程式設計（108-1）
作業七

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 ***11 月 19 日早上八點***。在你開始前，請閱讀課本的第 7.1–7.11 節和 22.9–22.13 節[1]。為這份作業設計測試資料並且提供解答的助教是莊日陞。

第一題

（20 分）承作業六第三題，在那題中我們假設工廠只有一條生產線，且生產線上只有一個工作站組成。而在這題中，我們將實作工廠有兩個工作站時的情形，也就是說，我們假設目前工廠只有一條生產線，此生產線由 2 個工作站組成，採購原料進來之後，要將原料投入工作站 1，得到工作站 1 產出的在製品（work-in-progress，WIP），再將工作站 1 產出的在製品投入工作站 2，得到完成品（最終產品）。工作站 i 中有 M_i 台機台。針對每一個工作站，我們假設工作站內的機台具有同質性，也就是說，不同工作站之間雖有不同的機台，但單一工作站內的機台都是相同的。

在工作站 i 中每開一個機台進行生產，都必須付出一定的開機成本 C_i^{ON}；開機後，一台機台每生產一公斤產品的生產成本為 C_i^D 元，且一台機台每日的產能上限為 B_i 公斤，代表機台一天最多能產出 B_i 公斤的產品；在製造過程中，當日投入原料當日就能通過兩個工作站並產出完成品。在本題我們也先不考慮製成率。

若工作站的產品成為存貨，每日每公斤的存貨成本則為 C_i^S 元，由於除了工作站 1 和工作站 2 的產品有庫存成本，原料購入後也有儲存幾日再投入工作站 1 生產的可能，我們可以把原料倉庫當作工作站 0，因此庫存成本共會有 3 項。另外，在規劃的第一天，每站皆有可能會有一些初始存貨 S_i（以公斤計算），初始存貨皆可投入生產，同理，初始存貨也會有 3 項。

已知需規劃的生產時段共有 T 天，規劃的目標為，在每日生產的最終產品數量（也就是生產線最後一個工作站的生產量）滿足當天需求量 D_t（以公斤計算）的條件下，找出能最小化總成本的最佳生產方案。總成本除上段中提到的開機成本 C_i^{ON}、單位生產成本 C_i^D 和存貨成本 C_i^S 外，顯然也必須包含原料採購成本。我們假設每日原料成本可能不同，以每公斤 P_t 元表示。

在本題中，我們將根據第 t 日工作站 i 的生產量 x_{it} 與原料採購量 w_t 去計算出總成本。給定生產與採購計畫，首先請先判斷這個計畫是否可行：完成品是否足以滿足每日需求量、各工作站之生產量是否滿足每日產能限制，以及原料和在製品是否足以供應生產計畫。如果可行，則計算總成本。

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。每個檔案中會有十三行：

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

- 第一行包含兩個整數 I 和 T（在本題中，I 必為 2）。
- 第二行至第六行皆包含 I 個整數，第二行依序為 M_1、M_2 直到 M_I，第三行依序為 C_1^{ON}、C_2^{ON} 直到 C_I^{ON}，第四行依序為 C_1^D、C_2^D 直到 C_I^D，第五行依序為 B_1、B_2 直到 B_I，第六行依序為 R_1、R_2 直到 R_I（在本題中，R_i 必為 1）。
- 第七行與第八行皆包含 $I+1$ 個整數，第七行依序為 C_0^S、C_I^S 直到 C_I^S，第八行依序為 S_0、S_1 直到 S_I。
- 第九行及第十行皆包含 T 個整數，第九行依序為 D_1、D_2 直到 D_T，第十行依序為 P_1、P_2 直到 P_T。
- 第十一行與第十二行包含一個 $I \times T$ 的矩陣 x_{it}，其中第十一行依序為 x_{11}、x_{12} 直到 $x_{1,T}$，其中第十二行依序為 x_{21}、x_{22} 直到 $x_{2,T}$。
- 第十三行為 w_1、w_2 直到 w_T。

