程式設計（112-1）
作業四

作業設計：孔令傑
國立臺灣大學資訊管理學系

繳交作業時，請至 PDOGS（`http://pdogs.ntu.im/`）為第一、二、三、四題各上傳一份 C++ 原始碼（以複製貼上原始碼的方式上傳）。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。這份作業的截止時間是 ***10 月 3 日早上八點***。為這份作業設計測試資料並且提供解答的助教是楊佳芊。

在你開始前，請閱讀課本的第 7 章（關於 Pointer）[1]。

本次作業滿分為 110 分，得幾分就算幾分。若整學期有 n 份作業，則學期的作業總成績即為 n 份作業的總分除以 n（不論超過 100 與否）。

第一題

（20 分）在本題中，你跟你的夥伴要一起用「函數」的做法重新改寫作業二第一題（題幹敘述、輸入與輸出格式皆維持不變）。以下將複製作業二第一題的內容，幫大家複習一下題目。

我們想要寫一支程式，檢查「開心觀光巴士」乘車紀錄狀況，題目將給定一條固定的路線，不包含起點站總共有 n 站。已知在起點站有 w 位乘客上車，而接下來的第 i 站有 y_i 位乘客下車，然後會有 x_i 位乘客上車，並且在終點站時，所有乘客都必須要下車。在每一站，都是所有要下車的乘客下車後，要上車的乘客才上車。此外，巴士的人數上限為 K 人。已知 $w \leq K$。

「開心觀光巴士」營運一段時間後，由於民眾上下車時有時候會有漏刷卡、多刷卡等各種狀況，而「開心觀光巴士」的資訊系統也不是太理想，因此可能會留下不合理的紀錄，讓公司員工在檢查過往乘車紀錄時，可能會看到兩種的不合理現象，第一種為「下車人數大於車上人數」，也就是找到一個車站 k 滿足

$$y_k > w + \sum_{j=1}^{k-1} x_j - \sum_{j=1}^{k-1} y_j ;$$

第二種則是「車上人數大於載客人數上限」，也就是找到一個車站 k 滿足

$$K < w + \sum_{j=1}^{k} x_j - \sum_{j=1}^{k} y_j 。$$

本題會給定一組巴士的乘車紀錄，請幫助公司同仁檢查是否有發生以上任一種不合理現象，有的話，則找出是在哪一站發生並輸出該站的編號。如果這班車發生了複數次不合理現象，也只需要輸出第一次發生在哪一站（亦即有發生的車站中編號最小的）就好。接著請依據該站發生之不合理現象的種類輸出一個字元，若是第一種不合理現象「下車人數大於車上人數」請輸出 N、第二種不合理現象「車上人數大於載客人數上限」請輸出 C、同時在這個車站出現兩種不合理現象則請輸出 B；若全程都沒有發生不合理現象，請直接輸出 0 。

在判定上車後人數是否超過 K 時，請一律以原問題的下車人數來檢查，即使在該站下車後的人數不合理（是負的），亦即如果下車紀錄已經出現問題（例如因為下車人數超過車上實際人數導致的人數

[1]課本是 Deitel and Deitel 著的 *C++ How to Program: Late Objects Version* 第七版。

為負值），那麼在計算上車人數時，應該基於這個不合理的人數來進行。舉例來說，假設車上最多只能載 40 人（即 $K = 40$）。假如到某一站時，車上原有 20 人，但下車紀錄為 39 人，導致車上實際人數為 -19 人。若接著紀錄顯示有 50 人上車，儘管這似乎是一個明顯不合理的紀錄，但如果我們基於前面的 -19 人來計算，車上實際人數將是 31 人，這在 $K = 40$ 的限制下是合理的。

你的夥伴已經把 main function 寫好了，要求你按照他的設計寫一個函數；他會把讀入的資料存入對應的變數或陣列，接著把這些變數和陣列傳進你寫的函數，讓你做計算並且回傳是否有車站的紀錄出錯，有的話是哪個車站，以及是哪種錯誤。這個函數的結構大體上與作業二的第二題、第四題的十分相似，最大的不同是這次回傳兩個值的方式是採用 call by reference 的形式，以及從使用靜態陣列變成用動態陣列。

