程式設計（105-2）
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/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 13 日凌晨一點***。在你開始前，請閱讀課本的第 6.1–6.4 節[1]。第 6.7 和 6.8 節也有幫助。如果你想知道在 C++ 裡面的「函數」是什麼，你可以先讀讀我們下週會教的第五章（的最前面幾頁）。為這份作業設計測試資料並且提供解答的助教是楊佩蓉（Sophie Yang）。

第一題

（20 分）請回答下列各小題：

(a)（10 分）請考慮下面這個程式：

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

int main()
{
  int a = 99;
  int b = 75;
  float r = static_cast<float>(a) / b;
  cout << r << " " << r * b << "\n";

  if(r * b != a)
    cout << "WOW!\n";

  return 0;
}
```

請寫出執行後會看到的輸出結果，並用自己的話解釋為什麼輸出結果會長那樣，特別是為什麼 `r * b` 不等於 `a`。

(b)（5 分）承 (a) 小題，請修改那個程式並使用 `setprecision()` 函數來證明自己在 (a) 小題的論點。

(c)（5 分）請考慮下面這個程式。

[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;

int main()
{
  bool hasBadRatio = false;
  for(int num = 1; num < 100; num++)
  {
    for(int deno = 1; deno < 100; deno++)
    {
      float r = static_cast<float>(num) / deno;
      if(num != deno * r)
      {
        cout << num << " " << deno << " " << r << "\n";
        hasBadRatio = true;
        break;
      }
    }
    if(hasBadRatio == true)
      break;
  }

  return 0;
}
```

請用自己的話說明這個程式在做什麼，特別是 `hasBadRatio` 這個 Boolean 變數以及那兩個 `break` 的用途。

重點說明：我們把如上面這個程式中的 `hasBadRatio` 這種變數叫做 flag（旗標），就是「如果某件事發生了，就趕快把旗子舉起來，這樣其他人看到之後，就去幹嘛幹嘛。」請試著體會 flag 的用途和使用時機（並且據此回答這一題）。你可以想想，但是不用回答：如果不使用這個 flag，要怎麼一次跳出兩層迴圈？如果很麻煩，那就是使用 flag 的理由。

第二題

（20 分；一小題 10 分）敬傑寫了一個程式，希望可以讓使用者輸入兩個整數 n 和 m 之後，讓程式輸出「在 n 個人之中取 m 個人的組合數」，例如 4 取 2 有 6 種組合、6 取 3 有 20 種組合之類的。他寫的程式碼如下：

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

int main()
{
  int n = 0, m = 0;
  cin >> n >> m;

  if(n > m)
  {
    int num = 1;
    for(int i = 1; i <= n; i++)
      num *= i;
    int de1 = 1;
    for(int i = 1; i <= m; i++)
      de1 *= i;
    int de2 = 1;
    for(int i = 1; i <= n - m; i++)
      de2 *= i;

