程式設計（105-2）
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/judge/`）為第一、二題上傳一個 PDF 檔，再為第三題與第四題各上傳一份 C++ 原始碼（以複製貼上原始碼的方式上傳）。第四題是 bonus 加分題。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。

這份作業的截止時間是 ***2017 年 3 月 20 日凌晨一點***。在你開始前，請閱讀課本的第 5.1–5.7 和 6.5–6.8 節[1]。為這份作業設計測試資料並且提供解答的助教是李昱賢（Rick Lee）。

第一題

（10 分）請考慮下面這個跟投影片第 58 頁的程式很像的程式：

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

void printArray(int [][2], int);
int main()
{
  int num[2][5] = {1, 2, 3, 4, 5}; // five 0s
  printArray(num, 5);
  return 0;
}
void printArray(int a[][2], int len)
{
  for(int i = 0; i < len; i++)
  {
    for(int j = 0; j < 2; j++)
      cout << a[i][j] << " ";
    cout << "\n";
  }
}
```

請用自己的話說明為什麼這樣寫會發生編譯錯誤，但把 `int num[2][5]` 改成 `int num[5][2]` 就可以成功編譯。

第二題

（30 分；每小題 10 分）很多人開始學程式設計的時候，會濫用 global variable。比如說下面這個程式：

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

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

const int cnt = 5;
int score[cnt] = {0};
int sum = 0;

void setScore();
float getAvg();

int main()
{
  setScore();
  cout << getAvg() << "\n";
  cout << getAvg() << "\n";

  return 0;
}

void setScore()
{
  for(int i = 0; i < cnt; i++)
    cin >> score[i];
}
float getAvg()
{
  for(int i = 0; i < cnt; i++)
    sum += score[i];
  return static_cast<float>(sum) / cnt;
}
```

這個程式的兩個函數想做的事都很簡單：`setScore()` 是去讓使用者輸入 5 個成績，將這 5 個成績記錄在 `score` 陣列中；`getAvg()` 則是計算陣列中的 5 個成績的平均數。這個程式裡有三個 global variable，包括記錄班上人數的 `cnt`、儲存成績的 `score` 陣列，以及在計算平均數時需要用到的 `sum`。

(a) 請執行這個程式，並且發現第二個印出的數字會是第一個數字的兩倍。請用自己的話解釋為什麼。接著請把 `sum` 這個 global variable 改成 local，寫出一個可以正確執行的程式以解決這個問題，並且讓 `getAvg()` 這個函數不管執行幾次都會得到一樣的結果（如果 `score` 沒被改變的話）。

(b) 現在的 `setScore()` 和 `getAvg()` 這兩個函數的 header 提供的資訊很少，因為它們都沒有任何參數，就算要在註解寫下這兩個函數會對哪些 global variable 做處理，讀者也必須離開這兩個函數去別處看其他變數的定義。請把 `score` 和 `cnt` 改成 local variable，寫出一個可以正確執行的程式，讓 `setScore()` 和 `getAvg()` 的 header 有比較多的資訊。

(c) 原本的三個 global variable 中，只有 `cnt` 是比較可以被接受的。請用自己的話解釋為什麼。

提示：它是個 constant variable！

第三題

（60 分）現在火車（或高鐵或各種列車）座位票的銷售，都是透過資訊系統進行的。每當有人想要透過網路或臨櫃訂購一張車票時，他必須輸入想搭乘的車次以及旅程的起訖站資訊。系統接著就必須查詢該車次是否還有座位能滿足這個需求，如果有，那就將座位保留給這名乘客，等乘客付款之後這張票就賣出，這趟旅程上的座位自然就不能再賣給別人；反之，如果已經沒有座位了，這名乘客就買不到票，若他就這麼放棄，所有座位的狀態就都會維持不變。

在本題中，我們要實作一個簡單的列車座位售票系統。由於每個車次都是獨立作業的，我們將只考慮一個車次（也就是一般列車）。這個車次由起站到訖站依序會通過車站 0、1、$\cdots\cdots$、m，也就是說車站 0 是起站，車站 m 是訖站，而這個車次共有 n 個座位，依序編號為座位 1、2、$\cdots\cdots$、n。我們稱呼車站 $i-1$ 跟車站 i 之間的那一段路為為路段 i，所以我們有路段 1、路段 2 一直到路段 m。

當一位乘客想要購買這個車次的座位時，他必須輸入起站編號 $s \in \{0,1,...,m\}$ 與迄站編號 $t \in \{0,1,...,m\}$，並且滿足 $s < t$。系統接著就會查詢售票狀態，看看能否賣出這張票。系統會依序做下列兩階段的查詢：

