程式設計（113-1）
作業三

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 ***9 月 24 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彭麒任。

在你開始前，請閱讀課本的第 6.1–6.4 節（關於陣列）、第五章（關於函數）[1]。第 6.7 和 6.8 節也有幫助。

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）許多大樓都有電梯，而電梯的派車規則對乘客的等待時間也頗有影響。在學術上和實務上，都有許多人在研究電梯派車的規則和演算法；在本題我們先熱個身，在一個只有一部電梯的大樓中，探討使用指定演算法的電梯會讓一位乘客等多久，亦即計算乘客的「等待時間」。為了簡單起見，我們會用電梯在接到這位乘客前移動的樓層數做為等待時間的定義。

本題指定的電梯派車演算法非常簡單，嚴格說來甚至沒有「派車」的概念，而是簡單的「無腦來回走到底」，亦即電梯往上的時候就一直往上直到最頂層，然後往下一路到最底層，再往上一路到最頂層，依此類推。當有乘客在等待時，如果電梯和乘客方向相同且乘客在電梯前進的路徑上，電梯會順路直接接到乘客；反之則電梯會先行駛到樓層盡頭再調頭接到乘客。在本題中你會被給定一棟有 n 層樓的大樓，樓層編號為 1、2 直到 n；有一臺電梯此刻在第 f_0 樓，正朝方向 d_0 前進，且有一個乘客此刻正在第 f_1 樓等待往方向 d_1 的電梯，其中 d_0 和 d_1 為 1 表示正在往上，為 -1 則表示正在往下。給定這些資訊，就可以算出電梯在接到這位乘客前移動了幾個樓層，亦即乘客的等待時間。

舉例來說，假設有 $n = 10$ 層樓，電梯正在四樓往下，而乘客正在七樓等待往上，則電梯將先移動 3 層到一樓，再移動 6 層到七樓然後接到乘客，乘客的等待時間即為 $3 + 6 = 9$。請注意我們只計算電梯「移動」了幾層（間隔數），所以電梯在一樓折返的動作不算入乘客的等待時間。因此，假設有 $n = 10$ 層樓，電梯正在四樓往下，而乘客正在一樓等待往上，則電梯將先移動 3 層到一樓，然後不移動任何一層就折返並且接到乘客，乘客的等待時間即為 $3 + 0 = 3$。乘客當然也有可能運氣很好，例如電梯正在四樓往上，而乘客也正在四樓等待往上，那等待時間就是 0。

在本題中，我們除了要請你完成前述計算，也要求你使用「函數」將你的程式模組化。我們已經把 main function 寫好了，你得按照我們的設計寫一個函數，在函數內做計算並且回傳乘客的等待時間。具體來說，函數的 prototype 為

```
int getWaitingTime(int floorCnt, int currentFloor, int currentDirection,
                   int passengerFloor, int passengerDirection);
```

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

其中 floorCnt 代表樓層總數，currentFloor 是電梯當前所在的樓層，currentDirection 是電梯當前行駛的方向（1 表示向上，−1 表示向下），passengerFloor 是乘客呼叫電梯所在的樓層，passengerDirection 是乘客呼叫電梯的方向（1 表示向上，−1 表示向下）。給定這些資訊，函數應該回傳這位乘客搭到電梯所需的等待時間。

已經寫好的 main function 如下（如附件 PD113-1_hw03_main01.cpp）：

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

// This is the prototype of the function that you need to implement
int getWaitingTime(int floorCnt, int currentFloor, int currentDirection,
                   int passengerFloor, int passengerDirection);

int main()
{
    // read data
    int floorCnt = 0, currentFloor = 0, currentDirection = 0,
        passengerFloor = 0, passengerDirection = 0;

    cin >> floorCnt;
    cin >> currentFloor >> currentDirection;
    cin >> passengerFloor >> passengerDirection;

    // calculate the waiting time of the passenger
    int waitingTime = getWaitingTime(floorCnt, currentFloor,
                                     currentDirection, passengerFloor,
                                     passengerDirection);

