第一大題 I

```
//a
struct ListNode
{
	ListNode *previousPtr;
	int data;
	ListNode *nextPtr;
};
//b
ListNode *getNewNode(int value )
{
	ListNode *newPtr = new (ListNode);
	if ( newPtr != NULL )
	{
		newPtr->data = value;
		newPtr->nextPtr = NULL;
		newPtr->previousPtr = NULL;
	}
	return newPtr;
}
//c
void InsertatFront(int val)
{
	ListNode *newPtr=getNewNode(val);

	newPtr->nextPtr=(*FirstPtr);
	newPtr->previousPtr=NULL;
	(*FirstPtr)=newPtr;
}
//d
void InsertatBack(int val)
{
	ListNode *newPtr=getNewNode(val);
	ListNode *prePtr,*currentPtr;

	prePtr = NULL;
	currentPtr = *FirstPtr;
```

```
        while(currentPtr!=NULL)
        {
                prePtr=currentPtr;
                currentPtr=currentPtr->nextPtr;
        }
        prePtr->nextPtr = newPtr;
        newPtr->previousPtr = prePtr;
}
```

第二大題 II

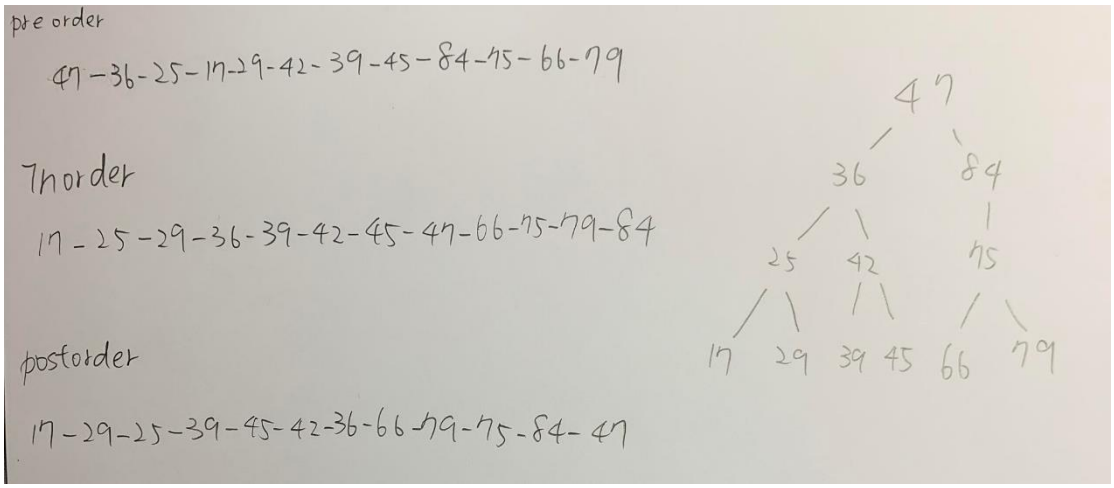
```
//a
#include<bits/stdc++.h>
using namespace std;

class TreeNode {
        public:
                TreeNode *left = NULL;
                TreeNode *right = NULL;
                TreeNode *parent = NULL;
                int val;
};
class Tree {
        public:
                TreeNode *root = NULL;
                void Insert(int val) {
                        TreeNode *newnode = new TreeNode();
                        TreeNode *cur = root;
                        newnode->val = val;
                        if(root == NULL) {
                                root = newnode;
                        }
                        else {
                                while(true) {
                                        if(val < cur->val) {
                                                if(cur->left == NULL) {
                                                        newnode->parent = cur;
                                                        cur->left = newnode;
                                                        break;
                                                }
```

```
                        else {
                                cur = cur->left;
                        }
                }
                else {
                        if(cur->right == NULL) {
                                newnode->parent = cur;
                                cur->right = newnode;
                                break;
                        }
                        else {
                                cur = cur->right;
                        }
                }
        }
    }
}
void vlr(TreeNode *t) {
    TreeNode *cur = t;
    cout << cur->val << " ";
    if(cur->left!=NULL)
        vlr(cur->left);
    if(cur->right!=NULL)
        vlr(cur->right);
    return;
}
void lvr(TreeNode *t) {
    TreeNode *cur = t;
    if(cur->left!=NULL)
        lvr(cur->left);
    cout << cur->val << " ";
    if(cur->right!=NULL)
        lvr(cur->right);
    return;
}
void lrv(TreeNode *t) {
    TreeNode *cur = t;
    if(cur->left!=NULL)
```

```
                lrv(cur->left);
            if(cur->right!=NULL)
                lrv(cur->right);
            cout << cur->val << " ";
            return;
        }
        void decending(TreeNode *t) {
            TreeNode *cur = t;
            if(cur->right!=NULL)
                decending(cur->right);
        cout << cur->val << " ";
            if(cur->left!=NULL)
                decending(cur->left);
            return;
        }
};

int main() {
    Tree t;
    t.Insert(47);
    t.Insert(36);
    t.Insert(84);
    t.Insert(25);
    t.Insert(42);
    t.Insert(75);
    t.Insert(29);
    t.Insert(17);
    t.Insert(39);
    t.Insert(45);
    t.Insert(66);
    t.Insert(79);
    cout << "pre-order ";
    t.vlr(t.root);
    cout << endl;
    cout << "in-order ";
    t.lvr(t.root);
    cout << endl;
    cout << "post-order ";
```

```
        t.lrv(t.root);
        cout << endl;
        cout << "decending order ";
        t.decending(t.root);
}
```

