程式設計（106-1）
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 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 ***11 月 13 日凌晨一點***。在你開始前，請閱讀課本的第 7.1–7.9 和 7.11 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）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) 用 `int* a;` 和 `int *a` 宣告指標，有根本上的不同。

(b) 用 `int* a = new int(5);` 和 `int *a = new int[5];` 宣告指標陣列，有根本上的不同。

(c) 給定兩個整數，如果要回傳比較大的那個整數的位址，下面的函數實做

```
int* max(int a, int b)
{
  int c = a;
  if(b > a)
    c = b;
  return &c;
}
```

是有問題的。

(d) 傳指標進去一個函數，跟傳位址進去一個函數，有根本上的不同。

(e) 就像 `double` 變數可以存 `int` 一樣，`double` 指標可以指向 `int` 變數。

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

（20 分）在上次作業中，我們寫過一個計算地圖上所有格子點之風險的程式。在本題中，我們要更進一步，來計算從一點到另一點的直線線段上的路段風險。直觀上來看，因為一個線段是由無限多個點集合起來的，我們似乎應該做個積分來算出路段風險，但因為這實在太麻煩了，所以我們簡化一點：我們假設飛行速率固定在每秒 1 公里，在路段上每隔 1 公里就標記一個點，然後把該路段上所有被標記的點（不含起點跟終點）的風險加總，來當作路段的風險。換言之，一個路段會被切成很多個小段，除了最後一小段以外，每段的長度都是 1 公里，而路段的風險就是這些小段的分段點的風險總和。

在本題中，你將被給定 m 個威脅點的座標、威脅半徑與威脅程度，一張橫座標與縱坐標皆從 0、1、2 一直到 n 的地圖，以及 k 條路段的起點和終點。請在所有路段中找出風險最小和最大的路段。如果有複數條路段的風險都最小，則在其中選編號最小的路段；如果有複數條路段的風險都最大，則在其中選編號最小的路段；舉例來說，如果 $n = 6$、$m = 3$，威脅點 1、2、3 分別在 $(2, 5)$、$(5, 4)$ 和 $(4, 2)$，威脅半徑依序為 2、3 和 2，而威脅程度依序為 2、1 和 2，我們來試算三條路徑的風險：

- $(1, 1)$ 到 $(4, 1)$：這條路段長度為 3，因此會有 2 個分段點，分別是 $(2, 1)$ 和 $(3, 1)$，這 4 點的風險分別約是 0 和 0.5858，這條路段的風險總和約是 0.5858。請注意起點和終點的風險是不含在內的。

- $(1,1)$ 到 $(4,5)$：這條路段長度為 5，因此會有 4 個分段點，分別是 $(1.6,1.8)$、$(2.2,2.6)$、$(2.8,3.4)$ 和 $(3.4,4.2)$，這 4 點的風險分別約是 0、0.1026、0.6072 和 0.8501，這條路段的風險總和約是 1.5598。

- $(6,1)$ 到 $(2,5)$：這條路段長度約為 5.66，因此會有 5 個分段點，分別約是落在 $(5.29,1.71)$、$(4.59,2.41)$、$(3.88,3.12)$、$(3.17,3.82)$ 和 $(2.46,4.54)$ 這 5 點的風險分別約是 0.9038、1.7362、1.3973、0.7310 和 1.4793，這條路段的風險總和約是 6.2477。請注意這 5 個分段點並不是把路段均勻地分成 6 個一樣長的小段，而是前 5 段的長度都為 1，最後一小段則比較短。

綜合以上，第一個路段的風險是最小的。

顯然地，這一題中得要用到很多浮點數運算。由於最後是一些浮點數之間比大小，而非問某些浮點數是否相等，你應該不用煩惱上課講過的浮點數精確度問題。倒是你要避免（事實上是不可以）在計算過程中做四捨五入，以免影響答案的正確性。舉例來說，上面的第三條路徑的分段點事實上都有無窮多位小數，如果在程式中我們擅自把小數點第三位以後捨去，計算結果就可能有誤差，累積起來就可能讓我們算出錯的答案。當然，你只要直接用 `double`，然後沒事不要去動那些變數裡面的內容，應該就沒事了。

