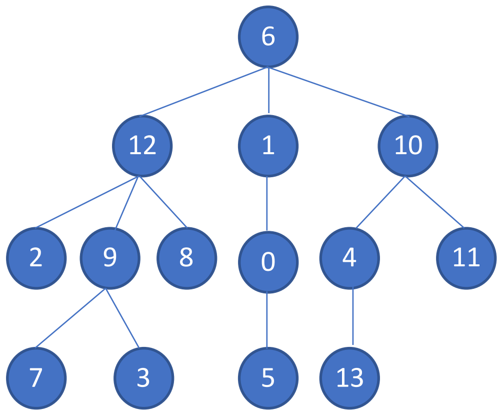
CS/DSA 5005 – Computing Structures – Project 2 – Summer 2019
Due July 31st, 2019, 11:59 PM

Objective: In this project you will learn to manipulate an arbitrary tree data structure.

Description: A tree is a data structure has a root node and each node in the tree children. Nodes that do not have children are leaf nodes. Below is an example of a tree data structure.



The following are some points about the above tree.

- The root of the above tree is 6. It has three children 12, 1, and 10.
- The height of the above tree is 4.
- Node 6 is at level 1 and its children are at level 2.
- Node 8 is at level 3.
- The parent of node 0 is 1 and the parent of node 13 is 4.
- Nodes 12, 1, and 10 are siblings and similarly 3 and 7 are siblings.
- Node 2 is a leaf node. There are other leaf nodes as well.
- The least common ancestor (LCA) for nodes 3 and 8 is 12 (denoted LCA (3,8) = 12). Similarly, LCA (5, 13) = 6. Also, LCA (7, 12) = 12. LCA (1,1) = 1.
- The nodes in level 3 are 2, 9, 8, 0, 4, and 11.

The above tree can be represented by the parent array as below where the ith location stores the parent of node i. The parent of the root is indicated by a -1.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
5	6	12	9	10	0	-1	9	12	12	6	10	6	4

Sample input file: The first line of the input is the total number of nodes in the tree. After that each line of the input will have a node number followed by number of children and the IDs of the children. Here is the example of the input for the above tree.

```
14
12 3 2 9 8
1 1 0
10 2 4 11
9 2 7 3
6 3 12 1 10
0 1 5
4 1 13
```

Implementation: You are required to design and create a class name it *Tree*. In the protected field you must define the parent array (whose size is defined at the time of creation) and any other fields. You also must implement the following methods:

1. Default constructor
2. Non-default constructor
3. Copy constructor
4. Destructor
5. ostream operator (which prints the parent array)
6. LCA (least common ancestor given two node numbers)
7. Parent(i) – Get the parent of node i
8. Children(i) – Get the children of node i
9. Siblings (i) – Get the siblings of node i
10. Root() – get the root of the tree
11. setRoot(rootNode)
12. setParent(node, parent)
13. nodesAtLevel(i) – Give the nodes at level i
14. Level(i) – Give the level of node i
15. height() – Give the height of the tree
16. Preorder () – Give the preorder traversal of the tree
17. Postorder () – Give the postorder traversal of the tree
18. Inorder () – Give the inorder traversal of the tree

Processing: Your main program will be as follows (you are responsible for syntax checks). We may modify this a little bit to add other methods:

```
int main () {
        Tree* myTree;
        int numNodes, node, parent, numChildren. child;

        cin >> numNodes;
        myTree = new Tree (numNodes);
        while (!cin.eof()) {
                cin >> parent  >> numChildren;
                for (int i=0; i < numChildren; i++) {
                   cin >> child;
                   (*myTree).setParent (child, parent);
                }
        }
        cout << "The tree that we just read is …… " << endl;
        cout << *myTree << endl; //overloading the ostream operator

        cout << "Printing other methods …." << endl;

        Tree* newTree = new Tree (*myTree); //copy constructor
        cout << "The tree that we just copied is …… " << endl;
        cout << *newTree << endl;

        cout << "The root of the tree is: " << (*myTree).Root() << endl;

        cout << "The least common ancestor of nodes 3 and 8 is: " << (*newTree).LCA (3,8) << endl;
        cout << "The least common ancestor of nodes 3 and 8 is: " << (*newTree).LCA (13,8) << endl;
        cout << "The least common ancestor of nodes 3 and 8 is: " << (*myTree).LCA (13,11) << endl;
        cout << “The children of node 12 is: “;
        (*myTree).Children (12)
        cout << endl;

        cout << “The children of node 10 is: “;
        (*myTree).Children (10);
        cout << endl;

        cout << “The siblings of node 3 is: “;
        (*myTree).Siblings (3)
        cout << endl;

        cout << “The siblings of node 12 is: “;
        (*myTree).Siblings (12)
        cout << endl;

        cout << “The nodes at level 3 are: “;
        (*myTree).nodesAtLevel (3);
        cout << endl;

        cout << “The height of the tree is: “ << (*myTree).height() << endl;

        cout << “The level of node 3 in the tree is: “ << (*myTree).level(3) << endl;
        cout << “The level of node 3 in the tree is: “ << (*myTree).level(12) << endl;

        cout << “The preorder traversal of the tree is: “;
        (*myTree).Preorder();
        cout << endl;

        cout << “The postorder traversal of the tree is: “;
        (*myTree).Postorder();
        cout << endl;

        cout << “The inorder traversal of the tree is: “;
        (*myTree).Inorder();
        cout << endl;

        delete *myTree;
        delete *newTree;
}
```

Constraints:

- In this project, the only header you will use is #include <iostream>
- None of the projects is a group project. Consulting with other members of this class on programming projects is strictly not allowed and plagiarism charges will be imposed on students who do not follow this.