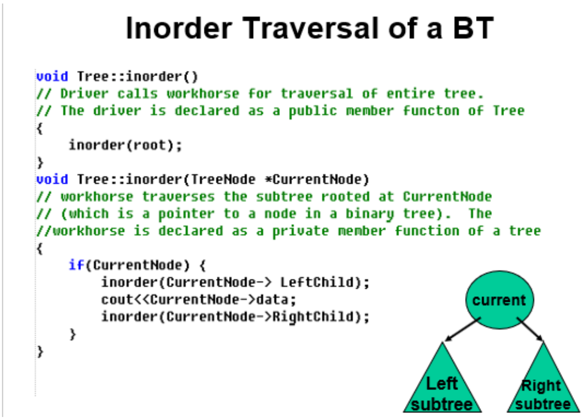
Data Structure Program Assignment #5
(Due: PM: 9:00, June 10, 2022)

Tree Traversal (lecture 5 Tree)

Instructor: Jiann-Jone Chen

- **Introduction**

In lecture5, we introduced a tree structure that can store data in a parent-child approach, such that the data can be visited in different orders. For example, we can traverse a tree by in-order, pre-order, and post-order for different applications. In general, we use the tree-node and pointers to perform recursive function execution to achieve the target. For example, the in-order traversal program under a binary tree structure can be designed in the following way. Pre-order and post-order can be carried out in a similar way. See lecture notes p25, p27, and p28.



Inorder Traversal of a BT

```
void Tree::inorder()
// Driver calls workhorse for traversal of entire tree.
// The driver is declared as a public member functon of Tree
{
    inorder(root);
}
void Tree::inorder(TreeNode *CurrentNode)
// workhorse traverses the subtree rooted at CurrentNode
// (which is a pointer to a node in a binary tree).  The
//workhorse is declared as a private member function of a tree
{
    if(CurrentNode) {
        inorder(CurrentNode-> LeftChild);
        cout<<CurrentNode->data;
        inorder(CurrentNode->RightChild);
    }
}
```

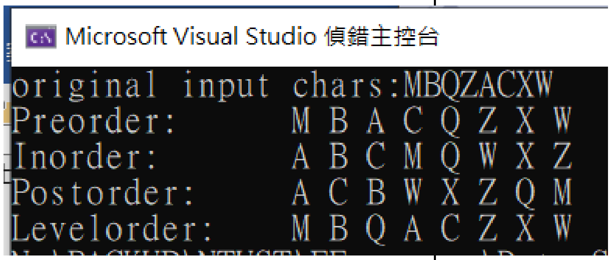
In this homework, you are asked to implement a tree traversal program based on a template TreeNode class and a template Tree class, as shown below:

```
template <class T> class TreeNode {
     friend class Tree<T>;
private:
     TreeNode<T> *LeftChild;
     T  data;
     TreeNode<T> *RightChild;
};
```

```
template <class T> class Tree{
public:
     Tree(TreeNode<T> data, TreeNode<T>* left, TreeNode<T>* right) {
          root = &data;
          root->LeftChild = left;
          root->RightChild = right;
     };

     Tree(){root=NULL;};
     ~Tree(){};
     bool isempty(){return(root==NULL); };
     void Insert(T);
     void Inorder();  // Driver
     void Inorder(TreeNode<T>*); // Workhorse
     void Preorder() {
          Preorder(root);
          cout << endl;
     };
     void Preorder(TreeNode<T>*);
     void Postorder() {
          Postorder(root);
          cout << endl;
     };
     void Postorder(TreeNode<T>*);
     void Levelorder();
private:
     TreeNode<T> *root;
};
```

If the main.cpp is the one shown below the output would be

```
/* Binary Tree Traversal - Preorder, Inorder, Postorder */
#include<iostream>
#include<iomanip>
#include "tree.h"       // design the tree class by yourself
#include <queue>
using namespace std;

int main() {
     /* construct a binary search tree
        according to the input chars

                 M
                /   \
               B     Q
              / \     \
             A   C     Z
                      /
                     X
                    /
                   W
     */
     Tree<char> bt;
     Tree<char> bt2;
     char input[30] = "MBQZACXW";
     for (int i = 0; i < 8; i++) bt.Insert(input[i]);
          cout <<left << setw(15) <<"original input chars:" << input << endl;
          cout<<left<<setw(15)<<"Preorder: ";    bt.Preorder();
          cout<<left<<setw(15)<<"Inorder: ";    bt.Inorder();
          cout<<left<<setw(15)<<"Postorder:";    bt.Postorder();
          cout<<left<<setw(15)<<"Levelorder:";   bt.Levelorder();
     return 0;
}
```



```
original input chars:MBQZACXW
Preorder:      M B A C Q Z X W
Inorder:       A B C M Q W X Z
Postorder:     A C B W X Z Q M
Levelorder:    M B Q A C Z X W
```