具體來說，這個函數的 prototype 為

```
void checkCorrectness(int stationNum, int initPass, int maxPass,
                      const int** passenger,
                      int& wrongStationID, int& wrongType);
```

其中 `stationNum` 代表除了起點站的總站數，`initPass` 是起點站上車人數，`maxPass` 是巴士人數上限、`passenger` 是一個大小為 `2 * (stationNum - 1)` 個變數的二維動態陣列，裡面每個元素就是從第一站、第二站，直到第 `stationNum - 1` 站（倒數第二站），每一站上車與下車的人數（請參考下方 main function 瞭解具體格式）。最後，`wrongStationID` 代表第一次發生不合理現象的車站編號，若函數參數代表的班次沒有不合理之處就存 0，`wrongType` 存 1、2、3 分別代表第一種（太多人下車）、第二種（太多人上車）或第三種（前兩者同時發生）錯誤，若沒有不合理之處則存什麼都無所謂。

助教們會提供好 main function（如附件 PD112-1_hw04_main01.cpp），程式碼如下：

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

// This is the prototype of the function to be completed
void checkCorrectness(int stationNum, int initPass, int maxPass,
                      const int** passenger,
                      int& wrongStationID, int& wrongType);

int main(){
    int stationNum = 0, initPass = 0, maxPass = 0;
    int** passenger = new int*[2];

    int wrongStationID = 0;
    int wrongType = 0;

    cin >> stationNum >> initPass >> maxPass;
    passenger[0] = new int[stationNum - 1];
    passenger[1] = new int[stationNum - 1];

    for(int i = 0; i < stationNum - 1; i++){
        cin >> passenger[0][i];
```

```
    }
    for(int i = 0; i < stationNum - 1; i++){
        cin >> passenger[1][i];
    }

    checkCorrectness(stationNum, initPass, maxPass,
                     const_cast<const int**>(passenger),
                     wrongStationID, wrongType);

    if(wrongStationID == 0){
        cout << wrongStationID;
    }
    else{
        if(wrongType == 1){
            cout << wrongStationID << ",N";
        }
        else if(wrongType == 2){
            cout << wrongStationID << ",C";
        }
        else if(wrongType == 3){
            cout << wrongStationID << ",B";
        }
    }

    delete[] passenger[0];
    delete[] passenger[1];
    passenger[0] = passenger[1] = nullptr;
    delete[] passenger;
    passenger = nullptr;

    return 0;
}

// Your codes will be copied and pasted here
```

特別注意之一：在這題之中，助教已經在 PDOGS 上設定好上面的「你的夥伴」寫的程式了。你需要完成一個完整的 `checkCorrectness` 函數，自己測試的時候當然需要結合上面的 main function，但在繳交到 PDOGS 時請只上傳這個 `checkCorrectness` 函數，PDOGS 會自動把你上傳的函數跟已經在 PDOGS 上的程式拼起來去編譯。換言之，在本題你被迫必須要實作本題指定的函數；如果你上傳了任何帶有你寫的 main function 的程式，你會無法得到分數的！

特別注意之二：大家應該有注意到，在上方程式碼中我們用了一個新關鍵字：`const_cast`，這是在 `static_cast` 之外的另一種轉型（casting，C++ 共有四種轉型）。首先，我們理解傳入函數的引數型態應該是 `const int**`，因為如果不設成 constant，那函數裡面就可能會修改傳入的 `passenger` 所

指向的動態陣列的內容，這是我們所不願發生的。但顯然在 main function 裡面我們宣告 `passenger` 時不能將之宣告成 `const int**`，因為這樣就不能用 `cin` 填入數值。如果我們想把 main function 中的一個 `int**` 指標當引數傳給型態為 `const int**` 的參數，就會發生 compilation error，因為在 C++ 中這兩種指標是不同且必須區隔的。為了解決這個問題，我們在要把 `passenger` 傳入函數前將之轉型成 `const int**`，而做法就是使用 `const_cast`。透過這個例子，我們也順便介紹 `const_cast` 給大家！

輸入輸出格式

系統會提供一共 10 組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有三列，第一列裝著三個正整數，依序是 n、w、K，代表除了起點站的總站數、起點站上車人數，以及巴士載客限制；第二列存了 $n-1$ 個非負整數，依序是 x_1、x_2 直到 x_{n-1}，代表第一站的上車人數、第二站的上車人數，直到終點站前一站的上車人數；第三列存了 $n-1$ 個非負整數，依序是 y_1、y_2 直到 y_{n-1}，代表第一站的下車人數、第二站的下車人數，直到終點站前一站的下車人數。已知 $2 \le n \le 20$、$1 \le w \le 500$、$1 \le K \le 500$、$0 \le y_i \le 50$、$0 \le x_i \le 50$、$w \le K$。請依題目指示，找到第一次出現不合理現象是在哪一站，並輸出該車站編號以及不合理現象種類，兩個字之間以一個逗號隔開。若全程都沒有出現上述不合理現象，就輸出 0。

舉例來說，如果輸入是