    // print out the waiting time of the passenger
    cout << waitingTime << endl;

    return 0;
}

// PDOGS will copy and paste the code you upload to this place
// and compile the resulting program
```

請完成 getWaitingTime 函數以完成這個程式。

特別注意：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 getWaitingTime 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 getWaitingTime 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！

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 3 列，第一列包含一個整數 n，第二列有兩個整數 f_0 和 d_0，第三列有兩個整數 f_1 和 d_1，每一列的兩個整數之間被一個空白隔開。已知 $1 \le n \le 10$、$1 \le f_i \le n$、$d_i \in \{-1, 1\}$，且乘客和電梯都不會在頂樓要往上或在一樓要往下，亦即不會 $f_i = n$ 且 $d_i = 1$，也不會 $f_i = 1$ 且 $d_i = -1$。讀入這些資訊後，請輸出一個整數，代表這位乘客搭到電梯前的等待時間。

舉例來說，如果輸入是

```
10
4 1
7 -1
```

則輸出應該是

```
9
```

如果輸入是

```
6
2 1
5 -1
```

則輸出應該是

```
5
```

如果輸入是

```
6
2 1
2 1
```

則輸出應該是

```
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）承上題，現在有多位乘客，我們想計算他們每個人的等待時間。在本題中你會被給定一棟有 n 層樓的大樓，樓層編號為 1、2 直到 n；有一臺電梯此刻在第 f_0 樓，正朝方向 d_0 前進，且有 m 個乘客此刻正在等待，第 $i \in \{1, ..., m\}$ 位乘客在 f_i 樓等待往方向 d_i 的電梯，其中 d_0 和 d_i 為 1 表示正在往上，為 -1 則表示正在往下。給定這些資訊，就可以算出電梯在接到每位乘客前移動了幾個樓層，亦即乘客的等待時間。舉例來說，如果這棟建築有 $n = 6$ 層樓，目前有 $m = 3$ 個乘客在呼叫電梯，電梯目前在一樓向上行駛，第一位乘客在二樓要往上，第二位乘客在三樓要往上，第三位乘客在五樓呼叫電梯要往下，則三位乘客的等待時間依序是 1、2 和 6。

在本題中，我們除了要請你完成前述計算，也要求你使用「函數」將你的程式模組化。我們已經把 main function 寫好了，你得按照我們的設計寫一個函數，在函數內做計算並且回傳乘客的等待時間。具體來說，首先我們定義一個全域常數（global constant）

```
const int MAX_PASSENGER_CNT = 18;
```

代表乘客人數的上限，其值為 18 是因為我們將限制總樓層數為 10 層、任兩位乘客的樓層和方向不會都一樣，且沒有乘客在一樓往下或在頂樓往上。函數的 prototype 為

```
void getWaitingTimes(int floorCnt, int passengerCnt,
        int currentFloor, int currentDirection,
        const int passengerFloors[], const int passengerDirections[],
        int waitingTimes[]);
```

其中 `floorCnt` 代表樓層總數，`passengerCnt` 代表乘客總數、`currentFloor` 是電梯當前所在的樓層，`currentDirection` 是電梯當前行駛的方向（1 表示向上，-1 表示向下），`passengerFloors` 是長度為 `MAX_PASSENGER_CNT` 的一維陣列，其中的第 $i-1$ 個元素是乘客 i 呼叫電梯所在的樓層，`passengerDirection` 是長度為 `MAX_PASSENGER_CNT` 的一維陣列，其中的第 $i-1$ 個元素是乘客 i 呼叫電梯的方向（1 表示向上，-1 表示向下）。函數應該將每位乘客搭到電梯所需的等待時間存入長度為 `MAX_PASSENGER_CNT` 的 `waitingTimes` 陣列，其中第 $i-1$ 個元素是乘客 i 的等待時間。

已經寫好的 main function 如下（如附件 PD113-1_hw03_main02.cpp）：

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

const int MAX_PASSENGER_CNT = 18;

// This is the prototype of the function that you need to implement
// the return values should be stored in the array waitingTimes
void getWaitingTimes(int floorCnt, int passengerCnt,
        int currentFloor, int currentDirection,
        const int passengerFloors[], const int passengerDirections[],
        int waitingTimes[]);

int main()
{
```

```
    // read data
    int floorCnt, passengerCnt;
    cin >> floorCnt >> passengerCnt;

    int currentFloor, currentDirection;
    cin >> currentFloor >> currentDirection;

    int passengerFloors[MAX_PASSENGER_CNT] = {0};
    int passengerDirections[MAX_PASSENGER_CNT] = {0};
    int waitingTimes[MAX_PASSENGER_CNT] = {0};

    for (int i = 0; i < passengerCnt; ++i)
        cin >> passengerFloors[i] >> passengerDirections[i];