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 $k+5$ 列，第一列含有兩個整數，依序是 n、m 和 k。第二列含有 m 個整數，第 i 個整數為 X_i。第三列含有 m 個整數，第 i 個整數為 Y_i。第四列含有 m 個整數，第 i 個整數為 R_i。第五列含有 m 個整數，第 i 個整數為 P_i。第六列起的第 $5+j$ 列含有四個整數，依序為 x_A^j、y_A^j、x_B^j 和 y_B^j，表示路段 j 是從 (x_A^j, y_A^j) 飛往 (x_B^j, y_B^j)。同一列的任兩個數字之間被一個空白字元隔開。已知 $n \in \{1,...,100\}$、$m \in \{1,...,10\}$、$k \in \{1,...,10\}$、$X_i \in \{0,...,n\}$、$Y_i \in \{0,...,n\}$、$R_i \in \{0,...,10\}$、$P_i \in \{1,...,10\}$、$x_A^j \in \{0,...,n\}$、$y_A^j \in \{0,...,n\}$、$x_B^j \in \{0,...,n\}$ 和 $y_B^j \in \{0,...,n\}$。

讀入這些資料之後，請找出風險最小和最大的路徑，並且印出其路徑編號。如果有複數條路徑的風險都最小或最大，則在風險最小的路徑中挑出編號最小的，在風險最大的路徑中也挑出編號最小的。兩個編號中間用一個空白字元隔開。舉例來說，如果輸入是

```
6 3 3
2 5 4
5 4 2
2 3 2
2 1 2
1 1 4 1
1 1 4 5
6 1 2 5
```

則輸出應該是

```
1 3
```

如果輸入是

```
6 3 1
2 5 4
5 4 2
```

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

則輸出應該是

```
1 1
```

針對這個題目，你可以使用任何方法。這一題的 20 分會根據程式運算的正確性給分，一筆測試資料佔 2 分。

第三題

（60 分）上次作業中，我們實作了最短路徑（shortest path）演算法，而我們建議的寫法是用 adjacency matrix 來儲存 graph。顯而易見地，如果給定的 graph 很稀疏（sparse），亦即點很多但邊不多，則用 adjacency matrix 會很浪費空間，搜尋起來也很浪費時間。本題請你改用 adjacency list 來實做。由於點的個數是固定的，你可以用靜態陣列儲存代表點的指標，每個指標指向一個一維的動態陣列；當然你要用二維動態陣列也可以。

本題並不要求你解最短路徑問題。讀入一個 graph 後，你會被給定數條可能的路徑（path），你的任務是判定每條被給定的路徑是否確實存在，若是則印出其總路徑長，若否則印出 -1。本題的 graph 可能有迴路（cycle）。

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。每個檔案中有 $m+k+1$ 行，其中第一行是三個正整數 n、m、k，依序是點的個數、邊的個數以及要判斷的路徑個數。第二行起的 m 行則每行是三個整數 u_i、v_i 和 d_{u_i,v_i}，表示第 i 條路段是從點 u_i 走到 v_i，距離為 d_{u_i,v_i}。第 $m+2$ 行起的 k 行中，第 $m+1+j$ 行有 n_j+1 個整數，依序是 n_j、$v_{j,1}$、$v_{j,2}$ 直到 v_{j,n_j}，表示要被判定的路徑是 $(v_{j,1}, v_{j,2}, ..., v_{j,n_j}$。同一行的任兩個數字中間用一個空白字元隔開。點的編號為介於 1 和 n（包含 1 和 n 的正整數。已知 $n \in \{1, ..., 100000\}$、$m \in \{1, ..., n(n-1)\}$、$k \in \{1, ..., 10\}$、$u_i \in \{1, ..., n\}$、$v_i \in \{1, ..., n\}$、$d_{uv} \in \{0, ..., 100\}$、$n_j \in \{1, ..., 50\}$、$v_{j,n_j} \in \{1, ..., n\}$、一條路段的資訊不會出現兩次、給定的路徑不會重複。

讀入這些資料後，請針對每條給定的路徑，一一判定其是否確實存在，若是則印出其總路徑長，若否則印出 -1。任兩個數字之間用一個空白字元隔開。舉例來說，如果輸入是

```
6 10 3
1 2 5
1 3 2
2 4 2
3 2 1
3 4 4
3 5 4
4 5 4
```

```
4 6 2
5 6 3
5 2 2
4 1 2 4 5
4 1 5 4 3
8 1 2 4 5 2 4 5 6
```

則輸出應該是

```
11 -1 22
```

你可以預期會出現一些測試資料有非常多點（n 非常大），因此你一宣告一個 100000×100000 的二維陣列，就超過題目的記憶體限制了。如果要拿到滿分，非得要使用動態陣列與 adjacency list 不可。

