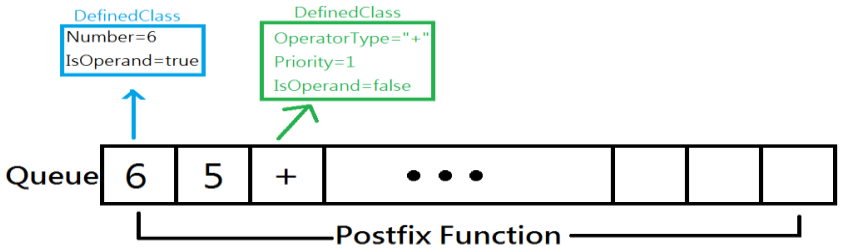
題目：

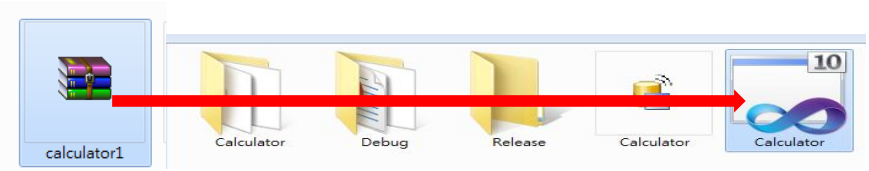
PART 1. 給予一運算子，實作此運算子存入堆疊內的所需經過的判斷過程與必要的運算過程。

