

this.d=d;

h=1;

}

}

public int height(node root){

if(root==null)

return 0;

return root.h;

}

public int max(int a,int b){

if(a>b){

return a; }

else{

return b; }

}

public node rotate\_right(node root){

node mid=root.l;

node q=mid.r;

mid.r=root;

root.l=q;

return mid;

}

public node rotate\_left(node root){

node mid=root.r;

node w=mid.l;

mid.l=root;

root.r=w;

return mid;

}

public node insert(node root,int val){

if(root==null){

root=new node(val);

return root; }

else if(root.d<val){

root.r=insert(root.r,val); }

else{

root.l=insert(root.l,val); }

root.h=1+max(height(root.l),height(root.r));

int bal\_f=BF(root);

if(bal\_f>1){

if(val<root.l.d){

return rotate\_right(root); }

if(val>root.l.d){

root.l=rotate\_left(root.l);

return rotate\_right(root); }

}

if(bal\_f<-1){

if (val>root.r.d){

return rotate\_left(root); }

else if(val<root.r.d){

root.r=rotate\_right(root.r);

return rotate\_left(root); }

}

return root;

}

public int BF(node root){

return height(root.l)-height(root.r);

}

public void display(node root){

if(root!=null){

display(root.l);

System.out.print(root.d+" , ");

display(root.r);

}

}

public static void main(String[] args) {

Main x= new Main();

node root=null;

int a[]={55,66,77,11,33,22,35,25,44,88,99};

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

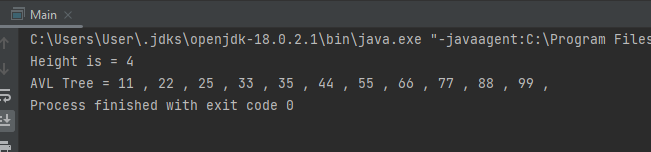
root=x.insert(root,a[i]); }

System.out.println("Height is = "+root.h);

System.out.print("AVL Tree = ");

x.display(root);

} }



**Question 03:**

public class Main{

class node{

int d,h;

node l,r;

node(int d){

l=null;

r=null;

this.d=d;

h=1;

}

}

public int height(node root){

if(root==null)

return 0;

return root.h;

}

public int max(int a,int b){

if(a>b){

return a; }

else{

return b; }

}

public node rotate\_right(node root){

node mid=root.l;

node q=mid.r;

mid.r=root;

root.l=q;

return mid;

}

public node rotate\_left(node root){

node mid=root.r;

node w=mid.l;

mid.l=root;

root.r=w;

return mid;

}

public node insert(node root,int val){

if(root==null){

root=new node(val);

return root; }

else if(root.d<val){

root.r=insert(root.r,val); }

else{

root.l=insert(root.l,val); }

root.h=1+max(height(root.l),height(root.r));

int bal\_f=BF(root);

if(bal\_f>1){

if(val<root.l.d){

return rotate\_right(root); }

if(val>root.l.d){

root.l=rotate\_left(root.l);

return rotate\_right(root); }

}

if(bal\_f<-1){

if (val>root.r.d){

return rotate\_left(root); }

else if(val<root.r.d){

root.r=rotate\_right(root.r);

return rotate\_left(root); }

}

return root;

}

public int BF(node root)

{

return height(root.l)-height(root.r);

}

public void inorder(node root){

if(root!=null){

inorder(root.l);

System.out.print(root.d+" , ");

inorder(root.r);

}

}

public boolean found(node root,int val){

if(root==null){

return false; }

else if(root.d==val){

return true; }

else if(root.d>val){

return found(root.l,val); }

else{

return found(root.r,val); }

}

public void Searching(node root, int val) {

if(found(root,val)==true){

System.out.println("node found"); }

else{

System.out.println("node not found"); }

}

public static void main(String[] args) {

Main x= new Main();

node root=null;

int a[]={55,66,77,11,33,22,35,25,44,88,99};

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

root=x.insert(root,a[i]); }

System.out.println("Height is = "+root.h);

System.out.print("AVL Tree = ");

x.inorder(root);

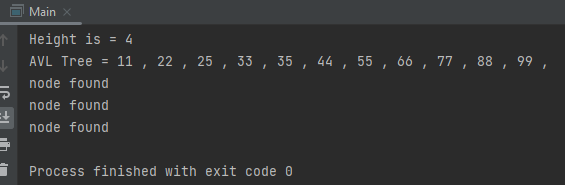
System.out.println();

x.Searching(root,66);

x.Searching(root,22);

x.Searching(root, 44);

