程式設計（113-1）
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/`）為第一、二、三、四題各上傳一份 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。這份作業的截止時間是 ***10 月 8 日早上八點***。為這份作業設計測試資料並且提供解答的助教是梁安哲。

在你開始前，請閱讀課本的第 7 章（關於 Pointer）[1]。

本次作業滿分為 110 分，得幾分就算幾分。若整學期有 n 份作業，則學期的作業總成績即為 n 份作業的總分除以 n（不論超過 100 與否）。

第一題

（20 分）給定一個整數 n，我們要請大家計算小於等於 n 的質數中不是費波納西數的數字個數。舉例來說，如果 $n = 15$，則小於等於 n 的質數有 2、3、5、7、11、13，其中也落在費波納西數列的有 2、3、5、13，所以答案是 2 個（只有 7 和 11）。

這一題很單純，想必網路上或 AI 工具都可以輕鬆提供答案（甚至課堂上就有幾乎把這題做完的範例程式碼了），因此我們主要的挑戰是效能與動態記憶體配置。這一題的 n 可能會很大，因此你得使用在時間上有效率的演算法，不然針對一筆測試資料如果執行時間超出 PDOGS 上規定的上限，即使結果正確也拿不到分數。此外，如果想要宣告一個做為區域變數的靜態陣列，空間要求過大時會被編譯器阻止，因此在本題中，我們建議大家練習用動態記憶體配置（Dynamic Memory Allocation）去宣告你需要的空間[2]。

說明：在 PDOGS 中，助教會設定時間上限，複雜度過高的演算法會無法得到滿分。助教也會設定記憶體用量上限，讓正確使用動態記憶體配置的同學可以拿到滿分。總之，請好好地選擇演算法並且使用記憶體吧！

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。已知 $2 \le n \le 10000000$。從第 1 到第 7 組資料，n 的值會比較小（$n \le 1000$），但從第 8 到第 10 組資料，n 的值會比較大。

讀入資料後，請輸出一個整數，代表小於等於 n 的質數中不是費波納西數的數字個數。舉例來說，如果輸入是

```
15
```

則輸出應該是

[1] 課本是 Deitel and Deitel 著的 *C++ How to Program: Late Objects Version* 第七版。
[2] 大家如果就宣告一個超大的全域靜態陣列，編譯器應該也會允許，但這畢竟並非合適的作法就是了。

```
2
```

如果輸入是

```
10000
```

則輸出應該是

```
1222
```

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

這一題的所有分數都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。

第二題

說明：本題和作業四的第四題基本上是一樣的，只是因為規模特別大，所以有一些新的要求。為了讓大家不用一直翻之前的題目，底下我們還是完整地把題目敘述一次，並且附上新的範例。

（20 分）在一個國家裡有 n 個小鎮，某些小鎮間有路相連，總共有 m 條路，每條路都是起自一個小鎮也結束於一個小鎮，路上沒有別的小鎮。小鎮間的道路關係可以用一個 $n \times n$ 的對稱矩陣 R 表示。以

$$R = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

為例，表示 $n = 6$、$m = 7$，小鎮 1 有總共四條路通往小鎮 2、小鎮 3、小鎮 4 和小鎮 5，但小鎮 1 跟小鎮 6 之間沒有路，依此類推。在小鎮 i 上住有居民 h_i 人。國王想要看看各小鎮發展的情況，也拜訪民眾，因此想要從小鎮 1（其實是首都）出發，經過 q 個不同的小鎮後回到小鎮 1。他擬定了路線 $(1, p_1, p_2, ..., p_q, 1)$，表示先從小鎮 1 走到小鎮 p_1，再從小鎮 p_1 走到小鎮 p_2，依此類推，最後從小鎮 p_q 回到小鎮 1。舉例來說，$(1, 3, 5, 1)$ 就是一個合情合理的路線，沿路可以拜訪 $h_1 + h_3 + h_5$ 這麼多個民眾（小鎮 1 雖然被經過兩次，但被拜訪的民眾數只被計算一次）。

以上一切都很完美，唯一就是國王不太擅長擬定路線，因此國王把路線交給你，請你檢查他擬的路線上是否每條路都確實存在。舉例來說，如果國王擬的路線是 $(1, 5, 3, 2, 1)$，這個路線就行不通，因為從小鎮 3 沒有路通往小鎮 2。國王給你的任務是，如果給定的路線是可行的，就計算可以拜訪的民眾