PART 2. 給予一儲存後序式(Postfix)之佇列(Queue<DefinedClass>)，實作一將此後序式計算並回傳最終結果之副程式。

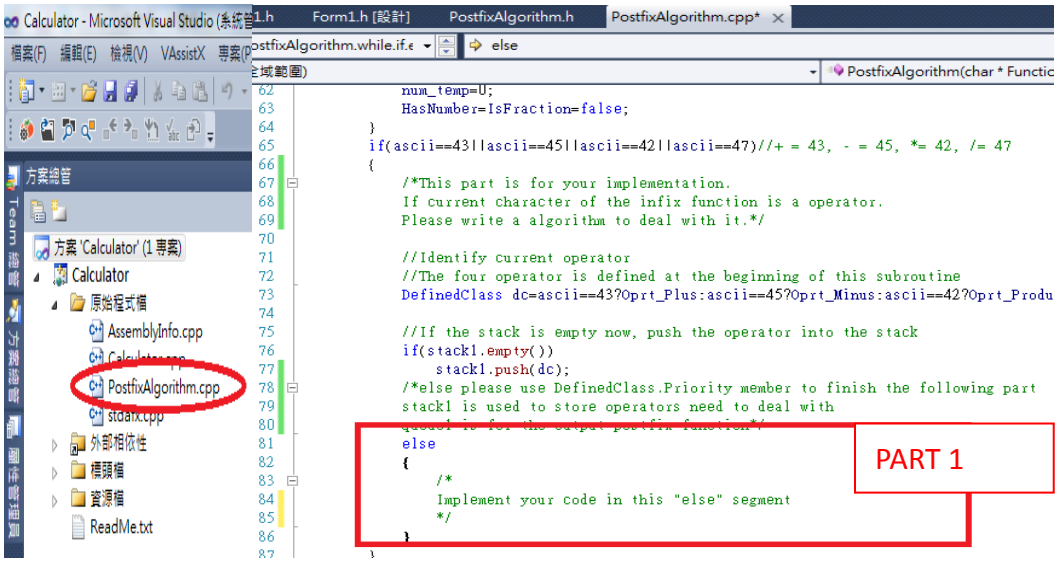


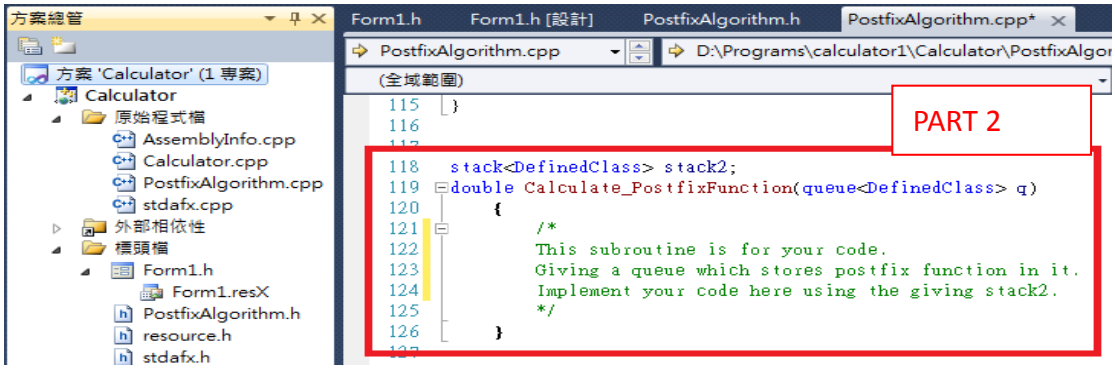
附錄檔案說明：

解開附錄之壓縮檔，以 Visual Studio 開啟 Calculator.sln

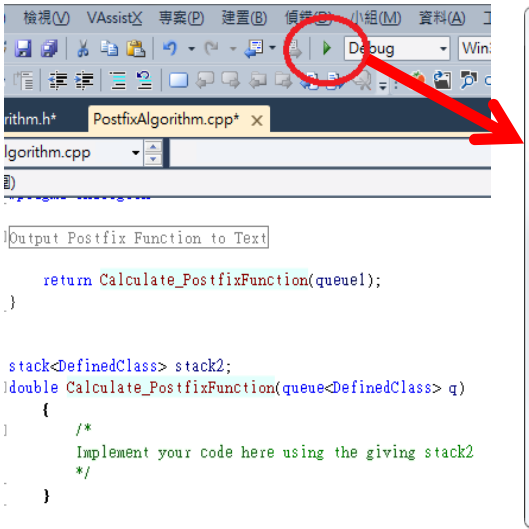


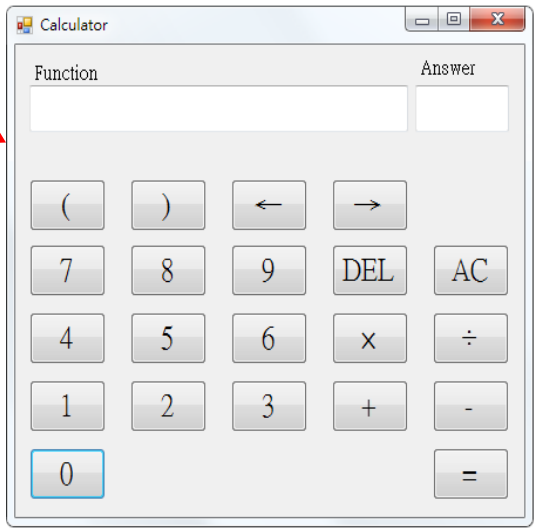
於方案總管中開啟 PostfixAlgorithm.cpp。



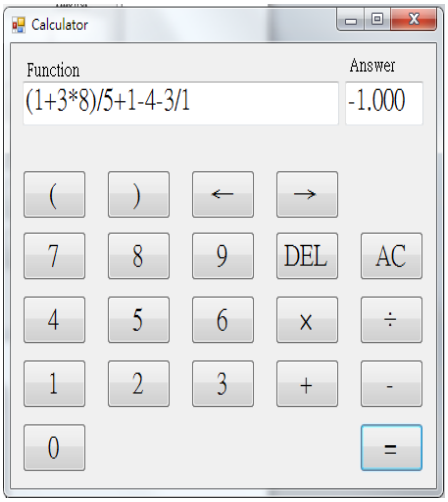


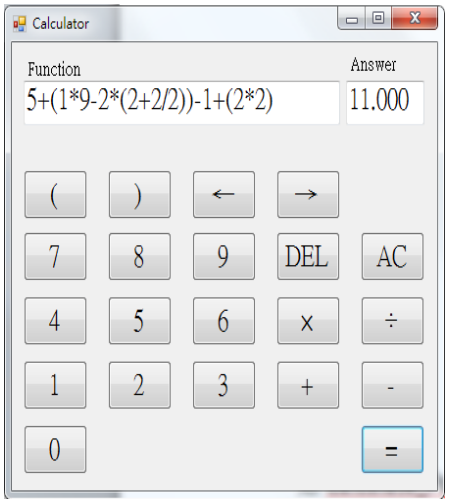
結果驗證：完成程式碼之後，按下圖中的執行按鈕檢視執行結果





完成輸入算式之後按下”=”