```
5 20 40
6 9 3 2
8 5 7 1
```

則輸出應該是

```
0
```

如果輸入是

```
5 2 40
6 9 3 2
8 5 7 1
```

則輸出應該是

```
1,N
```

如果輸入是

```
5 20 40
6 9 3 50
8 5 7 1
```

則輸出應該是

```
4,C
```

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，你可以使用任何方法。

評分原則

這一題的所有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。

第二題

（20 分）本題將給定三個整數 x_1、x_2 和 x_3，請找出他們的中位數，也就是他們之中第二大的數字。

特別說明：這一題是 110-1 的第一次期中考的第一題，應該明顯地比本次作業的其他題容易些。重新出這一題，一方面是讓大家喘口氣，畢竟學期至今每週都有作業、每次都要寫四題，而本週的另外三題也都不是什麼太簡單的題目，讓這一題容易些，希望能降低一部分同學的壓力。此外，也是讓大家對於即將到來的小考、期中考可以不要過度擔憂（但當然到時候不會每一題都這麼容易喔！）。你可能每一份作業都寫得很辛苦，但希望到學期中段時，你會覺得面對作業二、作業三的題目你是有把握的，到學期末時則是覺得作業七、作業八你搞得定。若是如此，並且在考試時正常發揮，那應該就不用太擔心成績了！不是每個人都能 A+，但能做到前述那樣的話，應該總是能通過這門課的！總之，請大家繼續加油吧！

輸入輸出格式

系統會提供一共 10 組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中有三個整數，依序是 x_1、x_2 和 x_3，已知 $0 \le x_i \le 1000$，數字兩兩以一個空白隔開。讀入這些資訊後，請依上述規則，輸出指定的數字。

舉例來說，如果輸入是

