A screenshot of a computer

Description automatically generated

***Tree***

class HelloWorld {

public static void main(String[] args) {

Node root=new Node(1);

root.n1=new Node(2);

root.n2=new Node(3);

System.out.println(root.n1.data);

}

}

class Node

{

int data;

Node n1=null,n2=null;

Node(int data)

{

this.data=data;

}

}

***Binary Tree***

class HelloWorld {

public static void main(String[] args) {

Tree btree=new Tree();

Node root=null;

root= btree.addNode(root,4);

btree.addNode(root,7);

btree.addNode(root,2);

System.out.println(root.right.data);

btree.contains(root,2);

btree.PreOrder(root);

System.out.println();

btree.PostOrder(root);

System.out.println();

btree.InOrder(root);

}

}

class Node

{

int data;

Node left,right;

Node(int data)

{

this.data=data;

left=right=null;

}

}

class Tree

{

Node addNode(Node root,int value)

{

if(root==null)

{

return new Node(value);

}

else if(value<root.data)

{

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

}

else if(value>root.data)

{

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

}

else

{

return root;

}

return root;

}

void contains(Node root,int key)

{

if(root!=null)

{

if(root.data==key)

{

System.out.println("Key found");

}

contains(root.left,key);

contains(root.right,key);

}

}

void PreOrder(Node root)

{

if(root!=null)

{

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

PreOrder(root.left);

PreOrder(root.right);

}

}

void PostOrder(Node root)

{

if(root!=null)

{

PostOrder(root.left);

PostOrder(root.right);

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

}

}

void InOrder(Node root)

{

if(root!=null)

{

InOrder(root.left);

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

InOrder(root.right);

}

}

}

A screenshot of a computer

Description automatically generated

***Preoeder with formulas:***

***L = [index\*2]+1***

***L = [index\*2]+2***

***Parent = (index+1)/2***

class HelloWorld {

public static void main(String[] args) {

Btree tree=new Btree();

int arr[]={3,2,5,4,6,7,1};

Node root=null;

root=tree.addNode(root,arr,0);

tree.preOrder(root);

}

}

class Node

{

int data;

Node left,right;

Node(int data)

{

this.data=data;

left=right=null;

}

}

class Btree

{

Node root=null;

Node addNode(Node root,int arr[],int index)

{

if(index<arr.length)

{

Node temp=new Node(arr[index]);

root=temp;

root.left=addNode(root.left,arr,(2\*index)+1);

root.right=addNode(root.right,arr,(2\*index)+2);

}

return root;

}

void preOrder(Node root)

{

if(root!=null)

{

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

preOrder(root.left);

preOrder(root.right);

}

}

}

A screenshot of a computer program

Description automatically generated

class HelloWorld {

public static void main(String[] args) {

Btree tree=new Btree();

int arr[]={3,2,5,4,6,7,1};

Node root=null;

root=tree.addNode(root,arr,0);

tree.preOrder(root);

System.out.println();

System.out.println("Height: "+tree.height(root));

}

}

class Node

{

int data;

Node left,right;

Node(int data)

{

this.data=data;

left=right=null;

}

}

class Btree

{

Node root=null;

Node addNode(Node root,int arr[],int index)

{

if(index<arr.length)

{

Node temp=new Node(arr[index]);

root=temp;

root.left=addNode(root.left,arr,(2\*index)+1);

root.right=addNode(root.right,arr,(2\*index)+2);

}

return root;

}

void preOrder(Node root)

{

if(root!=null)

{

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

preOrder(root.left);

preOrder(root.right);

}

}

int height(Node root)

{

if(root==null)

{

return -1;

}

int left=height(root.left);

int right=height(root.right);

return Math.max(left,right)+1;

}

}

A screenshot of a computer program

Description automatically generated

class HelloWorld {

public static void main(String[] args) {

Btree tree=new Btree();

int arr[]={3,2,5,4,6,7,1};

Node root=null;

root=tree.addNode(root,arr,0);

tree.preOrder(root);

System.out.println();

System.out.println("Height: "+tree.Height(root));

int h=tree.Height(root);

tree.LevelOrder(root,h);

}

}

class Node

{

int data;

Node left,right;

Node(int data)

{

this.data=data;

left=right=null;

}

}

class Btree

{

Node root=null;

Node addNode(Node root,int arr[],int index)

{

if(index<arr.length)

{

Node temp=new Node(arr[index]);

root=temp;

root.left=addNode(root.left,arr,(2\*index)+1);

root.right=addNode(root.right,arr,(2\*index)+2);

}

return root;

}

void preOrder(Node root)

{

if(root!=null)

{

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

preOrder(root.left);

preOrder(root.right);

}

}

int Height(Node root)

{

if(root==null)

{

return -1;

}

int left=Height(root.left);

int right=Height(root.right);

return Math.max(left,right)+1;

}

void LevelOrder(Node root,int height)

{

for(int i=0;i<=height;i++)

