資料結構與進階程式設計（106-2）
程式作業三

國立臺灣大學資訊管理學系

繳交作業時，請至 PDOGS（`http://pdogs.ntu.im/judge/`）為第一題上傳一份 C++ 原始碼（以複製貼上原始碼的方式上傳），為第二題上傳一份 PDF 檔，為第三題做同儕互評。每位學生都要上傳自己寫的解答。不接受紙本繳交；不接受遲交。請以英文或中文作答。

這份作業的前兩題的截止時間是 ***2018 年 5 月 31 日下午一點***；第三題的截止時間是 ***2018 年 6 月 4 日下午一點***。在你開始前，請閱讀課本的第 6 章[1] 為這份作業設計測試資料並且提供解答的是楊其恆。

第一題

（60 分）我們上課時曾經介紹過代數的運算式有 prefix, infix 與 postfix 三種表示方式。我們習慣的 infix expression 需要 precedence rules、associativity rules 及 parentheses 來解決代數運算時的歧異性 (ambiguity)，我們較不習慣的 postfix expression 則能清楚陳述運算元與運算子的關係。

你上傳的原始碼裡應該包含什麼

你的.cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你***不可以***使用上課沒有教過的方法，也***不可以***使用 `<vector>` 裡面的東西。

除此之外，你應該***實作出 ADT stack***，並用所教的方法來進行相關轉換及運算。需注意，ADT stack 需要以 **link-based** 方式進行實作，不可採用 array-based 方法。若你沒有這麼做，這份作業的第一題會***以 0 分計算***。

題目敘述

在本程式作業，我們需要撰寫一個 infix to postfix expression converter。對於一個給定的 infix expression，我們除了要輸出轉換後的 postfix expression，還須計算該 postfix expression 的運算結果。課程投影片已清楚說明如何運用將 Stack 來將 infix expression 轉成 postfix expression 及如何計算一個 postfix expression，然而與上課內容不同的是，在本次作業中我們將把運算式中的常數以多項式替代。在我們的程式中，需要對多項式進行多項式的加、減、乘、除的四則運算、取餘數、取次方等運算。乘法、除法、取餘數、取次方請依照以下規則進行運算。

- 乘法 (*)：請將兩個多項式相乘，例如 $(x^2+x+3)*(x-2)$ 的結果應該是 x^3-x^2+x-6。
- 除法 (/)：請做兩個多項式的除法，並以其商式為結果，即在 $f(x)/g(x)$ 之中，$f(x)=q(x)g(x)+r(x)$, $deg(r(x))<deg(g(x))$，結果應該是 $q(x)$，餘式 $r(x)$ 不需考慮。計算完後，$q(x)$ 之中的係數若有小數，請無條件捨去至整數。例如 $(3x^2+x+3)/(-2x-2)$ 的結果應該是 $-x+1$，因為將 $-1.5x+1$ 中的 -1.5 無條件捨去為 -1。

[1]課本是 Carrano and Henry 著的 *Data Abstraction and Problem Solving with C++: Walls and Mirrors* 第六版。

- 取餘數 (%)：請將兩個多項式相除後，取其餘式 $r(x)$。同樣的，如果出現小數，請無條件捨去至整數。例如 $(x^2+x+3)\%(x-2)$ 的結果應該是 9。
- 取次方 (^)：在 a ^ b 的運算式之中，請計算 a 的 b 次方，其中 a 可能是常數或一次以上多項式，而 b 是一個大於 0 的整數。例如 (x^2+x+3)^3 的結果應該是 $x^6+3x^5+12x^4+19x^3+36x^2+27x+27$

operator 之間的運算優先順序為：取次方 > 取餘數、乘法、除法 > 加法、減法。

輸入輸出格式

系統會提供一共 30 組測試資料，每組測試資料裝在一個檔案裡。每組測試資料只有包含一行沒有任何空白的 infix expression，可能包含的運算子包含了加法 (+)、減法 (-)、乘法 (*)、除法 (/)、取餘數 (%)、左括號 (()、右括號 ())、取次方 (^)。值得注意的是，取次方的運算除了可用來表示多項式的次數，也可以是對括號中的一整串 expression 取次方。

在 expression 之中所有的常數皆為整數，且不會出現 unary operator。所有的多項式之係數皆會以相乘的方式與變數連接，且若係數為 1，就不輸出係數，例如 $3x^5+x^2-9$ 在輸入資料中的格式會是 $3*x$^5 + x^2 − 9。請注意輸入的多項式並不一定會依照降冪排列。

讀入資料後，請將 infix expression 依照上課所提的規則，轉換為對應的 postifx expression。第一行請輸出對應的 postfix expression，任二個數字或 operator 之間請以空白隔開。第二行請輸出此運算式的運算結果，並將該結果以降冪排列，不包含任何空白字元、多項式的係數與 x 之間不需要輸出乘法符號 (*)，且若係數為 1，就不輸出係數。舉例來說，如果輸入是

```
12*x^2+(((5*x^2-3*x^3+x+2)/(2*x^2-2))^2)%(x^2+1)
```