    cout << num / (de1 * de2);
  }

  return 0;
}
```

(a) 佩蓉跟他說這個程式在 n 不大的時候可以算出正確解答，但若 n 太大就不行了（你可以輸入 $n = 13$、$m = 3$ 試試）。請幫佩蓉跟敬傑用白話解釋問題出在哪裡，並且修改敬傑的程式碼去加入一些 `cout` 敘述，以證明你的論點。

(b) 請修改敬傑的程式，在「不使用除了 `int` 以外的資料型態」的前提下，可以有比較多的機會計算出正確的答案。你的程式碼至少應該要在 $n = 13$、$m = 3$ 的時候算出正確答案。

重點說明 1：我們當然可以把 `num` 的型態改成 `long long int`，這樣就可以解決一些問題，但這不是這一題要你做的事。請從「演算法」的角度，也就是程式運算的流程和邏輯的角度去思考，看看如何修改程式運算的步驟和順序，來讓依然只用到 `int` 的程式可以有更多機會算出正確答案。

重點說明 2：不管怎樣，在只宣告整數變數的要求下，這個程式都不可能可以對所有輸入都計算出正確的結果。雖然不要求，但你可以試著想想看如何可以大幅度地擴充這個程式可接受的參數範圍。小提示一下：用陣列可能是個好主意！

第三題

（60 分）有一家電信公司正在研擬一個新服務區域的無線基地臺設置計畫[2] 在這個區域裡，一共有 n 個城鎮，編號為 1、2、$\cdots$、n，而城鎮 i 的人口數是 P_i。公司將此區域以一公里為單位，畫出了一個二維座標系，並且以 (x_i, y_i) 表示城鎮 i 的位置。換句話說，城鎮 i_1 跟城鎮 i_2 之間的距離是

$$\sqrt{(x_{i_1} - x_{i_2})^2 + (y_{i_1} - y_{i_2})^2}$$

公里。如果一個基地臺跟一個城鎮的距離在 d 公里以內，我們就說這個基地臺可以「覆蓋」這個城鎮，也就是這個城鎮的人可以收得到強度足夠的從該基地臺發出的訊號。公司預計在此區域的 n 個城鎮中挑選 p 個城鎮設置基地臺，以求能覆蓋最多的人口數。

佩蓉在這家電信公司工作，負責挑出這 p 個城鎮。為此，佩蓉設計了一個貪婪演算法（greedy algorithm）。首先在所有城鎮中，佩蓉去找出「如果蓋在這裡，將可以覆蓋最多人」的城鎮，然後設一個基地臺在那裡。現在佩蓉還能再設置 $p-1$ 個基地臺，所以她如法泡製，在所有還沒有基地臺的城鎮中，找出「如果蓋在這裡，將可以覆蓋最多還沒被覆蓋的人」的城鎮，設一個基地臺在那裡，然後繼續如此直到挑出 p 個城鎮去設置基地臺為止。如果在任一時刻遇到有兩個以上的城鎮可以被選，就選編號較小的那個。

