程式設計（107-1）
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/judge/`）為三題各上傳一份 C++ 原始碼（以複製貼上原始碼的方式上傳）；第三題的其中 20 分是加分題。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 ***10 月 16 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林翰伸。

第一題

（20 分；每題 4 分）針對以下五題是非題，我們會使用 PDOGS 自動批改，因此請寫一個 C++ 程式，內容就是先讀入一個整數，若讀入的數字為 i，則印出第 i 小題的答案，若為是則印出 1、若為否則印出 0。舉例來說，如果題目只有四題，且你認為答案依序是是、否、是、是，則你上傳的程式碼應該是

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

int main()
{
  int problem = 0;
  cin >> problem;
  if(problem == 1)
    cout << 1;
  else if(problem == 2)
    cout << 0;
  else if(problem == 3)
    cout << 1;
  else
    cout << 1;

  return 0;
}
```

PDOGS 會餵給你的程式的，一定是 1、2 直到 10 這十個整數。有別於作業中一般的程式題，本題在你上傳程式碼時，測試資料是還沒有放上 PDOGS 的，助教會等作業截止後才上傳測試資料（和答案）到 PDOGS 並重新批改此題。換言之，你上傳程式碼時是不會顯示你得幾分的，更不會顯示你對或錯哪些筆測試資料。你會看到你得 0 分，但此數字在助教重新批改之後就會被更新成正確的分數了。

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

以下題目如果沒有特別指名，請用 C++ 為基準作答。若你看到一段程式碼，請假設他們是被寫在一個有良好 include 敘述、using namespace 敘述的程式的結構正確的 main function 裡面。

(a) 每一個型態為 `char` 的字元，都是被編碼成整數。

(b) 一個二維陣列，可以有辦法直接取出其中一列，但沒有辦法直接取出其中一欄。

(c) 宣告一個二維陣列時，可以不用寫第一個維度的長度。

(d) 以下兩個變數

```
int a[50][100] = {0};
int b[100][50] = {0};
```

`a[2][3]` 距離 `a` 的位移（offset）跟 `b[3][2]` 距離 `b` 的位移是一樣的。

(e) 在宣告函數時，如果把函數中的陣列參數（parameter）設成 `const`，則此函數的執行過程將不能修改該陣列的內容，亦即函數執行完畢後，該陣列的內容必然跟執行前一模一樣。

小提醒：在 PDOGS 上面讓大家繳交此題的地方，會有兩組「與上面正式要計分的題目完全無關的」範例輸入輸出，純粹是用來讓大家確認自己那個被批改的 `if-else` 程式是可以被正確執行的。請確認你的程式在針對範例輸入輸出做撰寫後，能讓你在這一題得到「Accepted」，接著再去針對要計分的題目把你的正確答案寫上去然後繳交。當然，即使你曾經看到「Accepted」，也不代表你繳交的題目在這題已經得到滿分了。

第二題

（60 分）如大家所知，任何基本資料型態都有其被分配的記憶體空間。例如以 C++ 來說，在現在大部份的編譯器上 `int` 是 4 位元組、`double` 是 8 位元組。以整數來說，4 位元組讓我們可以表現大約 -2^{31} 和 2^{31} 間的整數，也就是大約負 21 億到正 21 億之間。如果想要儲存的整數超過這個範圍，就不能用 `int` 存了。但就算是 `long int` 或 `double`，總也是有個上限。

如果必須存更大（或更小）的整數，字元陣列是一個選項。以加法為例，我們可以在使用者輸入兩個很大的整數時，先將它們存入兩個字元陣列中，然後再用大家小時候學過的加法，從個位數開始一次處理一個位元，若有進位就進到下一位元，如此一直到所有位元都被處理好為止。當然，你算出來的東西也應該被存進另一個字元陣列，才能被好好保存。

為了讓本題更困難一點，我們在本題將要求你做大數的加減法，而且這些數字都是金額，因此輸出時要在最前面加上錢字號，並且在正確的地方加入逗號當千分位號。請寫兩個函數