    // calculate the waiting times of all passengers
    getWaitingTimes(floorCnt, passengerCnt, currentFloor, currentDirection,
                    passengerFloors, passengerDirections, waitingTimes);

    // print out the waiting times of all passengers
    for(int i = 0; i < passengerCnt - 1; i++)
        cout << waitingTimes[i] << ",";
    cout << waitingTimes[passengerCnt - 1];

    return 0;
}

// PDOGS will copy and paste the code you upload to this place
// and compile the resulting program
```

請完成 getWaitingTimes 函數以完成這個程式。

你可能已經發現了，如果在第一題你已經完成了 getWaitingTime 函數，那實做 getWaitingTimes 可以非常簡單：就呼叫 getWaitingTime m 次就好。在本題中你完全可以這麼做，甚至我們鼓勵你這麼做；如果你願意，也可以想想有沒有更有效率的計算方式，但本題我們不會要求執行效率。

特別注意：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 getWaitingTimes 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 getWaitingTimes 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！

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 $m+2$ 列，第一列包含兩個整數 n 和 m，第二列有兩個整數 f_0 和 d_0，從第三列起，第 $i+2$ 列有兩個整數 f_i 和 d_i，每一列的兩個整數之間被一個空白隔開。已知 $1 \le n \le 10$、$1 \le m \le 18$、$1 \le f_0 \le n$、$d_0 \in \{-1, 1\}$、$1 \le f_i \le n$、$d_i \in \{-1, 1\}$，且乘客和電梯都不會在頂樓要往上或在一樓要往下。讀入這些資訊後，請輸出 m 個整數，依序代表乘客 1、2 直到 m 搭到電梯前的等待時間，兩個數字之間以一個逗點隔開。

舉例來說，如果輸入是

```
6 3
1 1
2 1
3 1
5 -1
```

則輸出應該是

```
1,2,6
```

如果輸入是

```
9 4
2 -1
7 1
3 1
4 -1
1 1
```

則輸出應該是

```
7,3,14,1
```

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

這一題的其中 20 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。

第三題

（50 分）承上題，我們還是有 n 層樓、一部電梯、m 個乘客，但此刻電梯是在某個樓層靜止著。你必須使用「先往某個方向走到樓層 a，再回頭往另一個方向走到樓層 b，最後再回頭往一開始的方向走」的演算法去載乘客，但你可以決定電梯的初始方向（開始時是向上行駛還是向下行駛），以及電梯的迴轉樓層 a 與 b（亦即電梯不再需要傻傻地走到盡頭才迴轉），以最小化所有乘客的總等待時間。計算完成後，請輸出這個被最小化的所有乘客的總等待時間。

舉例來說，假設這棟建築有 $n = 6$ 層樓，目前有 $m = 2$ 個乘客在呼叫電梯，電梯目前在三樓，第一位乘客在二樓呼叫電梯要往上，第二位乘客在五樓呼叫電梯要往下。如果起始時電梯往上行駛，最佳走法是到五樓迴轉向下，然後到二樓迴轉向上，結果是兩位乘客的等待時間分別是 5 和 2，總計是 $5 + 2 = 7$。如果起始時電梯往下行駛，最佳走法是到二樓迴轉向上，然後到五樓迴轉向下，結果是兩位乘客的等待時間分別是 1 和 4，總計是 $1 + 4 = 5$。由於第二個方案比較好，最終我們應該輸出 5。

小提示：要完成這題，能不能使用前面寫過的函數呢？答案是可以！首先，我們知道電梯啟動時只有往上或往下這兩個選項，我們幫這兩個選項個別找出最佳折返樓層即可，而找最佳折返樓層也不難，就是所有乘客中的最高和最低樓層。假設找出來的最高和最低樓層分別是 H 和 L，那其實就很像是所有乘客在「另一棟」從 1 樓到 $H - L + 1$ 樓的大樓，等待的位置也相對應地調整。然後你也知道電梯的起始樓層，再做一點調整，就可以呼叫第二題寫好的函數來算出「如果啟動時往上」和「如果啟動時往下」的最小化總等待時間，就可以完成這一題了。這一題你沒有非這麼做不可，但希望大家都有感受到，寫一些好的函數讓自己的程式模組化之後，對於開發規模相對大的程式是很有幫助的！

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 $m + 2$ 列，第一列包含兩個整數 n 和 m，第二列有一個整數 f_0，從第三列起，第 $i + 2$ 列有兩個整數 f_i 和 d_i，每一列的兩個整數之間被一個空白隔開。已知 $1 \le n \le 10$、$1 \le m \le 18$、$1 \le f_0 \le n$、$1 \le f_i \le n$、$d_i \in \{-1, 1\}$，且乘客和電梯都不會在頂樓要往上或在一樓要往下。讀入這些資訊後，請輸出一個整數，代表被最小化後的乘客總等待時間。舉例來說，如果輸入是