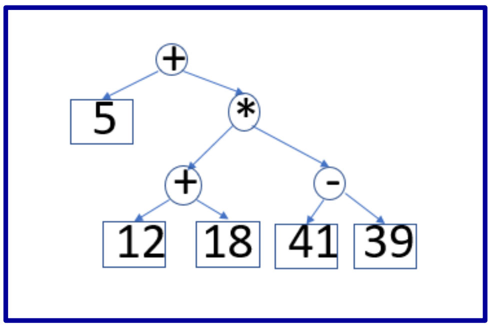
1. What to do (70%)

- A demo project is provided for you to start programming easily, and a demo exe file is provided for you to check the correctness of your program.
- Design the program for a binary search tree construction function.
- Programming for the four binary tree traversal functions, i.e., in-order, pre-order, post-order, and level order traversal functions.

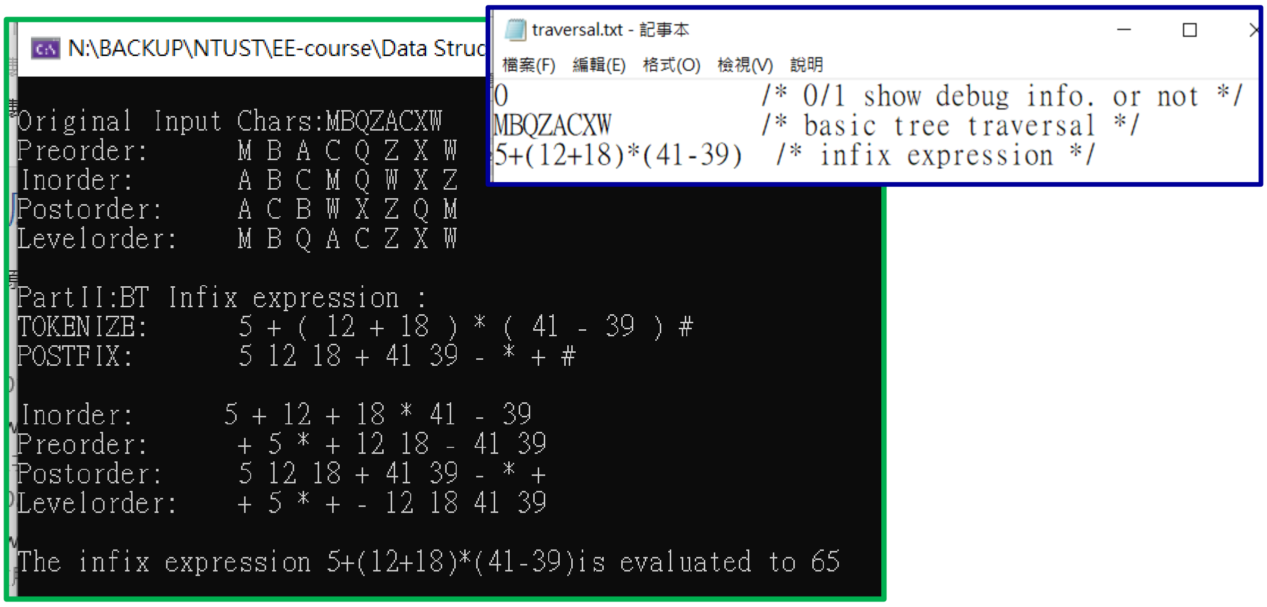
```
/* Binary Tree Traversal - Preorder, Inorder, Postorder */
#include<iostream>
#include<iomanip>
#include "Tree.h"       // design the tree class by yourself
#include "mysetting.h"  // don't care

using namespace std;
void init();
Tree<char> bt;
Tree<Token> bt2;
string infix, input;
int main() {
     char cont;
     init();
     cout << "Part I: Basic Tree Travrsal\n";
          for (int i = 0; i < 8; i++) bt.Insert(input[i]);
          cout << left << setw(15) << "\nOriginal Input Chars:" << input << endl;
          cout << left << setw(15) << "Preorder: ";    bt.Preorder();
          cout << left << setw(15) << "Inorder: ";   bt.Inorder();
          cout << left << setw(15) << "Postorder:";    bt.Postorder();
          cout << left << setw(15) << "Levelorder:";   bt.Levelorder();

     // ************************************************************
          cout << "\nPartII:BT Infix expression :\n";
          // Tree<Token> bt2;
          // string infix = "2+3*(4-50)/3-(4-30)";
          bt2.Infix(infix);
          cout << left << setw(15) << "\nInorder: ";       bt2.Inorder();
          cout << left << setw(15) << "Preorder: ";         bt2.Preorder();
          cout << left << setw(15) << "Postorder:";         bt2.Postorder();
          cout << left << setw(15) << "Levelorder:";        bt2.Levelorder();
          cout << "\nThe infix expression " << infix
               << "is evaluated to " << bt2.Evaluate() << endl;
          return 0;
}
void init() {  // read the infix from the file infix.txt
     ifstream inFile("traversal.txt", ios::in); // input file
     if (!inFile) { cerr << "Cannot open input file" << endl; exit(1); }
     inFile >> debug;    inFile.clear();  inFile.ignore(100, '\n');
     inFile >> input;    inFile.clear();  inFile.ignore(100, '\n');
     inFile >> infix;    inFile.clear();  inFile.ignore(100, '\n');
};
```

