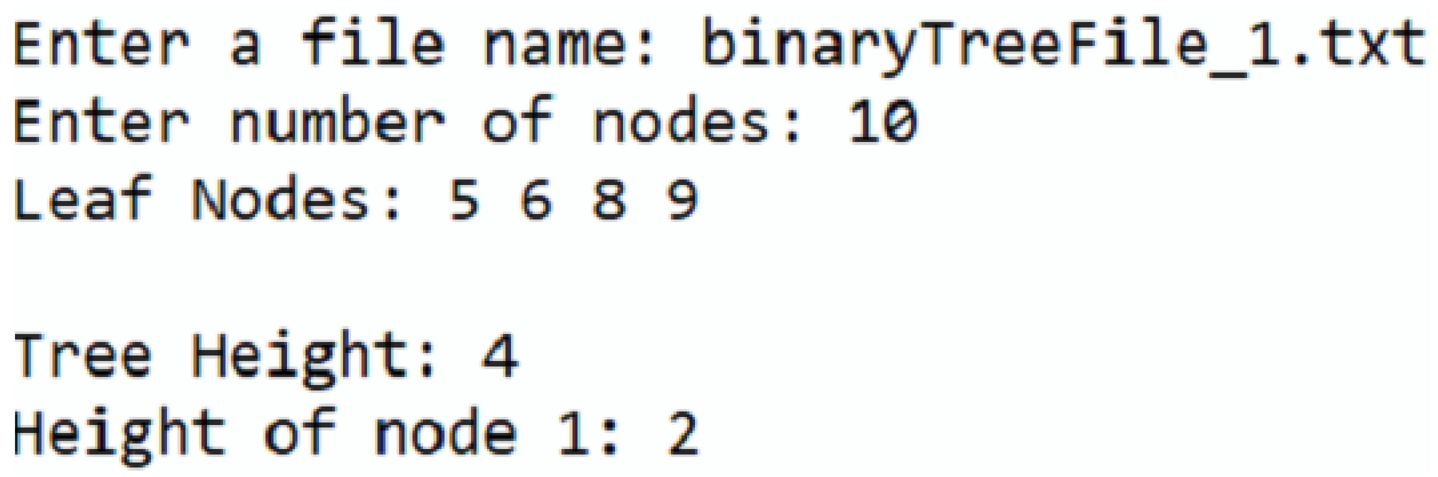
```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <cstring> // for string tokenizer and c-style string processing
#include <algorithm> // max function

using namespace std;

class BTNode{

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

    public:

        BTNode(){}

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

        int getNodeId(){
            return nodeid;
        }

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

        int getData(){
            return data;
        }

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

        int getLevelNum(){
            return levelNum;
        }

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

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

        BTNode* getLeftChildPtr(){
            return leftChildPtr;
        }

        BTNode* getRightChildPtr(){
            return rightChildPtr;
        }

        int getLeftChildID(){
            if (leftChildPtr == 0)
                return -1;

            return leftChildPtr->getNodeId();
        }

        int getRightChildID(){
            if (rightChildPtr == 0)
                return -1;

            return rightChildPtr->getNodeId();
        }
};






class BinaryTree{

    private:
        int numNodes;
        BTNode* arrayOfBTNodes;

    public:

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

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

        void setLeftLink(int upstreamNodeID, int downstreamNodeID){

            arrayOfBTNodes[upstreamNodeID].setLeftChildPtr(&arrayOfBTNodes[downstreamNodeID]);
        }

        void setRightLink(int upstreamNodeID, int downstreamNodeID){

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

        void printLeafNodes(){

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

                if (arrayOfBTNodes[id].getLeftChildPtr() == 0 && arrayOfBTNodes[id].getRightChildPtr() == 0)
                    cout << id << " ";
            }

            cout << endl;
        }


        bool isLeafNode(int nodeid){

            if (arrayOfBTNodes[nodeid].getLeftChildPtr() == 0 && arrayOfBTNodes[nodeid].getRightChildPtr() == 0)
                return true;

            return false;
        }

        int getNodeHeight(int nodeid){

            if (nodeid == -1)
                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 max(getNodeHeight(leftChildID), getNodeHeight(rightChildID)) + 1;

        }


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


};



int main(){

    string filename;
    cout << "Enter a file name: ";
    cin >> filename;

    int numNodes;
    cout << "Enter number of nodes: ";
    cin >> numNodes;

    BinaryTree binaryTree(numNodes);

    ifstream fileReader(filename);

    if (!fileReader){
        cout << "File cannot be opened!! ";
        return 0;
    }

    int numCharsPerLine = 10;

    char *line = new char[numCharsPerLine];
    // '10' is the maximum number of characters per line

    fileReader.getline(line, numCharsPerLine, '\n');
    // '\n' is the delimiting character to stop reading the line

    while (fileReader){

        char* cptr = strtok(line, ",: ");

        string upstreamNodeToken(cptr);
        int upstreamNodeID = stoi(upstreamNodeToken);

        cptr = strtok(NULL, ",: ");

        int childIndex = 0; // 0 for left child; 1 for right child

        while (cptr != 0){

            string downstreamNodeToken(cptr);
            int downstreamNodeID = stoi(downstreamNodeToken);

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

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

            cptr = strtok(NULL, ",: ");
            childIndex++;
        }

        fileReader.getline(line, numCharsPerLine, '\n');

    }


    cout << "Leaf Nodes: ";
    binaryTree.printLeafNodes();
    cout << endl;

    cout << "Tree Height: " << binaryTree.getTreeHeight() << endl;
    cout << "Height of node 1: " << binaryTree.getNodeHeight(1) << endl;

    return 0;
}
```



```
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
```