

Lab 09

Data Structures

BS DS Fall 2024 Morning/Afternoon

Objective:

Learning and Implementing the Representation of Binary Trees using Link base structure and implementing some basic operations on them.

ADT for Linked Representation

Note: you can add some utility functions as per required for the completion of public functions.

```
template<class T>
class Node
{
public:
    T info;
    Node<T> *left;
    Node<T> *right;

    Node()
    {
        left=0;
        right=0;
    }

    Node(int v)
    {
        info=v;
        left=0;
        right=0;
    }
};
```

```
template <class T>
class BinaryTree
{
private:
    Node<T> *root;
public:

};
```

Add the following public methods in the Binary Tree Class;

1. `BinaryTree();` // initializes root to 0.
2. `void setRoot(T);`
3. `T getRoot();`
4. `void setLeftChild(T parent, T child);`
5. `void setRightChild(T parent, T child);`
6. `T getParent(T node);`
7. `void remove(T node);` //removes the given node and all its descendants from tree.
8. `int isInternalNode(T node);` // Internal Node is one which has degree greater than zero
9. `int isExternalNode(T node);` // External Node is one which has degree equal to zero
10. `T findNodeSiblings(T node);` // return the sibling of given node
11. `void preOrder();` // do the VLR of tree.
12. `void postOrder();` // do the LRV of tree.
13. `void inOrder();` // do the LVR of tree.
14. `void levelOrder();` // do the level order traversal of tree.
15. `void displayDescendents(T node);` //display descendants of the given node
16. `void heightOfTree();` //returns the height (actual height) of tree.
17. `void displayParenthesizedView();` // display the tree in Parenthesize form.
18. `~BinaryTree();`
19. `BinaryTree<T> operator = (BinaryTree<T> &);`
20. `BinaryTree(BinaryTree<T> &);` // produces deep copy
21. `BinaryTree<T> getMirrorImage ();` // return the mirror image of the *this object.

Return a new tree which looks like the mirror image of the given tree. For example, given



You should return