```
void bigMoneyAdd(char m1[], char m2[], char sum[], int n);
void bigMoneySub(char m1[], char m2[], char diff[], int n);
```

去完成這個任務，其中 `m1` 和 `m2` 是兩個代表大整數的字元陣列，`sum` 和 `diff` 這兩個字元陣列，分別是裝 `m1 + m2` 的值或 `m1 - m2` 的值。`n` 是這些陣列的長度（因此你的大數運算依然是有位元限制的，不過至少這個限制可以很寬很寬了）。為了讓本題簡單一點，你可以假設所有數字都是非負整數，包含 `diff` 在內。

輸入輸出格式

系統會提供一共 30 組測試資料，每組測試資料裝在一個檔案裡。每個檔案中有三行，其中第一行是一個正整數 n，是等一下會出現的整數及其和與差的位數的上限，第二行和第三行則各是一個非負整數 m_1 和 m_2。讀入這些資料後，請先輸出以正確格式帶錢字號和千分位符號的和 $m_1 + m_2$，接著輸出以正確格式帶錢字號和千分位符號的差 $m_1 - m_2$。已知 $1 \le n \le 100$、$0 \le m_1 \le 10^n - 1$、$0 \le m_2 \le 10^n - 1$、$0 \le m_1 + m_2 \le 10^n - 1$、$0 \le m_1 - m_2 \le 10^n - 1$。

舉例來說，如果輸入是

```
25
9876543210000000000
876123456000000000
```

則輸出應該是

```
$10,752,666,666,000,000,000
$9,000,419,754,000,000,000
```

如果輸入是

```
1
4
4
```

則輸出應該是

```
$8
$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。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）。請寫一個「好」的程式吧！

第三題（加分題）

（40 分）有一個業務員，要從其公司去拜訪 n 個顧客。我們把該公司編號為 0，n 個顧客的家則依序編號為 1、2 到 n。我們用 (i,j) 來表示地點 i 和 j 之間的路段，並且說在路段 (i,j) 上移動要花的最短時間為 d_{ij} 分鐘。我們現在要幫此業務員決定送貨路線，以最小化這趟拜訪要花的總時間。這位業務員基本上是需要徒步的，而 d_{ij} 是他徒步要花的時間，但公司也有給他足以在兩個地點之間搭一趟公車的經費，若他在地點 i 跟 j 之間搭公車，則該移動時間將變成 d'_{ij} 分鐘。請注意他最多只能搭一趟公車。

在資訊科學（computer science）與作業研究（operations research）領域中，學者們普遍認為這個問題是「困難的」[2]，也就是說只要要拜訪的顧客夠多，那即使不考慮公車這回事，任何演算法都無法在合理的時間內求得最佳解。因此，本題並不要求大家求得最佳解。我們將給大家一個簡單的演算法，要求你按照這個演算法去執行，以求得一個可行解。

我們使用如下演算法來規劃拜訪路線。因為一共有 $n+1$ 個點，我們一共有 $\binom{n+1}{2} = \frac{(n+1)!}{2!(n-1)!}$ 個路段，也就是這麼多個搭公車的選擇。針對每一個路段，我們將問自己「如果我確定只在此路段搭公車，並且使用『每次都前往移動時間最短的尚未拜訪的顧客』的演算法，那麼我要花多少時間」，然後得到一個路線與相對應的時間。請注意這個路線有可能完全不會搭公車。得到 $\binom{n+1}{2}$ 個路線與步行、搭車規劃後，我們再從中挑出最好的一個。

舉例來說，假設我們共有 4 個顧客，且所有的步行時間 d_{ij} 可以被表示成一個矩陣（編號從 0 開始）

$$D = \begin{bmatrix} 0 & 4 & 5 & 7 & 9 \\ 4 & 0 & 6 & 2 & 3 \\ 5 & 6 & 0 & 4 & 2 \\ 7 & 2 & 4 & 0 & 5 \\ 9 & 3 & 2 & 5 & 0 \end{bmatrix},$$