{

PrintLevel(root,i);

}

}

void PrintLevel(Node root,int level)

{

if(root!=null)

{

if(level==0)

{

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

}

else

{

PrintLevel(root.left,level-1);

PrintLevel(root.right,level-1);

}

}

}

}

A screenshot of a computer program

Description automatically generated

import java.util.Queue;

import java.util.LinkedList;

class HelloWorld {

public static void main(String[] args) {

Btree tree=new Btree();

int arr[]={3,2,5,4,6,7,1};

Node root=null;

root=tree.addNode(root,arr,0);

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

tree.preOrder(root);

System.out.println();

System.out.println("Height: "+tree.Height(root));

int h=tree.Height(root);

tree.LevelOrder(root,h);

System.out.println();

System.out.println("Print Level Without Height");

tree.PrintLevelWithoutHeight(root);

}

}

class Node

{

int data;

Node left,right;

Node(int data)

{

this.data=data;

left=right=null;

}

}

class Btree

{

Node root=null;

Node addNode(Node root,int arr[],int index)

{

if(index<arr.length)

{

Node temp=new Node(arr[index]);

root=temp;

root.left=addNode(root.left,arr,(2\*index)+1);

root.right=addNode(root.right,arr,(2\*index)+2);

}

return root;

}

void preOrder(Node root)

{

if(root!=null)

{

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

preOrder(root.left);

preOrder(root.right);

}

}

int Height(Node root)

{

if(root==null)

{

return -1;

}

int left=Height(root.left);

int right=Height(root.right);

return Math.max(left,right)+1;

}

void LevelOrder(Node root,int height)

{

for(int i=0;i<=height;i++)

{

PrintLevel(root,i);

}

}

void PrintLevel(Node root,int level)

{

if(root!=null)

{

if(level==0)

{

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

}

else

{

PrintLevel(root.left,level-1);

PrintLevel(root.right,level-1);

}

}

}

void PrintLevelWithoutHeight(Node root)

{

Queue<Node> queue = new LinkedList<>();

queue.add(root);

while(!queue.isEmpty())

{

Node tempNode=queue.poll();

if(tempNode.left!=null)

{

queue.add(tempNode.left);

}

if(tempNode.right!=null)

{

queue.add(tempNode.right);

}

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

}

}

}

A screenshot of a computer program

Description automatically generated

import java.util.Queue;

import java.util.LinkedList;

class HelloWorld {

public static void main(String[] args) {

Btree tree=new Btree();

int arr[]={3,2,5,4,6,7,1};

Node root=null;

root=tree.addNode(root,arr,0);

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

tree.preOrder(root);

System.out.println();

System.out.println("Height: "+tree.Height(root));

int h=tree.Height(root);

tree.LevelOrder(root,h);

System.out.println();

System.out.println("Print Level Without Height");

tree.PrintLevelWithoutHeight(root);

System.out.println();

System.out.println(tree.PerfectBinaryTree(root));

tree.Mirror(root);

tree.LevelOrder(root,h);

}

}

class Node

{

int data;

Node left,right;

Node(int data)

{

this.data=data;

left=right=null;

}

}

class Btree

{

Node root=null;

Node addNode(Node root,int arr[],int index)

{

if(index<arr.length)

{

Node temp=new Node(arr[index]);

root=temp;

root.left=addNode(root.left,arr,(2\*index)+1);

root.right=addNode(root.right,arr,(2\*index)+2);

}

return root;

}

void preOrder(Node root)

{

if(root!=null)

{

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

preOrder(root.left);

preOrder(root.right);

}

}

int Height(Node root)

{

if(root==null)

{

return -1;

}

int left=Height(root.left);

int right=Height(root.right);

return Math.max(left,right)+1;

}

void LevelOrder(Node root,int height)

{

for(int i=0;i<=height;i++)

{

PrintLevel(root,i);

}

}

void PrintLevel(Node root,int level)

{

if(root!=null)

{

if(level==0)

{

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

}

else

{

PrintLevel(root.left,level-1);

PrintLevel(root.right,level-1);

}

}

}

void PrintLevelWithoutHeight(Node root)

{

Queue<Node> queue = new LinkedList<>();

queue.add(root);

while(!queue.isEmpty())

{

Node tempNode=queue.poll();

if(tempNode.left!=null)

{

queue.add(tempNode.left);

}

if(tempNode.right!=null)

{

queue.add(tempNode.right);

}

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

}

}

int TotalNodes(Node root)

{

if(root==null)

{

return 0;

}

return 1+TotalNodes(root.left)+TotalNodes(root.right);

}

boolean PerfectBinaryTree(Node root)

{

Btree tree=new Btree();

int temp=tree.TotalNodes(root);

int h=tree.Height(root);

double f=Math.pow(2,(h+1))-1;

if(temp==f)

{

return true;

}

else

{

return false;

}

}

void Mirror(Node root)