以上每一行中，兩兩整數之間皆被一個空白隔開。已知 $I = 2$、$1 \le T \le 1000$、$1 \le M_i \le 100$、$1 \le C_i^{ON} \le 1000$、$1 \le C_i^D \le 1000$、$1 \le B_i \le 10000$、$1 \le R_i \le 5$、$1 \le C_i^S \le 1000$、$1 \le S_i \le 10000$、$1 \le D_t \le 10000$、$1 \le P_t \le 10000$、$1 \le x_{it} \le 10000$、$1 \le w_i \le 10000$。

讀入資料後，請先判斷給定的生產及採購計畫是否可行，若不可行則輸出 -1。若可行，則依序輸出總開機成本、生產成本、原料存貨成本、工作站一的完成品存貨成本、工作站二的完成品存貨成本與原料採購成本，任兩個數字之間用一個逗點隔開。

舉例來說，如果輸入是

```
2 3
3 3
1000 1500
5 4
2000 1500
1 1
4 10 25
1000 1000 500
1200 1500 800
10 40 30
600 800 700
800 1400 800
200 1500 300
```

則輸出應該是

```
7500,22500,11200,11000,2500,71000
```

如果輸入是

```
2 3
3 3
```

```
1000 1500
5 4
2000 1500
1 1
4 10 25
1000 1000 500
1200 1500 800
10 40 30
100 500 300
0 1000 0
0 1500 300
```

則輸出應該是

```
-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 20 分會根據程式運算的正確性給分。PDOGS 會直譯並執行你的程式、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）承第一題，但在此題我們將考慮製造流程中的製成率 R。我們將製成率 R 定義為投入 R 公斤的原料會得到一公斤的產出，且 R 為 1 到 5 之間的正整數。舉例來說，若 R 為 2，則代表兩公斤的原料會有一公斤的產出。

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。本題的輸入格式和第一題幾乎完全一樣，只差在第一題的 R_i 必為 1，而本題的 R_i 為 1 到 5 的整數。本題的輸出格式和第一題完全一樣。

舉例來說，如果輸入是

```
2 3
3 3
1000 1500
5 4
```

```
2000 1500
1 2
4 10 25
1000 1000 1500
1200 1500 200
10 40 30
800 900 300
700 500 200
500 200 300
```

則輸出應該是

```
7500,15600,2800,9000,25000,22000
```

如果輸入是

```
2 3
3 3
1000 1500
5 4
2000 1500
1 2
4 10 25
1000 1000 500
1200 1500 800
10 40 30
100 500 300
0 1000 0
0 1500 300
```

則輸出應該是

```
-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 20 分會根據程式運算的正確性給分。PDOGS 會直譯並執行你的程式、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

（60 分）關於資料探勘（data mining），幾種常見的任務包含關聯性（association）、分類（classification）、分群（clustering）。今天讓我們來學點關聯性分析。一個常見的例子發生在零售領域：當某個消費者買了若干品項並結帳，一個零售商能不能根據歷史交易記錄來猜，推薦哪個商品給消費者會得到最高的購買機率？這也是一個推薦系統（recommender system）問題。

讓我們具體地描述這個問題。假設我們有在銷售的品項（item）集合為 $I = \{1, 2, ..., n\}$，而歷史上發生過的交易（transaction）集合為 $T = \{1, 2, ..., m\}$。在交易 j 中，某消費者買了 $T_j \subset I$，亦即他從品項集合 I 中挑了一些東西買，這些東西的集合 T_j 是 I 的子集合[2]。為了簡單起見，讓我們假設每個品項都最多被買一個。現在一個新的消費者買了品項集合 $S \subset I$ 並且結帳了，我們想要在他付錢閃人之前（或是在網站上按下「結帳」之前），從他沒有買的品項集合 $I \setminus S$ 中挑一個商品推薦給他，而我們的任務是在 $I \setminus S$ 中找出他購買機率最大的那個商品。