數；如果給定的路線不可行，就依照路線上的順序依序列舉不存在的路段。舉例來說，如果國王給的路線是 $(1, 5, 1)$，就輸出 $h_1 + h_5$；如果是 $(1, 5, 4, 2, 1)$，就輸出 $(5, 4)$、$(4, 2)$。

特別說明：在本題中新增的挑戰，是小鎮的個數可能高達一萬個，如果直接用 adjacency matrix 的方式儲存路線，就得建立一個有一億個元素的二維陣列，非常佔空間（和時間）。而且如果只有幾萬條路的話，這個陣列裡面就會絕大部分的值都是 0（這種矩陣也被稱為「稀疏矩陣」，sparse matrix），確實也很浪費空間。這種情況下，使用 adjacency list 可能就更合適一些。圖 1 是用 adjacency list 記錄上述例子的示意圖，其中 `neighborListCnt` 是一個指向整數的指標，`neighborListCnt[i]` 是第 `i + 1` 個小鎮往外聯通的道路數，而 `neighborLists` 是一個指向整數指標的指標，`neighborLists[i][j]` 是第 `i + 1` 個小鎮的第 `j + 1` 個相鄰的小鎮編號。在本題中，助教會在 PDOGS 上幫每筆測試資料設置記憶體用量上限，如果大家使用 adjacency matrix 的話，在某些 n 特別大的測試資料會無法拿到分數的！

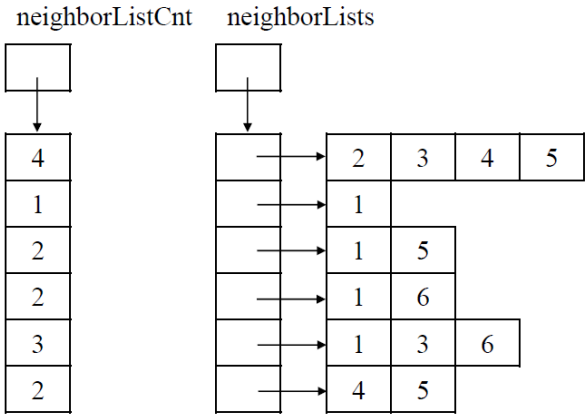

圖 1: 記錄小鎮間道路的 ajacency list

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。在每個檔案中會有 $m + 3$ 列，第一列裝著三個正整數，依序是 n、m、q；在第二列到第 $m + 1$ 列中，從第二列起算的第 j 列存有兩個整數 u_j 和 v_j，表示第 j 條路是連結小鎮 u_j 和小鎮 v_j；第 $m + 2$ 列存了 n 個正整數，依序是 h_1、h_2 直到 h_n；第 $m + 3$ 列存了 q 個正整數，依序是 p_1、p_2 直到 p_q。已知 $1 \le n \le 10000$、$1 \le m \le 100000$、$1 \le q \le n - 1$、$u_j \in \{1, ..., n\}$、$v_j \in \{1, ..., n\}$、$1 \le h_i \le 1000$、$p_i \in \{2, ..., n\}$、$p_1 \ne p_2 \ne \cdots \ne p_q$、$[u_1, v_1] \ne [u_2, v_2] \ne \cdots \ne [u_m, v_m]$。每一列的任兩個相鄰的整數間以一個空白字元隔開。

讀入這些資訊後，如果指定路線上的每個路段都存在，就輸出一個整數代表路線上會拜訪的總民眾數；反之則按照路線順序輸出所有不存在的路段，每當要輸出一個路段時，先輸出該路段的起點小鎮編號，接著輸出一個逗點，再輸出該路段的終點小鎮編號，如果後面還有下一個路段，兩個路段間用一個分號隔開。舉例來說，如果輸入是

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

```
4 6
5 2
100 200 300 400 500 600
3 5 2
```

則因為 $(1,3,5,2,1)$ 路線上每個路段都存在，輸出應該是

```
1100
```

如果輸入是

```
6 7 3
1 2
1 3
1 4
1 5
4 5
4 6
4 2
100 200 300 400 500 600
3 5 2
```

則因為 $(1,3,5,2,1)$ 路線上 $(3,5)$、$(5,2)$ 路段不存在，輸出應該是

```
3,5;5,2
```

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

這一題的所有分數都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。

第三題

特別說明：在作業六我們會延伸這個問題，讓一個電梯組一次考慮複數臺電梯呼叫並且規劃電梯的載客策略；在那之前，讓我們先處理一個呼叫，幫自己熱個身吧！

（50 分）一個由多臺電梯組成的電梯組，在某層樓有一個乘客呼叫電梯後，該派哪臺電梯去接這位乘客呢？在這一題中，讓我們來試著處理這個問題！

已知大樓內共有 n 臺電梯，且大樓共有 K 層樓，分別是一樓、二樓直到第 K 樓。對於這些電梯和這棟大樓，我們知道以下資訊：

- 電梯 i 目前在 f_i^E 樓往方向 d_i^E 前進，其中 $d_i^E = -1$ 表示電梯正在往下走、$d_i^E = 1$ 表示電梯正在往上走，而 $d_i^E = 0$ 表示電梯正靜止不動。