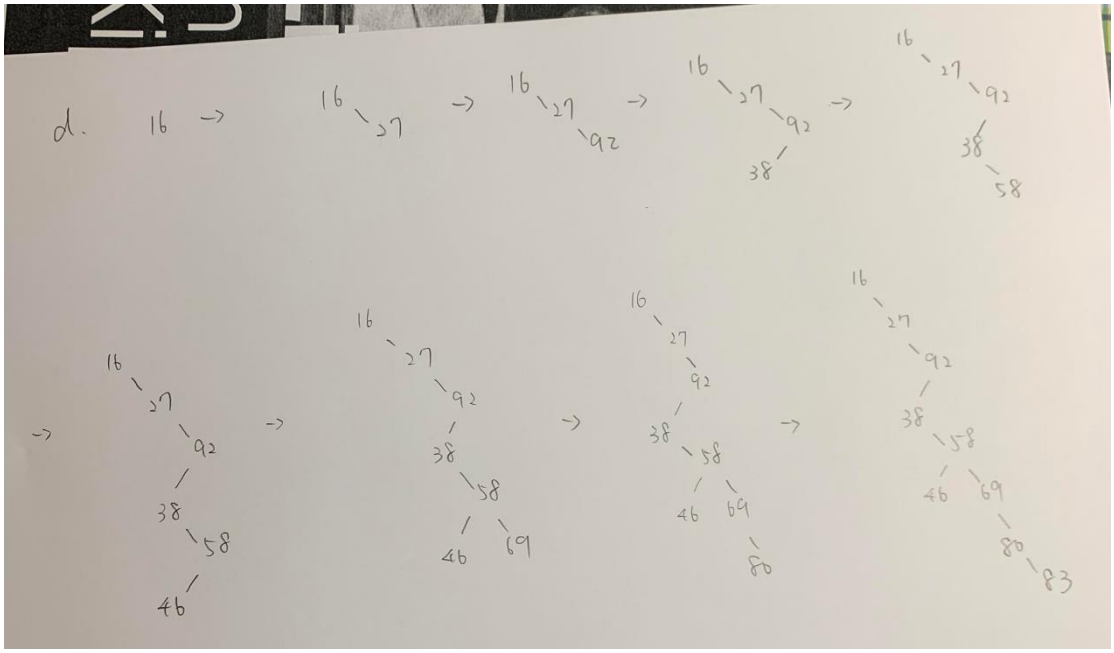
//b



//c

利用遞迴先判斷右邊有沒有 node 有的話就遞迴進去,接著輸出數字,再判斷左邊有沒有 node 有的話就遞迴進去,這樣就可以按照遞減的順序輸出,程式碼在第二大題的一開始