讓我們來舉個例子。假設我的店裡賣五種 A、B、C、D、E 開店至今一共有 10 個人來買過，交易記錄如表 **??** 所示，也就是第一個人買了 D 和 E、第二個人買了 A 和 C 和 D，依此類推。若是用我們剛剛定義的參數來描述，我們有 $n = 5$、$m = 10$、$I = \{A, B, C, D, E\}$、$T_1 = \{D, E\}$、$T_2 = \{A, C, D\}$，依此類推。

A	B	C	D	E
0	0	0	1	1
1	0	1	1	0
1	0	0	1	0
1	0	0	1	0
0	0	0	1	1
0	1	1	1	0
1	1	0	0	1
1	0	0	1	0
0	0	1	1	1
0	0	1	1	0

表 1: 歷史交易記錄範例

所謂的「購買機率最大」，需要考慮幾件事。

- 首先，我們會考慮消費者一起購買某些商品的機率，畢竟如果某人已經買了義美牛奶，你應該不想要推薦他林鳳營牛奶，應該會想推薦他麵包（如果他沒有買麵包）。給定任何一個品項集合（itemset）S，我們可以計算其出現過的次數 $f(S)$，再除以總交易數 m，就是該品項一起被購買的機率，在關聯性分析的領域中我們將之稱為 *support*（支持度）。舉例來說，我們有 $f(C) = 4$、$f(D) = 9$、$f(\{C, D\}) = 4$，以及 $f(\{C, D, E\}) = 1$，因此他們各自的 support 就是 $\text{supp}(C) = 0.4$、$\text{supp}(\{C, D, E\}) = 0.1$，依此類推。
- 我們另外也要考慮已知消費者購買一些商品後，也購買另一些商品的條件機率。以我們的例子來說，買了 C 的人也買 D 的機率是 100%，但買了 D 的人也買 C 的機率只有 44.4%，因此對買 C

[2]理論上，T_j 是有可能等於 I，但這表示此消費者買了店裡所有的商品。由於這未免太不切實際，讓我們假設 T_j 不會等於 I。

的人推薦 D 成功率較高，對買 D 的人推薦 C 成功率就比較低了。給定前提品項集合（antecedent itemset）X 跟結果品項集合（consequent itemset）Y，我們用 $f(Y|X)$ 表示買 X 的人也買 Y 的次數，再除以有購買 X 的交易次數 $f(X)$，就得到被稱為 *confidence*（信賴度）的條件機率 $\text{conf}(Y|X)$。在我們的例子中，$\text{conf}(D|C) = 1$、$\text{conf}(C|D) = \frac{4}{9}$，以及 $\text{conf}(D|\{C, E\}) = 1$。

有了 support 和 confidence 的觀念後，要根據某消費者的購買品項集合 S 來做推薦，就不是那麼沒有頭緒了。為了簡單起見，讓我們假設我們只想推銷一個單品（而不是一個集合），那麼我們要做的就是兩件事：

1. 對一個在集合 $I \setminus S$ 中的商品 i，計算其與 S 一起被購買的 support $\text{supp}(\{i\} \cup S)$。根據一個給定的目標值 s，如果 $\text{supp}(\{i\} \cup S) \geq s$ 就保留 i，反之則不考慮推薦 i。
2. 對於通過第一步驟篩選的品項，一一考慮購買 S 後也購買該品項的機率，也就是 $\text{conf}(i|S)$，然後挑出 confidence 最大的那個品項做推薦。

請想想我們為什麼要同時考慮 support 和 confidence。考慮 confidence 是直觀的：如果某人買了 S，而且以往有一堆買了 S 的人也買 i，那推薦他買 i 的成功率自然不低。但於此同時 S 和 i 的 support 也是需要注意的。如果 support 太低，那麼這個高 confidence 很可能就是個巧合，只有在某個組合有高 support 的情況下，我們才真的相信它們的 confidence 是有用的。

在本題中，你將會被給定品項與歷史交易資訊。接著你會被給定一筆交易中購買的品項集合，以及 support 必須夠高的門檻值。你的任務是根據上述規則，找出應該推薦的品項。如果有複數個品項都通過 support 的要求並且同樣有最高的 confidence，就推薦編號最小的那個。如果沒有任何品項可以推薦（因為都不滿足 support 門檻），就不要推薦任何東西。