```
3 6 2
```

則輸出應該是

```
3
```

如果輸入是

```
3 3 3
```

則輸出應該是

```
3
```

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，你可以使用任何方法。

評分原則

這一題的所有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。

第三題

（50 分）本題我們要來實作電影推薦系統「NTUflix」，題目會給定總共有 n 部電影，並且附上 m 位會員的電影評分紀錄。我們會指定實作一套演算法，根據評分紀錄去推薦第 p 號會員一部最合適的電影。推薦時，首先找出所有第 p 號會員沒有看過的電影。再去逐一對於這裡面每一部電影，計算 m 位成員的評分總和，並找出評分總和最大的電影予以推薦[2]。若同時有兩部以上的電影平手，我們優先選擇評分次數較多的電影，若仍平手，就選擇編號較小的。

電影評分的紀錄大致長的像表 1，本題中我們假設每一位會員看過一部電影後一定會評分剛好一次，並且評分範圍限制在 1 到 5 分，因此，如果這部電影還沒有被該會員觀看過，在表 1 中將直接以 0 顯示。舉例來說，1 號會員共看過三部電影（編號 4、6、7），給予的評分依序是 3、5、4 分。

會員編號	電影編號								
	1	2	3	4	5	6	7	8	9
1	0	0	0	3	0	5	4	0	0
2	0	0	1	4	0	3	2	0	1
3	1	1	5	3	0	3	1	1	0
4	2	0	1	0	4	4	3	1	0

表 1: 會員電影評分紀錄

請根據上述規則，找出最適合推薦給第 p 號會員的一部電影，並依序輸出推薦的電影編號、該電影的評分次數、該電影的評分總和。

特別注意：adjacency list

這一次我們將練習本週教到的動態記憶體配置（Dynamic Memory Allocation）以及 adjacency list。如果照著前幾週的作法，我們會將電影評分紀錄儲存成一個至少 $m \times n$ 的二維陣列（是一種 adjacency matrix），長得跟表 1 差不多，但這種作法在電影數多、會員數多時，會讓這個二維陣列非常地大，造成系統的負擔以及運算效率低落。更重要的是，在一般實務情境中大部分會員都只有看過少部分電影，使得這個 $m \times n$ 的陣列中儲存著許多的 0（這就是所謂的 Sparse Matrix、稀疏矩陣），顯然是不太好的

[2]這只是一個粗略的估算熱門程度的方式，但大致上也兼顧被觀看量與受歡迎度了。實際上當然有更好的演算法，但這題的重點不在此，我們就點到為止。

設計。為了解決這個問題，我們可以用動態記憶體配置的語法去建立兩個 adjacency list，分別儲存每個會員已觀看的電影編號和他們對這些電影的評分。如此一來，我們只需儲存每個成員已觀看的電影的評分，而不是儲存整個矩陣，這將大大減少所需的儲存空間，以及後續程式執行的效率。