DefinedClass 宣告方法：

DefinedClass Operator_Plus("+",1);　　//宣告一個加號，優先度為1

DefinedClass Operand_Nine(9);　　//宣告一個數字9

可用函式：

Queue.push();　Queue.pop();　Queue.front();　Queue.empty();　Queue.back();

Stack.push();　Stack.pop();　Stack.top();　Stack.empty();　Stack.size();

DefinedClass.IsOperand;　　DefinedClass. GetPriority();

DefinedClass.GetNumber();　　DefinedClass.GetOperatorType();

DefinedClass 說明(PostfixAlgorithm.h)

```
5  class DefinedClass;
6  double PostfixAlgorithm(char* c);
7  double Calculate_PostfixFunction(queue<DefinedClass>);
```

1. DefindClass：為自訂之資料結構，成員繼承自 BASIS，用以儲存算式中的運算元或是運算子。
2. PostfixAlgorithm(char* c)：程式進行中序(infix)轉後序(postfix)並計算結果所呼叫的副程式，輸入為中序式算式陣列之第一個字元的指標，回傳值為計算結果。這段副程式結尾將轉換完成的後序式傳給 Calculate_PostfixFunction 進行計算，最後將結果回傳。
3. Calculate_PostfixFunction(queue<DefinedClass>)：本次作業需實作的部分，給予一個儲存後序式的佇列 q(資料型別為 queue<DefinedClass>)，取出佇列的內容並利用一個 stack 進行後序式的計算，最後會將後序式的計算結果回傳。

```
class BASIS
{
private:
    string OperatorType;
    int Priority;
    double Number;

public:
    BASIS(string _OperatorType,int _Priority) { }

    BASIS(double _Number) { }

    bool IsOperand;
};
```

1. OperatorType：儲存運算子(+、-、*、/)
2. Priority：儲存運算子優先度之變數
3. Number：若這個類別儲存的是運算元，則此變數用來儲存運算元的數值
4. IsOperand：值為 true 時表示儲存之內容為運算元，反之則為運算子。
5. 建構子
   A. BASIS(_OperatorType, _Priority)：宣告一個用來儲存運算子的結構。
   B. BASIS(_Number)：宣告一個用來儲存運算元(數字)的結構。

```
class DefinedClass:public BASIS{
private:
    string OperatorType;
    int Priority;
    double Number;

public:
    DefinedClass(string _OperatorType,int _Priority)
        :BASIS(_OperatorType,_Priority)
    {
        OperatorType=_OperatorType;
        Priority=_Priority;
        IsOperand=false;
        Number=0;
    }
    DefinedClass(double _Number):BASIS(_Number)
    {
        Number=_Number;
        IsOperand=true;
        OperatorType="";
        Priority=0;
    }

    string GetOperatorType()
    {
        if(!IsOperand) return OperatorType;
        else return "Not a Operator.";
    }
    int GetPriority()
    {
        if(!IsOperand) return Priority;
        else return -1;
    }
    double GetNumber()
    {
        if(IsOperand) return Number;
        else return NULL;
    }
    ~DefinedClass(){}
};
```

1. GetOperatorType：當儲存的是運算子時，回傳運算子之種類。
2. GetPriority：回傳運算子之優先度。
3. GetNumber：當儲存的是運算元時，回傳數值。
4. 當宣告類別儲存的是運算子時使用第一個建構式，會將 Number 設為 0。若儲存的是運算元則使用第二個建構式，則將 Operator 設置為空值。