特殊要求：動態陣列

如果要實做這一題要求的任務，一個很直觀的想法是：如果總品項數為 n、總交易次數為 m，那就用一個 $m \times n$ 的靜態二維陣列來存交易記錄，然後再寫一些迴圈、函數之類的東西去反覆翻攪這個靜態二維陣列，去算出我們要的答案。

這樣的想法固然沒錯，但這只有在 n 和 m 夠小的時候才比較適合。實務上一個零售店通常有成千上萬個品項，如果是電子商務網站更是有可能有幾十萬、上百萬，要分析的交易筆數同樣也可能幾百萬、上千萬。假設各一百萬吧，如果建一個 1000000×1000000 的二維陣列，會非常浪費空間[3]。除此之外，使用靜態陣列意味著貴店不能增加品項，能容許的交易次數也有其上限，這些都會讓你的程式很不具彈性。

那怎麼辦呢？好消息是，絕大部分的消費者在一次交易中，通常只買頂多十幾個品項，因此如果用 $m \times n$ 的二維陣列來存交易記錄，就會有非常非常多的陣列元素都是存 0，這些 0 某方面來講其實就跳過不要存就好了。因此在這一題，我們要來練習用動態二維陣列儲存歷史交易資料[4]。

在我們即將實作的陣列中，我們依然會有 m 列，一列代表一次交易，但現在我們在一列中將只儲存該次交易中被購買的品項的編號。因為兩次交易中購買的品項數未必相同，所以我們的每一列都會是

[3]認真算一下，如果這二維陣列存的是整數，那 4×10^12 byte 可是 4 TB 之多，幾乎沒有誰的個人電腦有 4 TB 的記憶體

[4]要是真的有幾千萬個品項和幾百億個交易，我們這門課介紹的任何資料結構和演算法都不足以針對推薦問題做良好的運算。你肯定需要更好的資料結構和更好的演算法，而這可能是你得要去修更進階的課的動機吧。

一個一維動態陣列，長度恰好為該次交易被購買的品項數。換言之，我們的「表格」將會「各列長度不一」，而且「表格」的列數是可以持續增加的（不過在這一題中，我們不會要求你做這件事）。

本題有兩個具體要求：

1. 你必須使用二維動態陣列來儲存交易資料，而且兩個維度（列跟欄）都必須是動態的。
2. 你必須寫一個函數

```
void setTransactions(int** trans, int* itemCnt, int m);
```

其中 `trans` 是儲存歷史交易資料的二維動態陣列、`itemCnt` 是儲存每筆交易中各購買了幾個品項的一維動態陣列、`m` 是由測試資料讀入的歷史交易資料筆數。你的 main function 必須呼叫此函數一次，在此函數中讀入測試資料中的歷史交易資料，並將相關資訊記錄在 `trans` 和 `itemCnt` 中，以便後續程式做處理和運算。

為了強迫你這麼做，在這一題我們會對你的程式在 PDOGS 上的記憶體用量有所限制，讓你如果用 $m \times n$ 的二維靜態陣列就會超出記憶體的限制，因此在部份測試資料得不到分數。當然如果你真的應付不了動態陣列，那寫靜態陣列也是可以取得一部分規模較小的測試資料的分數的，不過，動態記憶體配置是會在未來持續被使用的功能，請大家還是盡可能地練習吧！

輸入輸出格式

系統會提供 20 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、m 和一個三位小數 s，分別代表總品項數、總交易數和 support 門檻。品項編號為 1、2、3 一直到 n，而交易編號為 1、2、3 一直到 m。在第二列至第 $m+1$ 列中，第 $j+1$ 列存放 k_j+1 個介於 1 到 n 的不重複整數，其中第一個數字 k_j 代表歷史上第 j 筆交易所購買的品項數，後面 k_j 個數字則是品項集合 T_{j-1} 中的品項編號。第 $m+2$ 列中也有 $k_{m+1}+1$ 個介於 1 到 n 的不重複整數，代表現在要被推薦的消費者購買的品項數以及品項集合 S 中的品項編號。每一列中的兩個數字都用一個空白鍵隔開。