```
6 2
3
2 1
5 -1
```

則輸出應該是

```
5
```

如果輸入是

```
9 4
2
1 1
3 1
4 -1
```

```
7 1
```

則輸出應該是

```
21
```

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、`<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、動態配置記憶體等等。

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！

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運算邏輯、可讀性（包含排版、變數命名、註解等等）、可擴充性，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）有一間工廠收到了 n 張訂單，每一張訂單都有其對應的營收 r_i 元、勞動力需求 h_i 與原料需求 s_i；只要花費 h_i 單位的勞動力和 s_i 單位的原料，就能完成訂單 i 並且收入 r_i 元。受限於此工廠有限的總勞動力 H 與總原料量 S，工廠可能無法一次接下所有的訂單，因此我們需要決定應該要接哪些訂單。

表 **??** 是一個例子。共有 $n = 7$ 張訂單。假設 $H = 80$、$S = 1000$，那麼「接受訂單 1、2、7」就是一個可行方案（花費勞動力 59 單位、原料 450 單位），可得到營收 $4100。因為還有餘裕，我們還能再接受訂單 3，進而將營收擴大到 $4900。

我們的目標是從 n 張訂單中接受一部份訂單，不超過總勞動力與總原料量限制，並且帶來最大的營收。若要更精確地用數學式子描述，我們可以令 x_i 表示是否接訂單 i，1 代表接單，0 代表不接單。那

訂單編號	營收	勞動力需求	原料需求
1	$1000	20	200
2	$500	15	100
3	$800	12	400
4	$400	22	200
5	$2000	40	500
6	$600	34	300
7	$2600	24	150

Table 1: 接單問題範例

我們要求解的最佳化問題就是

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} r_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} h_i x_i \le H \\ & \sum_{i=1}^{n} s_i x_i \le S \\ & x_i \in \{0,1\} \quad \forall i = 1, ..., n。 \end{aligned}$$

這個問題看似有點複雜，因此你的老闆給了你一個演算法來解決。一開始我們先假定讓工廠接所有的訂單，也就是說 $x = (x_1, x_2, ..., x_n) = (1, 1, ..., 1)$，如果勞動力和原料竟然足夠，那這就是最佳方案，也不用繼續演算下去了。如果這個方案以工廠的勞動力或原料量無法負擔，那我們就選擇捨棄一份訂單，若仍超過工廠負荷，就再捨棄一份訂單，直到工廠的勞動力及原料量都可以負擔為止。

那麼在以上過程中，我們應該如何決定捨棄哪一張訂單呢？在此就要使用到「C/P 值」的概念。舉例來說，若我們不考慮原料的因素，只考慮勞動力的影響，我們就會定義訂單 i 的 C/P 值為 $\frac{r_i}{h_i}$，進而能讓我們找出 C/P 值較低的訂單並將之捨棄。不過，本題的狀況必須同時考量勞動力與原料量兩種因素，因此我們定義每一份訂單的 C/P 值為

$$\frac{r_i}{w_h h_i + w_s s_i},$$

其中 w_h、w_s 分別為勞動力及原料的對應權重，會由你的老闆指定。有了所有目前還有被納入考慮的訂單的 C/P 值後，我們在每一輪放棄 C/P 值最小的那張訂單。在判斷 $\frac{z_1}{y_1} \le \frac{z_2}{y_2}$ 是否為真時，要注意浮點數精確度問題。如果這四個數字都是整數，那可以改以 $z_1 y_2 \le z_2 y_1$ 來做判斷，就可以徹底避免浮點數精確度問題。如果有多張訂單的 C/P 值都同為最小，則選擇丟棄營收較低的訂單；若還是有多個選擇，則選擇丟棄編號較小的訂單。

在本題中，請使用指定演算法處理指定資料，決定應該接受哪些訂單，並且計算該訂單組合下的總營收。

說明 1：雖然不強制，但我們建議你先把輸入的資料存入對應的變數或陣列，接著寫一個函數讀取這些資料並且回傳（或印出）結果。具體來說，可以寫一個函數去找出應該被丟棄的訂單。首先，我們定義一個全域常數（global constant）