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 分會在後續作業中被評定。屆時我們會讓同學們互相檢視彼此的本題程式碼，並且就可讀性、易維護性、模組化程度、排版等面向寫評語和給評分（當然一切都是匿名的）。該任務在本題中會佔 20 分，其中 10 分取決於檢視你的程式碼的同學給你的分數（必要時助教會出來主持公道，請不用緊張），另外 10 分取決於你對同學的程式碼的評語和評分的合理性和建設性。若你在本次作業中完全沒有寫這一題，那屆時自然沒有人能檢視你的程式碼，你也就得要損失這 10 分了。

  以下我們列出屆時的同儕互評標準，供你現在撰寫程式參考：

  > 請在 PDOGS 上批改你被隨機分配到的程式碼，根據它在正確性以外的部份給它 1 至 5 分的評分，並且說明你給分的依據。建議在評分時參考以下六個面向。在前五個面向上，一個面向上做得好就得一分，還不錯則半分，不好則零分；在第六個面向上則在有必要時扣分。六個面向的分數合計後無條件進入即為你最後給的總分。
  >
  > - 可讀性：變數與函數名稱是否具有合適的資訊量？程式碼排版是否良好且具有前後一致性？是否有合適的註解？關於註解，當然不需要每一行都有註解，但若你發現在某一大段落裡都沒有註解，或某個你感覺很不易看懂的部份沒有註解，你可以指出來；不要直接說「註解太少」但沒有說是哪邊缺乏註解。
> - 模組化程度：是否有宣告合適的函數？是否有避免將非常類似的程式片段寫複數次而非寫成函數？是否有避免一個函數做非常多事情？函數間是否有合適的 decoupling？直接閱讀 main function 是否能很快地理解程式在大方向上的運算邏輯？

- 效率：程式運算是否有合理的運算效率？當然我們不要求每個同學都寫出超級有效率的精妙演算法，但至少一個程式不應該進行過多不必要的運算，也不應該耗用過多不必要的記憶體空間。如果你看不出這個程式的效率有明顯的問題，我們建議你直接給一分。
- 擴充性：當要解的問題變得更複雜的時候，我們能不能簡單地修改這個程式以解決新的問題，而不是寧可砍掉重練？這個議題當然也很主觀，所以如果你不能明確地指出在怎樣的新問題上，這個程式會有擴充性問題，我們建議你直接給一分；如果你不能指出很嚴重的問題，我們建議你至少給半分。但對批改者來說，這個關於擴充性的思考其實是很好的訓練。試試看吧！
- 其他：如果有任何其他令你想扣分的理由，請明確地寫出來並且在這個面向上扣分；沒有的話就給一分。
- 題目規範：你應該檢查那份程式碼有沒有違反題目的規範，如果有（例如題目說不可以用上課沒教過的東西，但他用了，或者題目說一定要用指標和動態記憶體配置，但他沒用），就扣他三分。當然，請明確地指出他哪邊違反了題目的規範。

第四題（bonus）

（20 分）承上題，現在給定的路徑中可能有重複的點，因此可能出現迴路（cycle），例如 $(1, 2, 4, 8, 7, 4, 9)$ 這條路徑中就有 $(4, 8, 7, 4)$ 這條迴路。由於我們的路段長度都是正的，路徑中如果有迴路，以由起點到終點來說都是不必要的。本題要請你針對每條給定的路徑去判斷，如果不存在則印出 -1，如果存在且沒有迴路則印出路徑長，有迴路則印出跳過迴路不去走的路徑長。換言之，針對給定的路徑，請求出限制在此路徑範圍內的由起點到終點的最短路徑。請注意迴路可能蠻複雜的，例如可能有 $(1, 2, 4, 8, 7, 9, 6, 7, 4, 8, 10)$ 這種路徑，此路徑上要從 1 到 9 的最短路徑應該是 $(1, 2, 4, 8, 10)$。

本題的輸入輸出格式和第三題一模一樣，但給定路徑中的起點（$v_{j,1}$）和終點（v_{j,n_j}）都只會出現一次。舉例來說，如果輸入是

```
6 10 3
1 2 5
1 3 2
2 4 2
3 2 1
3 4 4
3 5 4
4 5 4
4 6 2
5 6 3
5 2 2
4 1 2 4 5
4 1 5 4 3
8 1 2 4 5 2 4 5 6
```

則輸出應該是

```
11 -1 14
```

針對這個題目，你可以使用任何方法。這一題的 20 分會根據程式運算的正確性給分，一筆測試資料佔 2 分。