//d



第三大題 III

//a

class Vehicle

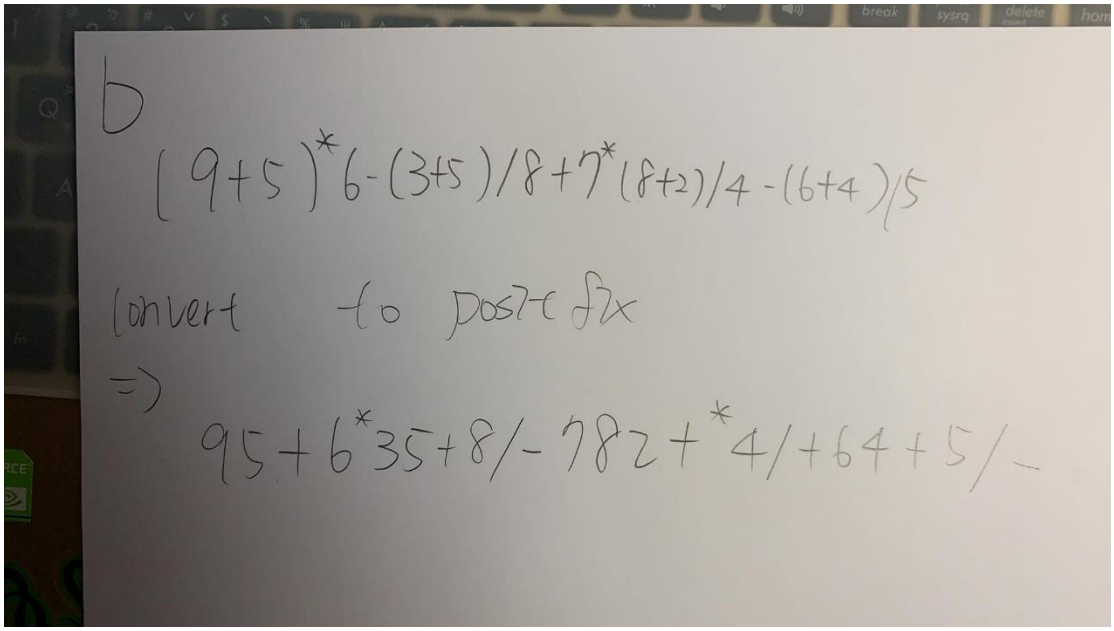
```
{
public:
	virtual sound()
	{
		cout<<"making a sound"<<endl;
	}
};
class Trains:public Vehicle
{
public:
	sound()
	{
		cout<<"Bun Bun"<<endl;
	}
};
class Molcar:public Vehicle
{
public:
	sound()
	{
		cout<<"Gi Gi"<<endl;
	}
};
class Bicycle:public Vehicle
{
public:
	sound()
	{
		cout<<"Lin Lin"<<endl;
	}
};
//b
	Vehicle v;
	Trains t;
	Molcar m;
	Bicycle b;

	vector <Vehicle *> Vehicles;
```

```
	Vehicles.push_back(&t);
	Vehicles.push_back(&m);
	Vehicles.push_back(&b);

	vector<Vehicle *>::iterator iter;

	for(iter = Vehicles.begin() ; iter != Vehicles.end() ; iter++ )
		{
			(*iter)->sound();
		}
```

第四大題

```
//a
		void inOrder(TreeNode *treePtr) {
			TreeNode *cur = treePtr;
			if(cur->data == '+' || cur->data == '-' || cur->data == '*' || cur->data
== '/')
				cout << '(';
			if(cur->left!=NULL)
				inOrder(cur->left);
			cout << cur->val;
			if(cur->right!=NULL)
				inOrder(cur->right);
			if(cur->data == '+' || cur->data == '-' || cur->data == '*' || cur->data
== '/')
				cout << ')';
			return;
		}
//b
```

轉換成 postfix



Expression tree