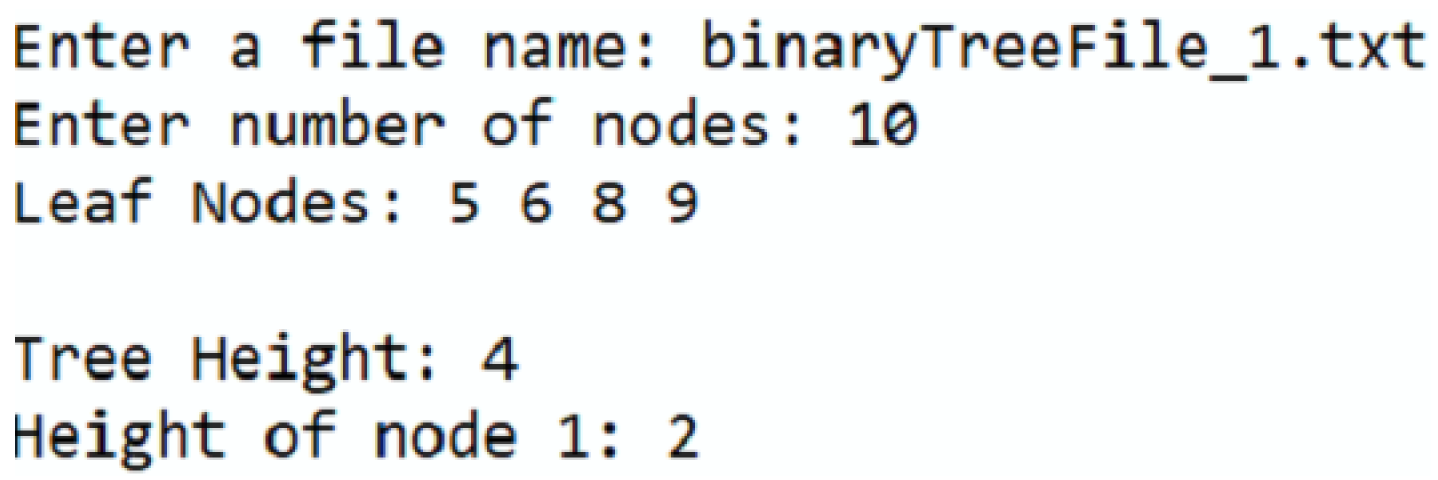
```java
import java.io.*;
import java.util.*;

class BTNode{

    private int nodeid;
    private int data;
    private int levelNum;
    private BTNode leftChildPtr;
    private BTNode rightChildPtr;

    public BTNode(){}

    public  void setNodeId(int id){
        nodeid = id;
    }

    public  int getNodeId(){
        return nodeid;
    }

    public  void setData(int d){
        data = d;
    }

    public  int getData(){
        return data;
    }

    public  void setLevelNum(int level){
        levelNum = level;
    }

    public  int getLevelNum(){
        return levelNum;
    }

    public  void setLeftChildPtr(BTNode ptr){
        leftChildPtr = ptr;
    }

    public  void setRightChildPtr(BTNode ptr){
        rightChildPtr = ptr;
    }

    public  BTNode getLeftChildPtr(){
        return leftChildPtr;
    }

    public  BTNode getRightChildPtr(){
        return rightChildPtr;
    }

    public  int getLeftChildID(){
        if (leftChildPtr == null)
            return -1;

        return leftChildPtr.getNodeId();
    }

    public  int getRightChildID(){
        if (rightChildPtr == null)
            return -1;

        return rightChildPtr.getNodeId();
    }
}





class BinaryTree{

    private int numNodes;
    private BTNode arrayOfBTNodes[];

    public BinaryTree(int n){
        numNodes = n;
        arrayOfBTNodes = new BTNode[numNodes];

        for (int id = 0; id < numNodes; id++){
            arrayOfBTNodes[id] = new BTNode();
            arrayOfBTNodes[id].setNodeId(id);
            arrayOfBTNodes[id].setLevelNum(-1);
            arrayOfBTNodes[id].setLeftChildPtr(null);
            arrayOfBTNodes[id].setRightChildPtr(null);
        }
    }

    public  void setLeftLink(int upstreamNodeID, int downstreamNodeID){
        arrayOfBTNodes[upstreamNodeID].setLeftChildPtr(arrayOfBTNodes[downstreamNodeID]);
    }

    public  void setRightLink(int upstreamNodeID, int downstreamNodeID){

        arrayOfBTNodes[upstreamNodeID].setRightChildPtr(arrayOfBTNodes[downstreamNodeID])
        ;
    }


    public void printLeafNodes(){

        for (int id = 0; id < numNodes; id++){

            if (arrayOfBTNodes[id].getLeftChildPtr() == null &&
            arrayOfBTNodes[id].getRightChildPtr() == null)
                System.out.print(id + " ");
            }

        System.out.println();
    }


    public  boolean isLeafNode(int nodeid){

        if (arrayOfBTNodes[nodeid].getLeftChildPtr() == null &&
        arrayOfBTNodes[nodeid].getRightChildPtr() == null)
            return true;

        return false;
    }


    public  int getNodeHeight(int nodeid){

        if (nodeid == -1)
```

```java
                return -1;

            if (isLeafNode(nodeid) )
                return 0;

            int leftChildID = arrayOfBTNodes[nodeid].getLeftChildID(); // -1 if not exist
            int rightChildID = arrayOfBTNodes[nodeid].getRightChildID(); // -1 if not exist

            return Math.max(getNodeHeight(leftChildID), getNodeHeight(rightChildID)) + 1;

        }


        public  int getTreeHeight(){
            return getNodeHeight(0);
        }


    }


    class BinaryTreeImplementation{

        public static void main(String[] args){

        try{

        Scanner input = new Scanner(System.in);

        String filename;
        System.out.print("Enter a file name: ");
        filename = input.next();

        int numNodes;
        System.out.print("Enter number of nodes: ");
        numNodes = input.nextInt();

        BinaryTree binaryTree = new BinaryTree(numNodes);

        FileReader fr = new FileReader(filename);
        BufferedReader br = new BufferedReader(fr);

        String line = null;

        while ( (line = br.readLine()) != null){

            StringTokenizer stk = new StringTokenizer(line, ",: ");

            int upstreamNodeID = Integer.parseInt(stk.nextToken());

            int childIndex = 0;

            while (stk.hasMoreTokens()){

                int downstreamNodeID = Integer.parseInt(stk.nextToken());

                if (childIndex == 0 && downstreamNodeID != -1)
                    binaryTree.setLeftLink(upstreamNodeID, downstreamNodeID);

                if (childIndex == 1 && downstreamNodeID != -1)
                    binaryTree.setRightLink(upstreamNodeID, downstreamNodeID);

                childIndex++;

            }

        }


        System.out.print("Leaf Nodes: ");
        binaryTree.printLeafNodes();
        System.out.println();

        System.out.println("Tree Height: " + binaryTree.getTreeHeight() );
        System.out.println("Height of node 1: " + binaryTree.getNodeHeight(1) );

        }
        catch(Exception e){e.printStackTrace();}

        }
    }
```



```
Enter a file name: binaryTreeFile_1.txt
Enter number of nodes: 10
Leaf Nodes: 5 6 8 9

Tree Height: 4
Height of node 1: 2
```