

```
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  #include <cstring> // for string tokenizer and c-style string processing
5  #include <algorithm> // max function
6
7  using namespace std;
8
9  class BTreeNode{
10
11     private:
12         int nodeid;
13         int data;
14         int levelNum;
15         BTreeNode* leftChildPtr;
16         BTreeNode* rightChildPtr;
17
18     public:
19
20         BTreeNode(){}
21
22         void setNodeid(int id){
23             nodeid = id;
24         }
25
26         int getNodeid(){
27             return nodeid;
28         }
29
30         void setData(int d){
31             data = d;
32         }
33
34         int getData(){
35             return data;
36         }
37
38         void setLevelNum(int level){
39             levelNum = level;
40         }
41
42         int getLevelNum(){
43             return levelNum;
44         }
45
46         void setLeftChildPtr(BTreeNode* ptr){
47             leftChildPtr = ptr;
48         }
49
50         void setRightChildPtr(BTreeNode* ptr){
51             rightChildPtr = ptr;
52         }
53
54         BTreeNode* getLeftChildPtr(){
55             return leftChildPtr;
56         }
57
58         BTreeNode* getRightChildPtr(){
59             return rightChildPtr;
60         }
61
62         int getLeftChildID(){
63             if (leftChildPtr == 0)
64                 return -1;
65
66             return leftChildPtr->getNodeid();
```

```

67     }
68
69     int getRightChildID(){
70         if (rightChildPtr == 0)
71             return -1;
72
73         return rightChildPtr->getNodeId();
74     }
75 };
76
77
78
79
80
81
82 class BinaryTree{
83
84     private:
85         int numNodes;
86         BTreeNode* arrayOfBTNodes;
87
88     public:
89
90         BinaryTree(int n){
91             numNodes = n;
92             arrayOfBTNodes = new BTreeNode[numNodes];
93
94             for (int id = 0; id < numNodes; id++){
95                 arrayOfBTNodes[id].setNodeId(id);
96                 arrayOfBTNodes[id].setLevelNum(-1);
97                 arrayOfBTNodes[id].setLeftChildPtr(0);
98                 arrayOfBTNodes[id].setRightChildPtr(0);
99             }
100     }
101
102     void setLeftLink(int upstreamNodeID, int downstreamNodeID){
103
104         arrayOfBTNodes[upstreamNodeID].setLeftChildPtr(&arrayOfBTNodes[downstreamNode
105             ID]);
106     }
107
108     void setRightLink(int upstreamNodeID, int downstreamNodeID){
109
110         arrayOfBTNodes[upstreamNodeID].setRightChildPtr(&arrayOfBTNodes[downstreamNode
111             ID]);
112     }
113
114     void printLeafNodes(){
115
116         for (int id = 0; id < numNodes; id++){
117
118             if (arrayOfBTNodes[id].getLeftChildPtr() == 0 &&
119                 arrayOfBTNodes[id].getRightChildPtr() == 0)
120                 cout << id << " ";
121         }
122
123         cout << endl;
124     }
125
126     bool isLeafNode(int nodeid){
127
128         if (arrayOfBTNodes[nodeid].getLeftChildPtr() == 0 &&
129             arrayOfBTNodes[nodeid].getRightChildPtr() == 0)
130             return true;

```

```

127         return false;
128     }
129
130     int getNodeHeight(int nodeid){
131
132         if (nodeid == -1)
133             return -1;
134
135         if (isLeafNode(nodeid) )
136             return 0;
137
138         int leftChildID = arrayOfBTNodes[nodeid].getLeftChildID(); // -1 if not exist
139         int rightChildID = arrayOfBTNodes[nodeid].getRightChildID(); // -1 if not
140         exist
141
142         return max(getNodeHeight(leftChildID), getNodeHeight(rightChildID)) + 1;
143     }
144
145
146     int getTreeHeight(){
147         return getNodeHeight(0);
148     }
149
150
151 };
152
153
154
155 int main(){
156
157     string filename;
158     cout << "Enter a file name: ";
159     cin >> filename;
160
161     int numNodes;
162     cout << "Enter number of nodes: ";
163     cin >> numNodes;
164
165     BinaryTree binaryTree(numNodes);
166
167     ifstream fileReader(filename);
168
169     if (!fileReader){
170         cout << "File cannot be opened!! ";
171         return 0;
172     }
173
174     int numCharsPerLine = 10;
175
176     char *line = new char[numCharsPerLine];
177     // '10' is the maximum number of characters per line
178
179     fileReader.getline(line, numCharsPerLine, '\n');
180     // '\n' is the delimiting character to stop reading the line
181
182     while (fileReader){
183
184         char* cptr = strtok(line, ",: ");
185
186         string upstreamNodeToken(cptr);
187         int upstreamNodeID = stoi(upstreamNodeToken);
188
189         cptr = strtok(NULL, ",: ");
190
191         int childIndex = 0; // 0 for left child; 1 for right child

```

```

192
193     while (cptr != 0){
194
195         string downstreamNodeToken(cptr);
196         int downstreamNodeID = stoi(downstreamNodeToken);
197
198         if (childIndex == 0 && downstreamNodeID != -1)
199             binaryTree.setLeftLink(upstreamNodeID, downstreamNodeID);
200
201         if (childIndex == 1 && downstreamNodeID != -1)
202             binaryTree.setRightLink(upstreamNodeID, downstreamNodeID);
203
204         cptr = strtok(NULL, ",: ");
205         childIndex++;
206     }
207
208     fileReader.getline(line, numCharsPerLine, '\n');
209
210 }
211
212
213 cout << "Leaf Nodes: ";
214 binaryTree.printLeafNodes();
215 cout << endl;
216
217 cout << "Tree Height: " << binaryTree.getTreeHeight() << endl;
218 cout << "Height of node 1: " << binaryTree.getNodeHeight(1) << endl;
219
220 return 0;
221 }

```

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