則輸出應該是

```
12 x 2 ^ * 5 x 2 ^ * 3 x 3 ^ * - x + 2 + 2 x 2 ^ * 2 - / 2 ^ x 2 ^ 1 + % +
12x^2-4x+3
```

請注意(5*x^2-3*x^3+x+2)/(2x^2-2) 部分的計算結果是 $-x+2$，因為商式 $-1.5x+2.5$ 經過無條件捨去後為 $-1x+2$。

本題中的運算式長度不限，但你可以假設其長度可以被存在 `int` 而不會溢位。而所有的常數或係數運算過程皆可以被完整儲存在 `double` 之中。

評分原則

這一題的分數都根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。測試資料的複雜度資訊如下：

- 前 10 筆測試資料中只會有加與減的運算。
- 第 11 到 15 筆測試資料中還會包含乘法。
- 第 16 到 20 筆測試資料中會包含加減乘除的四則運算。
- 第 21 到 25 筆測試資料中還會包含取餘數的運算

- 第 26 到 30 筆測試資料中會包含所有運算。

請注意第 1 到 25 筆的測試資料仍然會有二次以上多項式出現，只是不包含對 expression 取次方。

第二題

（20 分）請為你在第一題設計的各個 class 寫一份給開發者的說明文件，讓其他開發者（或未來的你）能快速理解你設計的這些 class。你只需要說明你的設計，不需要說明你的實做，因此你應該：

1. 展示你的那些 class，並且說明繼承與多型的關係（如果有的話）。
2. 對每個 class 列出其所有成員（instance variable、static variable、instance function、static function）的所有資訊（變數的類別、名稱、初始值等；函數的完整 header）。
3. 針對每一個成員變數，說明其在該 class 中作何用途。
4. 針對每一個成員函數，說明其參數的性質與用途、其回傳值的性質與用途，以及呼叫此函數前後你的成員變數會有何變化。
5. 這些 class 中若有用到樣板和例外處理，也請說明之。
6. 加上其他應該補充的資訊。

你的說明應該精確而簡潔（precise and concise）。所謂的精確就是不模糊（廢話 ?!），例如「執行完此函數後，某某陣列內的整數會被排序」就比「執行完此函數後，某某陣列內的整數會被由小到大排序」不精確。所謂的簡潔就是用字精要、少說廢話。這份文件***不能超過四頁***，超過的部份不列入批改。

第三題

（20 分）第一題截止後，我們會讓同學們互相檢視彼此的本題程式碼。請在 PDOGS 上批改你被隨機分配到的第一題程式碼，根據它在正確性以外的部份給它 1 至 5 分的評分，並且說明你給分的依據。建議在評分時參考以下六個面向。在前五個面向上，一個面向上做得好就得一分，還不錯則半分，不好則零分；在第六個面向上則在有必要時扣分。六個面向的分數合計後無條件進入即為你最後給的總分。

- 可讀性：變數與函數名稱是否具有合適的資訊量？程式碼排版是否良好且具有前後一致性？是否有合適的註解？關於註解，當然不需要每一行都有註解，但若你發現在某一大段落裡都沒有註解，或某個你感覺很不易看懂的部份沒有註解，你可以指出來；不要直接說「註解太少」但沒有說是哪邊缺乏註解。
- 模組化程度：是否有宣告合適的函數？是否有避免將非常類似的程式片段寫複數次而非寫成函數？是否有避免一個函數做非常多事情？函數間是否有合適的 decoupling？直接閱讀 main function 是否能很快地理解程式在大方向上的運算邏輯？
- 效率：程式運算是否有合理的運算效率？當然我們不要求每個同學都寫出超級有效率的精妙演算法，但至少一個程式不應該進行過多不必要的運算，也不應該耗用過多不必要的記憶體空間。如果你看不出這個程式的效率有明顯的問題，我們建議你直接給一分。

- 擴充性：當要解的問題變得更複雜的時候，我們能不能簡單地修改這個程式以解決新的問題，而不是寧可砍掉重練？這個議題當然也很主觀，所以如果你不能明確地指出在怎樣的新問題上，這個程式會有擴充性問題，我們建議你直接給一分；如果你不能指出很嚴重的問題，我們建議你至少給半分。但對批改者來說，這個關於擴充性的思考其實是很好的訓練。試試看吧！
- 資料結構技巧：你應該檢查那份程式碼有沒有合理地使用 stack 資料結構，如果沒有就扣他一分。當然，請明確地指出他應該錯過的使用時機。
- 其他：如果有任何其他令你想扣分的理由，請明確地寫出來並且在這個面向上扣分；沒有的話就給一分。

本題其中 10 分取決於檢視你的程式碼的同學給你的分數（必要時助教會出來主持公道，請不用緊張）；若你在本次作業中完全沒有寫第一題，那屆時自然沒有人能檢視你的程式碼，你也就得要損失這 10 分了。另外 10 分取決於你對同學的程式碼的評語和評分的合理性和建設性；只要你有完成評語和評分，且沒人檢舉你或有人檢舉你但助教檢視後認為你沒問題，你就會得到這 10 分。