Name: - Atish Kumar

Roll No: - 120CS0173

Lab Sheet: - 08

Q2.Write a program for Linked-list implementation of a complete binary tree. The program must have the following functionalities.
a) Insert(): inserts a new ITEM to the complete binary tree. The items are of integer type.
b) Height(): returns height of a node recursively. Height (N) = MAX(Height(L), Height(R)) + 1. Here, L and R respectively represent the Left child and Right child of node N. Height of a leaf node is 0.
c) Preorder(): returns the preorder traversal sequence of the binary tree. Use recursive implementation.
d) Postorder(): returns the postorder traversal sequence of the binary tree. Use recursive implementation

Program:-

```
#include <iostream>
#include <algorithm>
using namespace std;

struct Node
{
 int data;
 struct Node *right;
 struct Node *left;
 Node(int _data):data(_data),left(NULL),right(NULL){}
};

class BinaryList
{
        public:

                //a)Function for insertion new node to the Binary list
                struct Node* Insert(int value, struct Node *p) {
                        if(p==NULL)
                        p=new Node(value);
                        else if(value<p->data)
                        p->left=Insert(value,p->left);
                        else if(value>p->data)
                        p->right=Insert(value,p->right);
                        return p;
                }
```

```
		//b)Function to count Height of the Binary list
		int Height(struct Node *p) {
			if(p==NULL)return 0;
			else {
				int lhght=Height(p->left);
				int rhght=Height(p->right);
				return max(lhght,rhght)+1;
			}
		}

		//c)Function returns the preoder traversial sequence of the Binary
list
		void Preorder(struct Node *p) {
			if(p==NULL)return;
			cout << " --> " << p->data;
			Preorder(p->left);
			Preorder(p->right);
		}

		//d)Function returns the postoder traversial sequence of the Binary
list
		void Postorder(struct Node *p) {
			if(p==NULL)return;
			Postorder(p->left);
			Postorder(p->right);
			cout<< " --> " << p->data;
		}
};

int main()
{
	Node *root=new Node(9);
	BinaryList bl;
	bl.Insert(1,root);
	bl.Insert(120,root);
	bl.Insert(35,root);
	bl.Insert(42,root);
	bl.Insert(15,root);
	bl.Insert(20,root);
	bl.Insert(30,root);
	cout<<"\nHeight= "<<bl.Height(root);
	cout<<"\nPreorder traversial sequence: ";
	bl.Preorder(root);
	cout<<"\nPostorder traversial sequence: ";
```

```
        bl.Postorder(root);
}
```

Output:-