{

Node temp;

if(root!=null)

{

temp=root.left;

root.left=root.right;

root.right=temp;

Mirror(root.left);

Mirror(root.right);

}

}

}

A screenshot of a computer program

Description automatically generated

import java.util.\*;

class HelloWorld {

public static void main(String[] args) {

Btree tree=new Btree();

int arr[]={3,2,5,4,6,7,1};

Node root=null;

root=tree.addNode(root,arr,0);

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

tree.preOrder(root);

System.out.println();

System.out.println("Height: "+tree.Height(root));

int h=tree.Height(root);

tree.LevelOrder(root,h);

System.out.println();

System.out.println("Print Level Without Height");

tree.PrintLevelWithoutHeight(root);

System.out.println();

System.out.println("Is perfect Binary Tree: "+tree.PerfectBinaryTree(root));

System.out.println("Left view of the tree");

tree.leftView(root);

System.out.println();

System.out.println("Right view of the tree");

tree.rightView(root);

System.out.println();

System.out.println("Internal nodes other than left and right");

tree.InternalNodes(root);

System.out.println();

System.out.println("Top view of the Tree");

tree.TopView(root);

System.out.println();

System.out.println("Mirrored Tree");

tree.Mirror(root);

tree.LevelOrder(root,h);

System.out.println();

}

}

class Node

{

int data;

Node left,right;

Node(int data)

{

this.data=data;

left=right=null;

}

}

class Btree

{

Node root=null;

Node addNode(Node root,int arr[],int index)

{

if(index<arr.length)

{

Node temp=new Node(arr[index]);

root=temp;

root.left=addNode(root.left,arr,(2\*index)+1);

root.right=addNode(root.right,arr,(2\*index)+2);

}

return root;

}

void preOrder(Node root)

{

if(root!=null)

{

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

preOrder(root.left);

preOrder(root.right);

}

}

int Height(Node root)

{

if(root==null)

{

return -1;

}

int left=Height(root.left);

int right=Height(root.right);

return Math.max(left,right)+1;

}

void LevelOrder(Node root,int height)

{

for(int i=0;i<=height;i++)

{

PrintLevel(root,i);

}

}

void PrintLevel(Node root,int level)

{

if(root!=null)

{

if(level==0)

{

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

}

else

{

PrintLevel(root.left,level-1);

PrintLevel(root.right,level-1);

}

}

}

void PrintLevelWithoutHeight(Node root)

{

Queue<Node> queue = new LinkedList<>();

queue.add(root);

while(!queue.isEmpty())

{

Node tempNode=queue.poll();

if(tempNode.left!=null)

{

queue.add(tempNode.left);

}

if(tempNode.right!=null)

{

queue.add(tempNode.right);

}

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

}

}

int TotalNodes(Node root)

{

if(root==null)

{

return 0;

}

return 1+TotalNodes(root.left)+TotalNodes(root.right);

}

boolean PerfectBinaryTree(Node root)

{

Btree tree=new Btree();

int temp=tree.TotalNodes(root);

int h=tree.Height(root);

double f=Math.pow(2,(h+1))-1;

if(temp==f)

{

return true;

}

else

{

return false;

}

}

void Mirror(Node root)

{

Node temp;

if(root!=null)

{

temp=root.left;

root.left=root.right;

root.right=temp;

Mirror(root.left);

Mirror(root.right);

}

}

void leftView(Node root)

{

Queue<Node> queue = new LinkedList<>();

queue.add(root);

while(!queue.isEmpty())

{

int n=queue.size();

for(int i=0;i<n;i++)

{

Node temp=queue.poll();

if(i==0)

{

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

}

if(temp.left!=null)

{

queue.add(temp.left);

queue.add(temp.right);

}

}

}

}

void rightView(Node root)

{

Queue<Node> queue = new LinkedList<>();

queue.add(root);

while(!queue.isEmpty())

{

int n=queue.size();

for(int i=0;i<n;i++)

{

Node temp=queue.poll();

if(i==n-1)

{

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

}

if(temp.left!=null)

{

queue.add(temp.left);

queue.add(temp.right);

}

}

}

}

void InternalNodes(Node root)

{

Queue<Node> queue = new LinkedList<>();

queue.add(root);

while(!queue.isEmpty())

{

int n=queue.size();

for(int i=0;i<n;i++)

{

Node temp=queue.poll();

if(i!=n-1&&i!=0)

{

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

}

if(temp.left!=null)

{

queue.add(temp.left);

queue.add(temp.right);

}

}

}

}

void TopView(Node root)

{

ArrayDeque<Node> queue = new ArrayDeque<>();

Node temp=root.left;

while(temp!=null)

{

queue.addFirst(temp);

temp=temp.left;

}

while(!queue.isEmpty())

{

temp=queue.remove();

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

}

temp=root;

while(temp!=null)

{

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

temp=temp.right;

}

}

}

A screenshot of a computer program

Description automatically generated