基於以上，我們建議你先建立了一個整數指標 `int* movieListSizes` 來儲存每個成員已觀看的電影的數量。接著你可以宣告兩個指向整數指標的指標（例如 `int** movieLists` 和 `int** scoreLists`），分別儲存每個會員看過的電影編號，以及每個會員給過的分數，並根據會員的數量、每一位會員觀看過的電影數量，來動態分配適當的空間。舉例來說，如果用這些動態陣列去儲存表 1 的內容，結果應該如圖 1 所示。以後面的第三筆範例測試資料，可以看出資料中的 10 位會員，實際上看過的電影都不超過 5 部，但是平台上總共有 1000 部電影，若以二維陣列來儲存電影評分資料，將耗費非常多記憶體去儲存 0。若使用 adjacency matrix，即使用了動態記憶體配置，也要大約 40 KB 的空間，但用了 adjacency list 之後，只要 0.4 KB 左右，差距高達百倍。

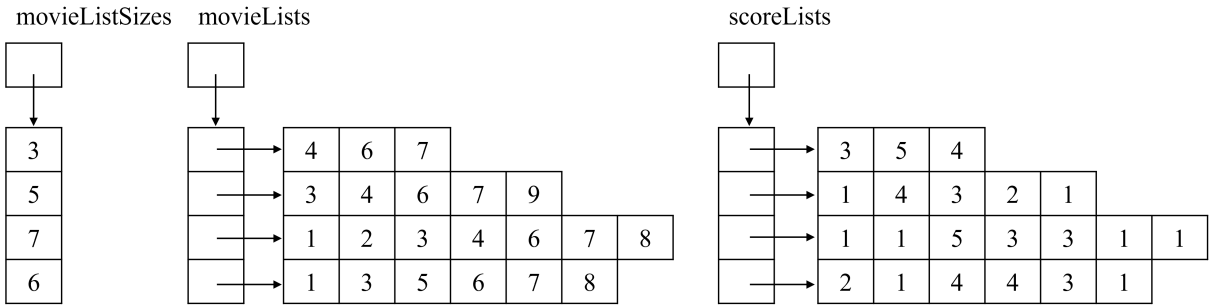

圖 1: 用動態陣列儲存觀看與評分紀錄

所以，請大家在這一題好好練習使用 adjacency list 來實作吧！請注意，在 PDOGS 中，助教會設定好記憶體使用量上限，也會設計好對應的測資，讓大家的程式一寫 adjacency matrix 就會超出記憶體用量上限，跳出 Memory Limit Exceed 的錯誤並因此無法得分。

輸入輸出格式

系統會提供一共 15 組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有 $m+1$ 列，第一列裝著三個正整數，依序是 m、n、p，依序代表會員數量、電影數量、指定的會員編號；第二列到第 $m+1$ 列中的第 i 列代表第 i 號會員的電影評分，在一列中會先有一個整數 u_i，代表第 i 號會員總共看過幾部電影。接著會有 u_i 對整數 (j, x_{ij})，其中每一對的兩個整數代表看過的電影編號 j 與電影評分 x_{ij}，例如若第二行輸入為 3 4 3 6 5 7 4，就代表著第 1 號會員，一共看過 3 部電影，分別是 4 號電影並且評價了 3 分、6 號電影並且評價了 5 分、7 號電影並且評價了 4 分。已知每個會員看過的電影編號 j 不會重複，並且會由小到大依序輸入；也已知 $2 \le m \le 1000$、$2 \le n \le 1000$、$1 \le p \le m$、$1 \le u_i \le n$、$x_{ij} \in \{1, 2, 3, 4, 5\}$。最後，也已知 $u_i \le n$ 且其中 $u_p < n$，也就是會員 p 至少有一部電影沒看過，可以推薦給會員 p。每一列的任兩個相鄰的整數間以一個空白字元隔開。

請依題目要求與指示，實作演算法，並依照推薦之順序依序輸出被推薦的電影編號、這部電影被觀看的次數、這部電影的評分總和，兩個數字之間以一個逗點隔開。舉例來說，如果輸入是