```
const int MAX_ITEM_CNT = 50;
```

這個函數可能長得像

```
int findWorst(int itemCnt, int weightH, int weightS, const int revenues[],
    const int labors[], const int supplies[], const int currentSol[]);
```

其中 itemCnt 代表題目給定的訂單數量，weightH 和 weightS 分別是題目給定的勞動力與原料權重，revenues、labors、supplies 皆為長度為 MAX_ITEM_CNT 的一維陣列，裡面每個元素分別為題目給定的各訂單營收、勞動力需求和原料需求，currentSol 是一個長度為 MAX_ITEM_CNT 的一維陣列，裡面每個元素存放著目前每一份訂單的接單狀況。這個函數可以去處理給定的資料，並且回傳應該丟棄的訂單編號。如果你對這些 const 的用途不熟，建議回頭複習課程影片。

說明 2：課堂上我們教過關於函數、模組化、變數生命週期、浮點數精確度、常數等等議題，我們用這一題把它們都串起來，大家可以透過這次練習更瞭解這些議題，當然也同時練習開發程式解決複雜的問題。

補充說明：定義與使用函數

由於本週的學習重點是函數，因此我們建議你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。一個標準的原則是：當你幫每個函數命名時，應該可以很貼切地把函數命名為那個函數會完成的事，而如果你發現因為這個函數做了太多事所以很難命名，那就是應該把這個函數再切成幾個小一點的函數了。

除此之外，本週也提到了變數的作用範圍（scope）與生命週期（life cycle），並且建議除非你完全知道自己在做什麼，不然不要使用全域變數（global variable），以下就是一個錯誤示範：

```
int itemCnt;
int weightH;
// ...
int findWorst(int currentSol[]);
```

雖然以上寫法或許仍然可以達成你想要做的任務，但是我們難以確保在執行多次的函數後，不應該被更動到的 global variables 會變成甚麼樣子。因此，為了加強程式的模組化（modularization），我們建議你使用 local variables，以達成「在每一個函數被呼叫時，只要看該函數的程式碼以及傳入函數的參數值，就完全可以判斷該函數的行為和回傳值」的設計。

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。在每個檔案中會有四列，第一列包含五個整數 n、w_h、w_s、H 和 S，第二列中包含 n 個整數 r_1、r_2 直到 r_n，第三列中包含 n 個整數 h_1、h_2 直到 h_n，第四列中包含 n 個整數 s_1、s_2 直到 s_n。已知 $1 \leq n \leq 50$、$w_h \in \{0, 1, ..., 5\}$、$w_s \in \{0, 1, ..., 5\}$、$1 \leq H \leq 50000$、$1 \leq S \leq 50000$、$1 \leq r_i \leq 10000$、$1 \leq h_i \leq 1000$、$1 \leq s_i \leq 1000$，並且不會出現同時 w_h 及 w_s 皆為 0 的情況。同一列的任意兩個整數之間被一個空白字元隔開。

讀入資料後，請按照題目的規定，印出 x_1、x_2 直到 x_n，代表應該接受哪些訂單，兩兩之間以一個逗號隔開，最後印出一個分號，再印出此訂單組合下的營收。舉例來說，如果輸入是

```
7 2 3 80 1000
1000 500 800 400 2000 600 2600
20 15 12 22 40 34 24
200 100 400 200 500 300 150
```

則輸出應該是

```
1,1,0,0,0,0,1;4100
```

請注意這個答案雖然不是最佳解，但我們還是印出它，畢竟這一題就是要用指定的演算法求解。如果輸入是

```
5 4 2 70 900
400 900 1400 300 500
13 35 28 10 21
300 200 500 600 100
```

則輸出應該是

```
0,1,0,0,1;1400
```

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。