2. Applications (30%)

- In our previous program design homework, we designed infix to postfix function. In this homework, you can write a program to construct an infix expression binary tree. For example, you can design a member function for the BinaryTree (bt) class and this member function, bt.Infix("5+(12+18)*(41-39)"), will construct the binary tree as shown on the right.



- Based on this infix expression binary tree, you can design another member function bt.evaluate() that can evaluate the value of this infix expression. i.e., cout << bt.**Evaluate()** will output **65.**
- You had to design this program based on the provided program template. In short, do the homework by yourself and not just download another working program available on websites.



N:\BACKUP\NTUST\EE-course\Data Struc
traversal.txt - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
0                   /* 0/1 show debug info. or not */
MBQZACXW            /* basic tree traversal */
5+(12+18)*(41-39)  /* infix expression */
```

```
Original Input Chars:MBQZACXW
Preorder:      M B A C Q Z X W
Inorder:       A B C M Q W X Z
Postorder:     A C B W X Z Q M
Levelorder:    M B Q A C Z X W

PartII:BT Infix expression :
TOKENIZE:      5 + ( 12 + 18 ) * ( 41 - 39 ) #
POSTFIX:       5 12 18 + 41 39 - * + #

Inorder:     5 + 12 + 18 * 41 - 39
Preorder:      + 5 * + 12 18 - 41 39
Postorder:     5 12 18 + 41 39 - * +
Levelorder:    + 5 * + - 12 18 41 39

The infix expression 5+(12+18)*(41-39)is evaluated to 65
```

Report:

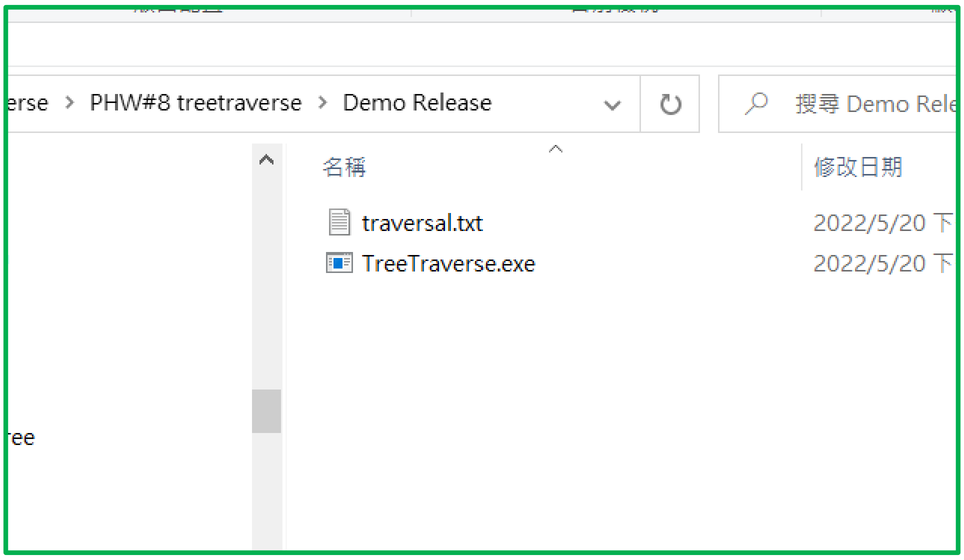
Write a report to describe how do you solve this problem and design the program.

Program plagiarism will give you a big egg. "0".

3. Bonus (30%)

- In PHW#7, you designed an available linked list of nodes for the programs to allocate (**new**) and de-allocate (**delete**) memory space for (polynomial) nodes. It helps you better manage and use memory space during program execution.

- In this PHW#8, you may need to design general node and treenode data structures to implement the required tree traversal functions. You can try to design an available list for each node type and verify that all allocated node memory can be 100% recycled to the av_list, as we have shown in the PHW#7 video.
- An executable program in the directory DemoRelease is provided for your reference in programming.



- You must have to use your own Stack.h, Queue.h and Tree.h to design the program instead of using the STL of Visual C.
- See the video for details.
https://youtu.be/JcGYvoGyUaU