} }



**Question 04:**

public class Q4\_Lab10 {

public static void main(String[] args) {

AVIL a = new AVIL();

Tree root=null;

root=a.Insert(root, 55);

root=a.Insert(root, 66);

root=a.Insert(root, 77);

root=a.Insert(root, 11);

root=a.Insert(root, 33);

root=a.Insert(root, 22);

root=a.Insert(root, 35);

root=a.Insert(root, 25);

root=a.Insert(root, 44);

root=a.Insert(root, 88);

root=a.Insert(root, 99);

a.Inorder(root);

System.out.println();

System.out.println("Delete 35");

root=a.Delete(root, 35);

a.Inorder(root);

System.out.println();

System.out.println("Delete 99");

root=a.Delete(root, 99);

a.Inorder(root);

System.out.println();

}

}

class Tree

{

int data;

Tree left;

Tree right;

int height;

Tree()

{

this.height=1;

}

}

class AVIL

{

Tree root;

public int height(Tree root)

{

if(root==null)

{

return 0;

}

return root.height;

}

public int max(int a, int b)

{

if(a>b)

{

return a;

}

else

{

return b;

}

}

public Tree rightrotate(Tree y)

{

Tree x=y.left;

Tree t=x.right;

x.right=y;

y.left=t;

y.height=max(height(y.left),height(y.right))+1;

x.height=max(height(x.left),height(x.right))+1;

return x;

}

public Tree leftrotate(Tree x)

{

Tree y=x.right;

Tree t=y.left;

x.right=t;

y.left=x;

x.height=max(height(x.left),height(x.right))+1;

y.height=max(height(y.left),height(y.right))+1;

return y;

}

public int getbalance(Tree root)

{

if(root==null)

{

return 0;

}

return height(root.left)-height(root.right);

}

public Tree Insert(Tree root, int key)

{

if(root==null)

{

Tree t = new Tree();

t.data=key;

root=t;

return root;

}

if(root.data>key)

{

root.left=Insert(root.left, key);

}

else if(root.data<key)

{

root.right=Insert(root.right, key);

}

else

{

return root;

}

root.height=1+max(height(root.left),height(root.right));

int balance=getbalance(root);

if(balance>1&&key<root.left.data)

{

return rightrotate(root);

}

if(balance<-1&&key>root.right.data)

{

return leftrotate(root);

}

if(balance>1&&key>root.left.data)

{

root.left=leftrotate(root.left);

return rightrotate(root);

}

if(balance<-1&&key<root.right.data)

{

root.right=rightrotate(root.right);

return leftrotate(root);

}

return root;

}

public void Inorder(Tree root)

{

if(root==null)

{

return;

}

Inorder(root.left);

System.out.print(root.data+" ");

Inorder(root.right);

}

public Tree Delete(Tree root, int value)

{

if(root==null)

{

return null;

}

else if(root.data>value)

{

root.left=Delete(root.left,value);

}

else if(root.data<value)

{

root.right=Delete(root.right,value);

}

else

{

if(root.left==null)

{

return root.right;

}

if(root.right==null)

{

return root.left;

}

root.data=InorderSuccessor(root.right);

root.right=Delete(root.right, root.data);

}

if(root==null)

{

return root;

}

root.height=1+max(height(root.left),height(root.right));

int balances=getbalance(root);

if(balances>1&&root.left.data<0)

{

return rightrotate(root);

}

else if(balances<-1&&root.right.data>0)

{

return leftrotate(root);

}

else if(balances>1&&root.left.data>0)

{

root.left=leftrotate(root.left);

return rightrotate(root);

}

else if(balances<-1&&root.right.data<0)

{

root.right=rightrotate(root.right);

return leftrotate(root);

}

return root;

}

public int InorderSuccessor(Tree root)

{

int succ=root.data;

while (root.left!=null)

{

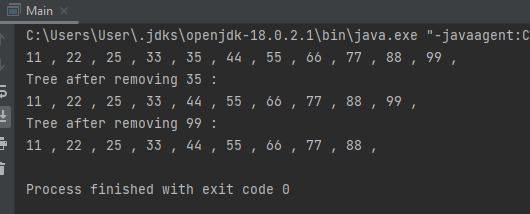
succ =root.left.data;

root=root.left;