舉例來說，假設有八個城鎮的分佈如圖 1 所示，而他們的人口數 P_i 由城鎮 1 至城鎮 8 分別是 10、15、10、20、20、25、15 和 10。假設覆蓋半徑 $d = 3$，要如何用佩蓉的演算法找出 $p = 3$ 個城鎮來設置基地臺呢？首先，我們要對每個城鎮都計算「如果蓋在這裡，將可以覆蓋多少人」，例如若是蓋在城鎮 1，可以覆蓋城鎮 1 和 7 的共 25 人，蓋在城鎮 2 則可以覆蓋城鎮 2、3 和 8 的共 35 人，依此類推。我們很快可以發現蓋在城鎮 8 可以覆蓋共 55 人是最多的（城鎮 4 與 8 的距離恰好是 3，剛剛好可以被覆蓋），所以我們會將第一個基地臺設置在城鎮 8，如圖 2 所示。

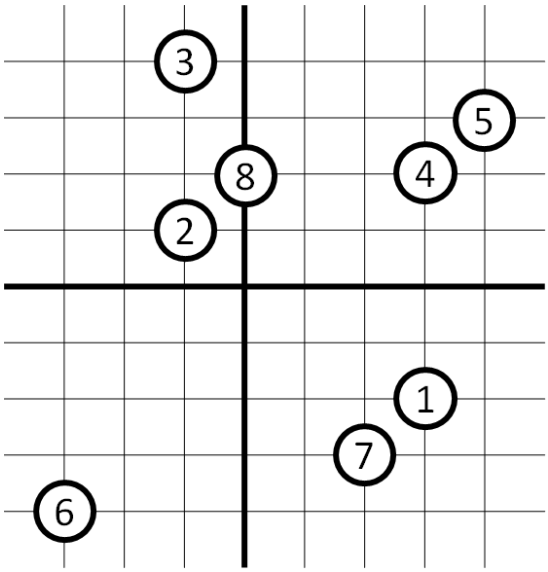

Figure 1: 八個城鎮

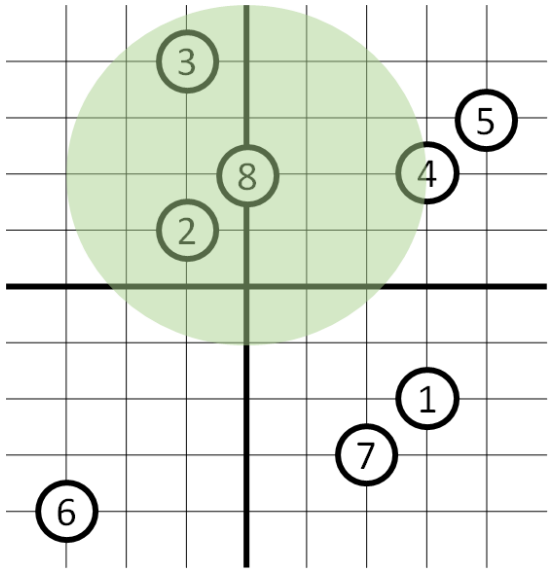

Figure 2: 第一次選取

接著我們對城鎮 1 到 7 再計算「如果蓋在這裡，可以覆蓋多少還沒被覆蓋的人」。舉例來說，城鎮 2 跟 3 此時能再多覆蓋的人數是 0 了，因為這兩個城鎮都已經被城鎮 8 的基地臺覆蓋了；城鎮 4 跟 5

[2]如果你有修 105-1 學期的「商管程式設計」，對，這一題跟那時的某份作業的某一題是一樣的。不過你也不用太開心；當你要用 C++ 重寫一次，當時寫的 Python 程式碼派不上什麼用場的。

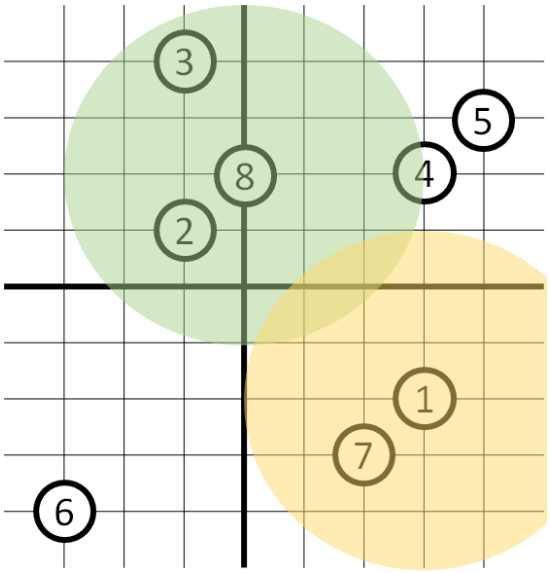

Figure 3: 第二次選取

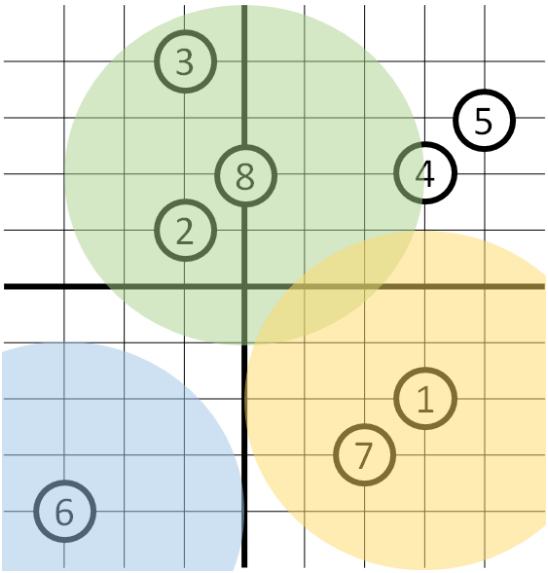

Figure 4: 第三次選取

則都還能再多覆蓋城鎮 5 的 20 人。我們很快可以發現，城鎮 1、6 跟 7 都各可以再多覆蓋 25 人，因此我們選擇在城鎮 1（編號最小）設置一個基地臺，如圖 3 所示。請注意雖然城鎮 4 可以覆蓋 50 人，但我們不會選它，因為它能「再多覆蓋」的人只有 20 人。

最後，我們再針對城鎮 2 到 7 計算「如果蓋在這裡，可以覆蓋多少還沒被覆蓋的人」。很快地我們會發現城鎮 6 的 25 人是最大的，因此將最後一個基地臺設置在城鎮 6，如圖 4 所示。總共我們覆蓋了 105 人（只有城鎮 5 的人沒有被覆蓋到）。

有一些情況下，我們執行上面的演算法後，只蓋了不到 p 個基地臺，就已經覆蓋所有的城鎮了。如果是這樣，在演算法的最後，我們就再去把基地臺由編號最小而還沒有基地臺的城鎮開始蓋，直到蓋滿 p 個為止。

在本題中，請幫佩蓉實做這個演算法。

補充說明：Pseudocode

在描述一個演算法時，我們經常喜歡以 ***pseudocode*** 呈現[3]。所謂 pseudocode，就是把演算法的步驟以程式邏輯的方式編排與呈現，而不要只是寫成一段文字。寫 pseudocode 經常是「從文字描述的演算法到以程式碼實做的程式」之間的一個過程，也就是從「設計」到「實做」的一個中間步驟。

以佩蓉的演算法為例，最初步的 pseudocode 可以長這樣：