表示 $d_{01} = 4$、$d_{13} = 2$，依此類推。假設 $D' = D$，亦即在每個路段搭公車都不會省下時間，則我們的演算法自然也不需要考慮搭公車這個選項，那麼我們根據演算法，將依序拜訪顧客 1、3、2、4，五段移動時間依序是 $d_{01} = 4$、$d_{13} = 2$、$d_{32} = 4$、$d_{24} = 2$，以及 $d_{40} = 9$，而總出訪時間是 $4+2+4+2+9 = 21$ 分鐘。這是我們的「無公車演算法解」。

現在，假設車行時間矩陣是

$$D' = \begin{bmatrix} 0 & 4 & 5 & 1 & 6 \\ 4 & 0 & 6 & 2 & 3 \\ 5 & 6 & 0 & 4 & 2 \\ 1 & 2 & 4 & 0 & 5 \\ 6 & 3 & 2 & 5 & 0 \end{bmatrix},$$

表示在路段 $(0,3)$ 和 $(0,4)$ 之間搭公車能省下時間，讓我們來找「有公車演算法解」。雖然我們一共有 $\binom{5}{2} = 10$ 個路段可以搭公車，但若我們決定在其中八個不會省下時間的路段搭公車，都會得到同一個「無公車演算法解」，拜訪順序 1、3、2、4，總時間 21 分鐘。若我們決定在路段 $(0,4)$ 搭公車，我們會發現 $d'_{04} = d'_{40} = 6$，還是會得到拜訪順序 1、3、2、4，總時間 18 分鐘。最後，若我們決定在路段 $(0,3)$ 搭公車，我們會發現 $d'_{03} = d'_{30} = 1$，我們會改成採用新的拜訪順序 3、1、4、2，總時間 $1+2+3+2+5 = 13$ 分鐘。

在本題中，我們要請你實做這個演算法。很顯然地，要找「有公車演算法解」，就要先會找「無公車

[2]學理上我們稱之為「NP-hard」

演算法解」，而你找「有公車演算法解」的演算法，將呼叫你找「無公車演算法解」的演算法最多 $\binom{n+1}{2}$ 次。因此你顯然應該把找「無公車演算法解」的演算法寫成一個函數，例如

```
void noBusTour(int d[][MAX_CUS_CNT + 1], int tour[], int n);
```

之類的，其中 d 是移動時間矩陣、MAX_CUS_CNT 是 n 最大可能的值，tour 是存「無公車演算法解」的陣列，n 是顧客數。當然你也可以有別的設計，但不管怎樣，希望大家都同意寫函數顯然會讓程式更模組化、更清晰易懂、更具彈性與擴充性，也更能反覆利用自己寫的程式碼。

輸入輸出格式

系統會提供一共 20 組測試資料，每組測試資料裝在一個檔案裡。每個檔案含有 $n+2$ 行，第一行含有一個正整數 n，第二行到第 $n+2$ 行中的第 i 行依序含有 $d_{i-2,0}$、$d_{i-2,1}$ 到 $d_{i-2,n}$，接著 $d'_{i-2,0}$、$d'_{i-2,1}$ 到 $d'_{i-2,n}$。已知 $1 \le n \le 100$、$1 \le d_{ij} \le 30$、$1 \le d'_{ij} \le 30$。一行中任兩個數字用一個空白字元隔開。讀入這些值之後，請依照題目指定的演算法找出一組解，並輸出這組解的總出訪時間。

舉例來說，如果輸入是

```
4
0 4 5 7 9 0 4 5 1 6
4 0 6 2 3 4 0 6 2 3
5 6 0 4 2 5 6 0 4 2
7 2 4 0 5 1 2 4 0 5
9 3 2 5 0 6 3 2 5 0
```

則輸出應該是

```
13
```

針對這個題目，你可以使用任何方法。這一題的 40 分會根據程式運算的正確性給分，一筆測試資料佔 2 分。