}

return succ;

} }



**Question 05:**

class node{

int d,h;

node l,r;

node() {

h=1;

}

}

class Avl{

node root;

public int height(node root){

if(root==null) {

return 0;}

return root.h;

}

public int max(int a,int b){

if(a>b){

return a; }

else{

return b; }

}

public node rotate\_right(node root){

node mid=root.l;

node q=mid.r;

mid.r=root;

root.l=q;

root.h=max(height(root.l),height(root.r))+1;

mid.h=max(height(mid.l),height(mid.r))+1;

return mid;

}

public node rotate\_left(node root){

node mid=root.r;

node t=mid.l;

root.r=t;

mid.l=root;

root.h=max(height(root.l),height(root.r))+1;

mid.h=max(height(mid.l),height(mid.r))+1;

return mid;

}

public int BF(node root){

if(root==null){

return 0; }

return height(root.l)-height(root.r);

}

public node insert(node root, int val){

if(root==null){

node z=new node();

z.d=val;

root=z;

return root; }

if(root.d>val){

root.l=insert(root.l,val); }

else if(root.d<val){

root.r=insert(root.r,val); }

else{

return root; }

root.h=1+max(height(root.l),height(root.r));

int bal=BF(root);

if(bal>1&&val<root.l.d){

return rotate\_right(root); }

if(bal<-1&&val>root.r.d){

return rotate\_left(root); }

if(bal>1&&val>root.l.d){

root.l=rotate\_left(root.l);

return rotate\_right(root); }

if(bal<-1&&val<root.r.d){

root.r=rotate\_right(root.r);

return rotate\_left(root); }

return root;

}

// public void inorder(node root){

// if(root==null){

// return; }

// inorder(root.l);

// System.out.print(root.d+" , ");

// inorder(root.r);

// }

public node delete(node root,int val){

if(root==null){

return null; }

else if(root.d>val){

root.l=delete(root.l,val); }

else if(root.d<val){

root.r=delete(root.r,val); }

else{

if(root.l==null){

return root.r; }

if(root.r==null){

return root.l; }

root.d=InorderSuccessor(root.r);

root.r=delete(root.r,root.d); }

if(root==null){

return root; }

root.h=1+max(height(root.l),height(root.r));

int bal=BF(root);

if(bal>1&&root.l.d<0){

return rotate\_right(root); }

else if(bal<-1&&root.r.d>0){

return rotate\_left(root); }

else if(bal>1&&root.l.d>0){

root.l=rotate\_left(root.l);

return rotate\_right(root); }

else if(bal<-1&&root.r.d<0){

root.r=rotate\_right(root.r);

return rotate\_left(root); }

return root;

}

public int InorderSuccessor(node root){

int s=root.d;

while(root.l!=null){

s=root.l.d;

root=root.l; }

return s;

}

public void inorder(node root){

if(root==null){

return; }

inorder(root.l);

System.out.print(root.d+" ");

inorder(root.r);

}

public void preorder(node root){

if(root==null){

return; }

System.out.print(root.d+" ");

inorder(root.l);

inorder(root.r);

}

public void postorder(node root){

if(root==null){

return; }

inorder(root.l);

inorder(root.r);

System.out.print(root.d+" ");

}

}

public class Main {

public static void main(String[] args) {

Avl x=new Avl();

node root=null;

root=x.insert(root, 55);

root=x.insert(root, 66);

root=x.insert(root, 77);

root=x.insert(root, 11);

root=x.insert(root, 33);

root=x.insert(root, 22);

root=x.insert(root, 35);

root=x.insert(root, 25);

root=x.insert(root, 44);

root=x.insert(root, 88);

root=x.insert(root, 99);

x.inorder(root);

System.out.println();

System.out.println("Tree after removing 35 :");

root=x.delete(root, 35);

x.inorder(root);

System.out.println();

System.out.println("Tree after removing 99 :");

root=x.delete(root, 99);

x.inorder(root);

System.out.println();

System.out.println("Inorder :");

root=x.delete(root, 35);

root=x.delete(root, 99);

x.inorder(root);

System.out.println(" ");

System.out.println("Postorder :");

x.postorder(root);

root=x.delete(root, 35);

root=x.delete(root, 99);

System.out.println(" ");

System.out.println("Pre order :");

root=x.delete(root, 35);

root=x.delete(root, 99);

x.preorder(root);

} }