1. 系統會依序查詢座位 1、座位 2 直到座位 n。如果有任何一個座位從車站 s 到車站 t 之間的 $t-s$ 個路段都是空的（未賣出），就把這個座位的這 $t-s$ 個路段賣給這個乘客。換言之，如果有多個座位滿足這個條件，就賣編號最小的那一個座位。

2. 如果沒有任何一個座位能單獨地滿足這位乘客的需求，就進入第二階段。為了（從某個角度來說）最大化能搭載的乘客人數，這個系統會嘗試讓乘客在旅途中「換座位」，也就是用多個座位來滿足這名乘客的需求。系統會把這 $t-s$ 個路段都拆開來獨立處理，嘗試對每一個路段都找編號最小的空位來賣給這個乘客。換言之，除非有任何一個路段是所有 n 個座位都已經被賣掉了，否則這個乘客就會得到一張（可能支離破碎）的車票[2]。請注意系統不會做任何嘗試去減少這名乘客換座位的次數。

讓我們舉個例子來說明上述的座位銷售方法。我們用 (s_i, t_i) 表示第 i 筆購票時傳入的資訊，其中 s_i 跟 t_i 分別是起站跟訖站編號。假設有個車次有 7 個路段和 3 個座位，並且依序有下列六張訂單：

i	1	2	3	4	5	6
(s_i, t_i)	$(0,3)$	$(2,5)$	$(4,7)$	$(0,4)$	$(0,4)$	$(3,7)$

則上述的售票規則會將座位已如圖 1 的方式售出。在圖 1 中，3 列表示 3 個座位、7 欄表示 7 個路段、格子中填入 i 表示該座位該路段被賣給第 i 位乘客，而空格則表示該座位該路段沒有被賣出。請注意乘客 3 買到的座位 1 而不是座位 3，因為我們會在所有能獨自滿足該乘客需求的座位中，賣給他編號最小的；乘客 4 不需要換座位，因為座位 3 可以完整地滿足他的需求；乘客 5 沒有買到票，因為他訂購的時候路段 3 已經被完全賣完了；最後，請注意乘客 6 會買到需要換座位的票，而且根據上述的演算法，我們賣給他的座位惠要求她換兩次座位，即使在他訂購的當下座位 3 的路段 6 跟 7 都還是空的。

[2]精確點說，是系統會問這名乘客是否願意接受這樣的一張車票；我們只是為了簡化題目而假設乘客都會接受。

座位	路段						
	1	2	3	4	5	6	7
1	1	1	1	6	3	3	3
2			2	2	2	6	6
3	4	4	4	4	6		

圖 1: 售票範例

在本題中，你將會被給定一個車次的路段數、座位數，以及許多依序湧入的訂票資訊。你要按照上述規則進行售票、隨時更新座位銷售狀況、在必要時讓某些乘客空手而回，並在最後告訴我們完整的售票結果。

補充說明：定義與使用函數

由於本週的學習重點是函數，因此你應該定義適當的函數來模組化你的程式。這種事情沒有一定要怎麼做，大原則大概包含：

1. 利用定義函數將 main function 切成幾個大步驟，再怎樣也比一個又臭又長的 main function 好；
2. 會被反覆使用到的程式碼就放進一個函數，不要在數個地方複製貼上；
3. 將一個複雜（或尚稱複雜）的功能在一個自訂函數中實做。至於何謂「複雜」，需要寫上數十行可能是一個標準，或者會讓你生出「要是有人幫我做出這個功能，我就拿它來如此這般」的念頭的，大概就是複雜了。

另外下面的第四題會讓你用另一個演算法處理同樣的售票問題。如果你這一題的程式碼確實做了不錯的模組化，那麼到了下一題你的 main function（和大部分的自訂函數）應該會完全不變，而你只需要修改一兩個你自己定義的函數的內容即可。

總之，經驗是慢慢累積的，請好好試試看吧！

輸入輸出格式

系統會提供許多筆測試資料，每筆測試資料裝在一個檔案裡。在每個檔案中，第一列存放兩個整數 n 和 m，分別代表座位數和路段數。從第二列開始，第 $i+1$ 列代表第 i 筆訂單，其中包含兩個整數 s_i 和 t_i，分別代表起站和訖站的編號。以上每一列的兩個整數都被一個空白鍵隔開。我們知道 $n \in \{1, 2, ..., 50\}$、$m \in \{1, 2, ..., 20\}$、$s_i \in \{1, 2, ..., m\}$ 以及 $t_i \in \{1, 2, ..., m\}$。如果 $s_i < t_i$，則依照本題指定的方式決定是否售票、如何售票給這位乘客，並更新售票狀況；如果 $s_i \geq t_i$，則表示乘客輸入了不合理的購票資訊，此時請直接忽略這一筆訂單。