```
for 基地臺 j 從 1 到 p
  在沒有基地臺的城鎮中找出「若蓋在這裡可以覆蓋最多還沒被覆蓋的人」的
  設基地臺 j 在該城鎮
```

可以看到其實就是上面那段文字描述，只是已經有一點迴圈的階層架構了。更進一步地，我們可以寫：

[3]中文翻譯成「虛擬碼」，聽起來挺怪的。

```
for 基地臺 j 從 1 到 p
  maxCover = -1
  maxCoveringTown = 0
  for 城鎮 i 從 1 到 n
    if 城鎮 i 沒有基地臺
      cover = 如果蓋在這裡，將可以覆蓋的還沒被覆蓋的人的人數
      if cover > maxCover
        maxCover = cover
        maxCoveringTown = i
  設基地臺 j 在城鎮 maxCoveringTown
```

如果我們再更進一步：

```
宣告一個空的基地臺清單
for j 從 1 到 p
  maxCover = -1
  maxCoveringTown = 0
  for 城鎮 i 從 1 到 n
    if 城鎮 i 沒有基地臺
      cover = 0
      for 城鎮 k 從 1 到 n
        if 城鎮 k 還沒被覆蓋 and 「i 跟 k 的距離」小於等於 d
          cover += 城鎮 k 的人口
      if cover > maxCover
        maxCover = cover
        maxCoveringTown = i
  在基地臺清單新增城鎮 maxCoveringTown
```

如此這般，逐步將細節加入 pseudocode 中，最後 pseudocode 就會變成真的 code，就可以實做出佩蓉的演算法了[4]！

隨著要解決的問題規模愈來愈大，要寫的程式就會愈來愈複雜，如果只有一個粗略的想法就開始寫程式，常常會難以下手或錯誤百出。有經驗的程式設計師通常會先寫 pseudocode，透過逐步在 pseudocode 中加入細節，一邊釐清想法，一邊逐步接近最終要撰寫的程式。這樣的經驗累積得夠多之後，就不用「寫」pseudocode 了，這樣的流程會自然在你的腦中進行，然後你就會發現比起之前，自己能夠寫更困難的程式了！

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。在每個檔案中，第一列存放三個整數 n、p 跟 d；在第二列至第 $n+1$ 列中，第 i 列存放三個整數 x_{i-1}、y_{i-1} 與 P_{i-1}，分別表示第 $i-1$ 個城鎮的 x 座標、y 座標和人口數。在任意一列中，兩個數字之間都以一個空白隔開。已知 $2 \le n \le 50$、

[4]在這些 pseudocode 中，我們設定 `maxCover = -1`。如果我們改成寫 `maxCover = 0`，看起來很像，但執行下去會有一些細微的差別。請試著理解差別在哪裡吧！

$2 \le p \le n$、$0 \le d \le 300$、$-100 \le x_i \le 100$、$-100 \le y_i \le 100$、$1 \le P_i \le 100$。不會有兩個城鎮落在同一個地點。

讀入這些資料之後，請根據題目指定的演算法，求出應該在哪 p 個城鎮設置基地臺，然後依照選擇的先後順序由先而後印出這些城鎮的編號，最後輸出被覆蓋的總人數。任兩個城鎮編號間，用一個空白隔開。舉例來說，如果輸入是

