```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;
```

```
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  class BinaryTree{
80
81      private:
82          int numNodes;
83          BTNode* arrayOfBTNodes;
84
85      public:
86
87          BinaryTree(int n){
88              numNodes = n;
89              arrayOfBTNodes = new BTNode[numNodes];
90
91              for (int id = 0; id < numNodes; id++){
92                  arrayOfBTNodes[id].setNodeId(id);
93                  arrayOfBTNodes[id].setLevelNum(-1);
94                  arrayOfBTNodes[id].setLeftChildPtr(0);
95                  arrayOfBTNodes[id].setRightChildPtr(0);
96              }
97          }
98
99          void setLeftLink(int upstreamNodeID, int downstreamNodeID){
100
                arrayOfBTNodes[upstreamNodeID].setLeftChildPtr(&arrayOfBTNodes[downstreamNode
                ID]);
101         }
102
103         void setRightLink(int upstreamNodeID, int downstreamNodeID){
104
                arrayOfBTNodes[upstreamNodeID].setRightChildPtr(&arrayOfBTNodes[downstreamNod
                eID]);
105         }
106
107         void printLeafNodes(){
108
109             for (int id = 0; id < numNodes; id++){
110
111                 if (arrayOfBTNodes[id].getLeftChildPtr() == 0 &&
                    arrayOfBTNodes[id].getRightChildPtr() == 0)
112                     cout << id << " ";
113             }
114
115             cout << endl;
116         }
117
118
119         bool isLeafNode(int nodeid){
120
121             if (arrayOfBTNodes[nodeid].getLeftChildPtr() == 0 &&
                arrayOfBTNodes[nodeid].getRightChildPtr() == 0)
122                 return true;
```

```
123
124             return false;
125         }
126
127         int getNodeHeight(int nodeid){
128
129             if (nodeid == -1)
130                 return -1;
131
132             if (isLeafNode(nodeid) )
133                 return 0;
134
135             int leftChildID = arrayOfBTNodes[nodeid].getLeftChildID(); // -1 if not exist
136             int rightChildID = arrayOfBTNodes[nodeid].getRightChildID(); // -1 if not
                exist
137
138             return max(getNodeHeight(leftChildID), getNodeHeight(rightChildID)) + 1;
139
140         }
141
142
143         int getTreeHeight(){
144             return getNodeHeight(0);
145         }
146
147
148         void PreOrderTraversal(int nodeid){
149
150             if (nodeid == -1)
151                 return;
152
153             cout << nodeid << " ";
154
155             PreOrderTraversal(arrayOfBTNodes[nodeid].getLeftChildID());
156             PreOrderTraversal(arrayOfBTNodes[nodeid].getRightChildID());
157
158         }
159
160
161         void PrintPreOrderTraversal(){
162
163             PreOrderTraversal(0);
164             cout << endl;
165
166         }
167
168
169
170 };
171
172
173
174 int main(){
175
176     string filename;
177     cout << "Enter a file name: ";
178     cin >> filename;
179
180     int numNodes;
181     cout << "Enter number of nodes: ";
182     cin >> numNodes;
183
184     BinaryTree binaryTree(numNodes);
185
```

```
186       ifstream fileReader(filename);
187
188       if (!fileReader){
189           cout << "File cannot be opened!! ";
190           return 0;
191       }
192
193       int numCharsPerLine = 10;
194
195       char *line = new char[numCharsPerLine];
196       // '10' is the maximum number of characters per line
197
198       fileReader.getline(line, numCharsPerLine, '\n');
199       // '\n' is the delimiting character to stop reading the line
200
201       while (fileReader){
202
203           char* cptr = strtok(line, ",: ");
204
205           string upstreamNodeToken(cptr);
206           int upstreamNodeID = stoi(upstreamNodeToken);
207
208           cptr = strtok(NULL, ",: ");
209
210           int childIndex = 0; // 0 for left child; 1 for right child
211
212           while (cptr != 0){
213
214               string downstreamNodeToken(cptr);
215               int downstreamNodeID = stoi(downstreamNodeToken);
216
217               if (childIndex == 0 && downstreamNodeID != -1)
218                   binaryTree.setLeftLink(upstreamNodeID, downstreamNodeID);
219
220               if (childIndex == 1 && downstreamNodeID != -1)
221                   binaryTree.setRightLink(upstreamNodeID, downstreamNodeID);
222
223               cptr = strtok(NULL, ",: ");
224               childIndex++;
225           }
226
227           fileReader.getline(line, numCharsPerLine, '\n');
228
229       }
230
231       cout << "PreOrderTraversal: ";
232       binaryTree.PrintPreOrderTraversal();
233
234
235
236       return 0;
237   }
```

```
Enter a file name: binaryTreeFile_1.txt
Enter number of nodes: 10
PreOrderTraversal: 0 1 3 6 2 4 7 9 8 5
```