在前 10 筆測試資料中，我們已知 $1 \le n \le 20$ 以及 $1 \le m \le 500$。在後 10 筆測試資料中，我們已知 $1 \le n \le 20000$ 以及 $1 \le m \le 50000$。

根據規則找出應該推薦的品項後，請依序輸出該品項的編號、該品項與 S 共同出現的交易次數（亦即 support 乘以 m），以及 S 出現的交易次數（亦即給定 S 會購買該品項的 confidence 的分母）。各數字用一個逗點隔開。如果沒有任何品項可以推薦（因為都不滿足 support 門檻），就不要印出任何東西。

舉例來說，如果輸入是

```
5 10 0.100
2 4 5
3 1 3 4
2 1 4
2 1 4
2 4 5
3 2 3 4
3 1 2 5
2 1 4
```

```
3 3 4 5
2 3 4
1 4
```

則輸出應該是

```
1,4,9
```

請注意雖然品項 3 的 support 也達到門檻、confidence 也並列最高，但此時我們推薦編號最小的商品 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***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。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，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）。助教還會看你有沒有照規定寫我們指定的函數。請寫一個「好」的程式吧！

第四題（加分題）

（20 分）有一個關於寫作的建議（特別是針對初學者）是「句子不要寫太長」。因此，計算一個句子平均含有幾個字，可能有其用途。本題將給定一段由英文字母、數字、標點符號和空白組成的段落，這次要計算的是此段落中有幾個句子，以及每個句子平均有幾個單字（無條件捨去至整數位）。本題的標點符號包含有「`,.:;!?`」，它們被定義為句子的分隔符號。在兩個句子的分隔符號中間就是一個句子，即使只有空白或根本沒有字元也算一個句子[5]。但如果給定段落的最後一個字元是標點符號，我們並不說最後面有一個長度為 0 的句子；如果給定段落的最前面一個字元是標點符號，我們也不說最前面有一個長度為 0 的句子。

舉例來說：

- 「Among the ten students, two got zero.」會被判定為兩個句子，前句有 4 個單字，後句有 3 個單字，則平均下來 $(4+3)/2 = 3.5$，無條件捨去至整數位，可得 3。
- 「Hi! I am 22 years old. I am a student. I live in Taipei」此段落有 4 個句子（即使最後沒有句點），單字分別有 1、5、4、和 4 個，因此平均每句有 $(1+5+4+4)/4 = 3.5$，無條件捨去至整數位，可得 3 個單字。

[5]這個和實務當然有一點落差，例如刪節號「...」理論上並不造成兩個空的句子，但在本題中為了簡單起見，我們就這麼定義。

- 若是「A...Z...」則有 6 個句子，分別有 1、0、0、1、0、0 個單字，平均無條件捨去後每句有 0 個單字。請注意最後一個句點後面並沒有一個句子。
- 若是「.A.Z」則有 2 個句子，分別有 1、1 個單字，平均無條件捨去後每句有 1 個單字。請注意第一個句點前面並沒有一個句子。
- 若是「I like programming」則有 1 個句子、3 個單字，平均無條件捨去後每句有 3 個單字。

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，第一行會有一段由英文字母、數字、標點符號（,.:;!?）和空白所組成的段落。已知一個段落的字元不少於 1 個、不超過 1000 個、一個單字的字元不超過 50 個，總共不超過 1000 個單字。讀入這些資訊後，請依照題目指定的規則，輸出該段落的句子數，以及平均每句有幾個單字（無條件捨去到整數位），兩者以一個逗號隔開。

舉例來說，如果輸入是

```
Among the ten students, two got zero.
```

則輸出應該是

```
2,3
```

如果輸入是

```
A...Z...
```

則輸出應該是

```
6,0
```

如果輸入是

```
I like programming
```

則輸出應該是

```
1,3
```

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 20 分會根據程式運算的正確性給分。PDOGS 會直譯並執行你的程式、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。