```
8 3 3
3 -2 10
-1 1 15
-1 4 10
3 2 20
4 3 20
-3 -4 25
2 -3 15
0 2 10
```

則輸出應該是

```
8 1 6 105
```

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。

  之所有有「測資組」的設定，簡單地來說就是減少不完全正確的程式碰巧得到高分的機率。請大家努力考慮所有的可能性吧！

- 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！

第四題（bonus）

（20 分）承上題，我們來試著改進一下我們的演算法。這個演算法雖然簡單，但有個明顯的缺點：每一輪都只考慮蓋「一個」基地臺的影響，也就是考慮得不夠全面。考慮到全盤考慮要花太多時間，讓我們來試試看考慮「兩個」基地臺吧。現在每一輪要挑選基地臺時，我們要在所有還未有機地臺的城鎮中，一次挑選「在那兩個城鎮各蓋了一個基地臺之後能覆蓋到最多人」的那兩個城鎮來蓋基地臺。假設我們

最多可以蓋 p 個而且 p 是偶數，那麼我們會在 $\frac{p}{2}$ 輪之後結束挑選；假如 p 是奇數，那麼前 $\frac{p-1}{2}$ 輪我們會各挑兩個城鎮，第 $\frac{p+1}{2}$ 輪我們就回復到只挑選一個城鎮。

本題的輸入格式和第三題一樣，輸出時也依然是將每一輪挑選的城鎮編號依序輸出，而在單一輪內要輸出兩個城鎮編號時，請先輸出編號小的再輸出編號大的。最後一樣要輸出總覆蓋人數。如果在單一輪中，有好幾對城鎮能覆蓋一樣多人的話，該怎麼選擇呢？假設一對城鎮是城鎮 i 跟城鎮 j，$i < j$，我們說城鎮 i 是這對城鎮中的小編號城鎮，城鎮 j 則是大編號城鎮。當能覆蓋最多人的城鎮組合超過一組時，就挑小編號城鎮的編號最小的那對城鎮興建基地臺；如果還有平手，就挑大編號城鎮最小的那對城鎮。舉例來說，城鎮 3 跟 7 的組合優先於城鎮 4 跟 5 的組合，也優先於城鎮 3 跟 8 的組合。

請考慮下列範例。如果輸入是

```
6 3 2
0 0 20
1 1 25
2 2 30
3 3 25
4 4 20
4 0 35
```

則輸出應該是

```
1 4 6 155
```

請注意如果使用第三題的演算法，我們會依序挑選城鎮 3、6、1，最終只覆蓋 135 人。也請注意在第一輪中，我們挑城鎮 1 和 4，但城鎮 2 和 4 或 2 和 5 也可以覆蓋一樣多人，我們選城鎮 1 和 4 是因為其小編號城鎮的編號最小。

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。