```
4 9 3
3 4 3 6 5 7 4
5 3 1 4 4 6 3 7 2 9 1
```

```
7 1 1 2 1 3 5 4 3 6 3 7 1 8 1
6 1 2 3 1 5 4 6 4 7 3 8 1
```

則輸出應該是

```
5,1,4
```

如果輸入是

```
4 9 3
3 4 3 6 5 7 4
5 3 1 4 4 6 3 7 2 9 5
7 1 1 2 1 3 5 4 3 6 3 7 1 8 1
7 1 2 3 1 5 4 6 4 7 3 8 1 9 5
```

則輸出應該是

```
9,2,10
```

如果輸入是

```
10 1000 3
2 1 5 2 4
4 11 4 12 5 13 1 14 2
2 31 4 32 2
3 46 1 47 3 48 5
1 76 3
2 1 5 2 4
4 11 4 12 5 13 1 14 2
2 31 4 32 2
3 46 1 47 3 48 5
1 76 3
```

則輸出應該是

```
1,2,10
```

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：

- 確定可以使用的語法包含 `if-else`、`for`、`while`、陣列、函數、指標、動態記憶體配置、call by reference、`<climits>` 裡面所有的東西、`<iomanip>` 裡面所有的東西、`<cmath>` 裡面的 `abs()` 和 `sqrt()`、`sizeof()`、`static_cast()`、constants 等。

- 確定不可以使用的語法包含 `printf`、`scanf`、`max`、`min`、`<cmath>` 裡面除了 `abs()` 和 `sqrt()` 以外的函數等等。

請注意正面表列的固然是都確定可以用，但沒有被負面表列的不表示可以用喔！

此外，請幫你的程式寫適當的函數，讓你的程式藉由使用函數來實現理想的程式結構。

評分原則

- 這一題的其中 30 分會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。
- 這一題的其中 20 分會根據你所寫的程式的品質來給分。助教會打開你的程式碼並檢閱你的程式的結構、運算邏輯、可讀性（包含排版、變數命名、註解等等）、可擴充性、模組化程度，以及是否使用了還沒教過的語法。請寫一個「好」的程式吧！

第四題

（20 分）承上題，實作完基礎的電影推薦後，我們將指定另一套更精細的演算法來進行電影推薦。一樣我們要根據 m 位會員的電影評分紀錄，在 n 部電影中，推薦一部最適合的電影給第 p 號會員。在此演算法中，會需要衡量兩個會員的相似度，我們定義兩種「會員間的距離」。在第一種距離中，第 p 號會員和第 r 號會員的第一種距離 $d_1(r,p)$ 是所有電影的評分差值的絕對值總和，亦即若 x_{ij} 為會員 i 對第 j 部電影的評分（若沒看過則為 0），而 J 為所有電影編號的集合，則我們有

$$d_1(r,p) = \sum_{j\in J} |x_{rj} - x_{pj}|。$$

而第二種距離 d_2 則為「看過與否」的差異總和，一部電影若兩個人中僅有一個人看過，則距離就加一，否則就加零。更精確地說，令

$$f(x_{ij}) = \begin{cases} 1 & \text{若 } x_{ij} \geq 1 \\ 0 & \text{若 } x_{ij} = 0 \end{cases},$$

則我們有

$$d_2(r,p) = \sum_{j\in J} |f(x_{rj}) - f(x_{pj})|。$$

以表 1 的數據為例，會員 1 和會員 2 的第一種距離是 $d_1(1,2) = 1+1+2+2+1 = 7$，這些差距分別來自他們評分不同或看過與否不同的第 3、4、6、7、9 部電影，但他們的第二種距離是 $d_2(1,2) = 1+1$，因為在「有沒有看過」方面，他們只有第 3、9 部電影有差別。

現在我們定義本題要用的演算法。用最簡單的方式說，如果要推薦一部電影給會員 p，我們的目標是找到跟會員 p 最像的會員 r，然後從會員 r 看過且會員 p 沒看過的電影中挑出會員 r 最喜歡的去推薦給會員 p。為此，我們會先幫所有其他會員計算與會員 p 的第一種距離 d_1，並試著找出和會員 p 距離最小的會員。如果有兩位會員以上在第一種距離同為最小，就再比較這些平手的會員的第二種距離 d_2，若還是有兩位會員以上在第二種距離同為最小，就從其中選擇編號最小的會員。如此找出與會員 p 最相似的會員後，我們稱其為會員 r，接著從會員 r 看過但會員 p 沒看過的電影中找會員 r 評分最高的電影去推薦給會員 p。如果有複數部電影都同為評分最高，就從其中找編號較小的電影。如果最相似的會員 r 看過的電影會員 p 都已經看過了，就使用第三題的演算法去推薦一部電影給會員 p。

請根據題目指示，找出最適合推薦給第 p 號會員的一部電影，並依序輸出推薦的電影編號、該電影的評分次數、該電影的評分總和。

輸入輸出格式

系統會提供一共 10 組測試資料，輸入與輸出格式與第三題相同。舉例來說，如果輸入是

```
4 9 3
3 4 3 6 5 7 4
5 3 1 4 4 6 3 7 2 9 1
7 1 1 2 1 3 5 4 3 6 3 7 1 8 1
6 1 2 3 1 5 4 6 4 7 3 8 1
```

則輸出應該是

```
9,1,1
```

如果輸入是

```
4 9 3
3 4 3 6 5 7 4
5 3 1 4 4 6 3 7 2 9 5
7 1 1 2 1 3 5 4 3 6 3 7 1 8 1
7 1 2 3 1 5 4 6 4 7 3 8 1 9 5
```

則輸出應該是

```
9,2,10
```

如果輸入是

```
10 1000 3
2 1 5 2 4
4 11 4 12 5 13 1 14 2
2 31 4 32 2
3 46 1 47 3 48 5
1 76 3
2 1 5 2 4
4 11 4 12 5 13 1 14 2
2 31 4 32 2
3 46 1 47 3 48 5
1 76 3
```

則輸出應該是

```
1,2,10
```

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，你可以使用任何方法。

評分原則

這一題的所有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。