處理完所有的訂單後，請用 n 列輸出完整的售票結果，每一列上輸出 m 個整數，其中第 i 列的第 j 個整數代表買到第 i 個座位的第 j 個路段的乘客編號。如果某座位的某路段沒被賣掉，則輸出 0。一列中的任兩個整數用一個空白鍵隔開。舉例來說，如果輸入是

```
3 7
0 3
2 5
4 7
0 4
0 4
3 7
```

則輸出應該是

```
1 1 1 6 3 3 3
0 0 2 2 2 6 6
4 4 4 4 6 0 0
```

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。

評分原則

- 這一題的其中 40 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。前 30 分由 15 筆測試資料判定分數，一筆測試資料佔 2 分；後 10 分由 5「組」測試資料判定分數，每一組裡面有若干筆測試資料，全對的話才能得到 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運算邏輯、可讀性，以及可擴充性（順便檢查你有沒有使用上課沒教過的語法，並且抓抓抄襲）。請寫一個「好」的程式吧！

  重點提醒：在 main function 之外沒有自行定義函數者，這部份會得到零分。有自行定義函數當然也不表示這部份會滿分。助教會根據你定義的函數的合理性和適切性給分。

第四題（bonus）

（20 分）雖然第三題的演算法是可以賣很多票，但許多顧客怨聲載道，因為頻繁地換座位非常麻煩。事實上，不用換座位的乘客也有抱怨，因為大量的座位更換事實上影響到每一個人。這樣「過度積極」的售票方式，反而讓許多乘客在買不到免換座位的車票時，就直接選擇其他交通方式了。

有鑑於此，在本題中我們要來修改一下我們的演算法。我們的修改很簡單：讓乘客換座位可以，但最多只能換一次。因此現在在一個乘客輸入購票資訊後，我們的售票流程如下。首先，我們還是先看看有沒有任何一個座位能完全滿足這個乘客，如果有就把能滿足他的編號最小的座位賣給他。這階段和第三題的演算法一樣。有改變的是第二階段：我們試著找座位 j 跟座位 k 來滿足這位乘客，而且讓他只換一次座位。可以是從 j 換一次到 k，也可以是從 k 換一次到 j，但不能從 j 換到 k 再換回 j、不能從 k

換到 j 再換回 k，當然更不能換更多次。所以按照這個新規則，第三題圖 1 中的例子的售票狀況就會變成圖 2，因為原本的售票方式讓乘客 6 換了兩次座位，因此必須改成如圖 2 那樣只換一次。

座位	路段						
	1	2	3	4	5	6	7
1	1	1	1	6	3	3	3
2			2	2	2		
3	4	4	4	4	6	6	6

圖 2: 售票範例二

座位	路段						
	1	2	3	4	5	6	7
1	1	1	1	5	5	5	3
2			2	2	2		5
3	4	4	4	4			

圖 3: 售票範例三

如果有複數對座位都能滿足這位乘客，我們優先挑選兩個座位的編號接近的（因為這樣移動距離通常比較短）；如果還是有多個選擇，則挑選兩個座位的編號和最小的。我們用 $\{j,k\}$ 來表示一對座位。舉例來說：

- 如果有座位組合 $\{1,3\}$、$\{1,5\}$ 和 $\{3,4\}$ 可以滿足某位乘客的需求，我們選 $\{3,4\}$ 來賣給這位乘客，因為這組座位的編號差 1 是最小的。
- 如果有座位組合 $\{5,7\}$、$\{8,10\}$ 和 $\{1,4\}$ 可以滿足某位乘客的需求，我們選 $\{5,7\}$ 來賣給這位乘客，因為在座位編號差最小的兩組座位組合中，$\{5,7\}$ 的編號和是最小的。

選定座位組合 $\{j,k\}$ 之後，最後還有一個問題：在座位 j 跟 k 上各要賣哪些路段？我們的規則是「最大化在編號較小的座位上被賣掉的路段數」。舉例來說，如果在一個 3 個座位、7 個路段的車次上依序有 $(0,3)$、$(2,5)$、$(6,7)$、$(0,4)$ 和 $(3,7)$ 這 5 張訂單，則售票結果將如圖 3 所示。請注意對於乘客 5，我們選了座位組合 $\{1,2\}$ 而非 $\{1,3\}$，而且在選了 $\{1,2\}$ 之後，為了最大化在編號較小的座位（座位 1）上賣出的路段數，我們是讓乘客坐在座位 1 上直到抵達車站 6，而不是在車站 5 就讓他換座位。

本題的輸入輸出格式和第三題一模一樣，只有演算法改變。

針對這個題目，你***可以***使用任何方法。這一題的 20 分都根據程式運算的正確性給分，前 15 分由 15 筆測試資料給分，一筆 1 分；後 5 分由 5 組測試資料給分，一組 1 分。