程式設計（105-2）
作業五

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。請以英文或中文作答。

這份作業的截止時間是 ***2017 年 3 月 27 日凌晨一點***。在你開始前，請閱讀課本的第 5.20–5.20 和第 19 章[1]。為這份作業設計測試資料並且提供解答的助教是林敬傑（Jack Lin）。

第一題

（20 分，每小題 10 分）「C 幾取幾」這個排列組合概念，大家應該都學過，讓我們精確地問：從 n 個不同的整數中任意取出 m 個，有幾種不同的組合（combination）呢？若我們用 $C(n, m)$ 表示這個組合個數，則我們小時候學過公式

$$C(n, m) = \frac{n!}{m!(n-m)!},$$

前提是 $n \geq m$。

(a) 敬傑寫了一個函數來計算 $C(n, m)$：

```
int fact(int n)
{
  int prod = 1;
  for(int i = 2; i <= n; i++)
    prod *= i;
  return prod;
}

int combiBad(int n, int m)
{
  return fact(n) / (fact(m) * fact(n - m));
}
```

其中 `fact(n)` 會回傳 $n!$，`combiBad(n, m)` 則回傳 $C(n, m)$。

雖然概念完成正確，計算例如 $C(10, 5)$、$C(12, 6)$ 也會得到正確答案，但計算 $C(15, m)$ 的時候就不對了，不論 m 是多少。請告訴敬傑為什麼會這樣。

(b) 佩蓉幫敬傑改寫了這兩個函數：

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

```
int multi(int s, int t)
{
  int prod = 1;
  for(int i = s; i <= t; i++)
    prod *= i;
  return prod;
}

int combiRep(int n, int m)
{
  if(2 * m > n)
    return multi(       ,       ) / multi(       ,       );
  else
    return multi(       ,       ) / multi(       ,       );
}
```

但是她故意把 `combiRep()` 裡呼叫 `multi()` 的參數擦掉了[2]。請幫敬傑完成 `combiRep()`，並且說明為什麼 `combiRep()` 是正確的演算法，而且相較於 `combiBad()` 可以接受更大的輸入（例如 $C(15,7)$、$C(18,8)$ 等等）。

第二題

（20 分）承第一題，昱賢覺得佩蓉寫的程式還可以再改進，畢竟 $C(18,9)$ 它就算不出來了。他說：「如果要從 n 個東西之中取出 m 個，那其實所有組合可以被歸納為兩大類。首先，我們先不要看第 n 個東西，先去前 $n-1$ 個東西中取出 m 個；接著我們強迫自己一定要取到第 n 個東西，所以我們在前面 $n-1$ 個東西中取出 $m-1$ 個。這兩大類組合的個數合計，就是 $C(n,m)$ 了，也就是說

$$C(n,m) = C(n-1,m) + C(n-1,m-1)。$$

(a)（10 分）請根據昱賢的想法，寫一個遞迴函數，其 header 如下：

```
int combiRec(int n, int m)
```

而其行為是當依序傳入 n 跟 m，則若 $n < m$ 則回傳 -1，若 $n \geq m$ 則回傳 $C(n,m)$。請只使用 `int` 跟 `bool`，不要用其他基本資料型態。

(b)（5 分）請試試看 $C(30,15)$，你會發現只有昱賢的函數可以算出正確答案，佩蓉跟敬傑的都不行。請說明為什麼昱賢的演算法能克服佩蓉跟敬傑無法克服的 overflow（溢位）問題。

(c)（5 分）昱賢的演算法當然也有其極限，例如 $C(35,15)$ 它就要算好幾秒，而且算出來的也不對。請解釋為什麼昱賢的演算法的時間複雜度高過敬傑的跟佩蓉的。

[2] 人怎麼這麼壞 ?! XD

第三題

（60 分）關於資料探勘（data mining），有一種說法是它有三大類型的任務：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[3]。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 1 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

[3]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。

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、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。我們已知 $1 \leq n \leq 20$ 以及 $1 \leq m \leq 500$。每一列中的兩個數字都用一個空白鍵隔開。

根據規則找出應該推薦的品項後，請依序輸入該品項的編號、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）。各數字用一個空白鍵隔開。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。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！