- 在電梯 i 裡面有 m_i^I 個「內部呼叫」(inner call)，分別要前往 $t_{i,1}^I$、$t_{i,2}^I$ 直到 $t_{i,m_i^I}^I$ 樓。內部呼叫即為乘客在電梯內部按下想要前往的樓層，這些內部呼叫的目的地不會是電梯目前所在的樓層，但可能在電梯目前移動方向的反方向（例如電梯正在六樓往上走，但有一個內部呼叫是要去三樓）。在實務上，一個內部呼叫可能來自多位乘客，但就結果來說就是一個內部呼叫而不是多個，我們也無從得知是幾個乘客要去同一層樓，所以為了簡單起見，我們就會假設每個內部呼叫都恰好對應到一個乘客。一臺電梯的內部呼叫彼此不會重複。如果電梯 i 現在靜止中，它身上不會有任何內部呼叫。

- 電梯 i 也已經被分配到 m_i^O 個「外部呼叫」(outer call)，其中第 j 個外部呼叫是要從第 f_{ij}^O 樓往方向 d_{ij}^O 移動，其中 $d_{ij}^O = -1$ 表示相對應的乘客想要往下、$d_{ij}^O = 1$ 表示相對應的乘客想要往上。外部呼叫即為乘客在所在樓層的電梯外部按下想要前往的方向。所有電梯的外部呼叫都不會重複，且外部呼叫不會從一樓要往下或從第 K 樓要往上；一臺電梯已經被分配的外部呼叫不會和該電梯同樓層。為了簡單起見，我們假設一個外部呼叫對應到恰好一個乘客，而且這個乘客必須也一定會搭已經被分配好要去接他的電梯了，其他電梯到那一層樓的時候是不會停的（除非剛好有乘客要出電梯，而且即使如此，在等候區等待的乘客還是會繼續等待要接他的電梯抵達才進入那臺電梯）。如果電梯 i 現在靜止中，它身上不會有任何已分配的外部呼叫。

- 最後，有一個獨立的、剛被新增的、還沒有被分配電梯的外部呼叫，我們稱之為「待分配的外部呼叫」。這個待分配的外部呼叫發生在第 F 樓，方向為 D，其中 $D = -1$ 表示相對應的乘客想要往下、$D = 1$ 表示相對應的乘客想要往上。這個外部呼叫不會和任何已經被分配電梯的外部呼叫重複，也不會落在任何電梯現在所在的樓層。我們假設這個外部呼叫也對應到恰好一個乘客，我們稱之為「待服務的乘客」，而且這個待服務的乘客必須也一定會搭之後我們分配去接他的電梯。

給定上述資訊，你的任務是為最後那個待分配的外部呼叫分配一臺電梯，目標是使得所有呼叫的總等待時間最小，其中外部呼叫的等待時間被定義為從現在起到該乘客走進電梯的時間差，而內部呼叫的等待時間被定義為從現在起到該乘客離開電梯的時間差。如果有多臺電梯都能讓總等待時間最小，就分配其中編號最小的電梯給這個待分配的外部呼叫。在計算各呼叫的等待時間時，我們需要知道電梯移動和開關門所需的時間。我們假設電梯移動一個樓層（例如從三樓到四樓）需要 s^F 秒，且我們忽略電梯開始移動和準備停靠時會有不同速度，亦即在任何情況下移動一個樓層都需要同樣的秒數。此外，電梯在某層樓停下來開關門一次（不論多少人進出）需要 s^S 秒。在電梯抵達一個樓層時，我們假設要離開電梯的乘客會在電梯抵達該樓層的瞬間就離開電梯，要進電梯的乘客也會在電梯抵達該樓層的瞬間就進入電梯；換句話說，所有的等待時間都不包含在所屬樓層開關門的秒數。

若要完成給定的演算任務，在我們將待分配的外部呼叫分配給 n 臺電梯中的任何一臺，我們必須有辦法知道該電梯會以什麼方式移動、處理其身上的內部呼叫和已分配的外部呼叫，才能計算那個待服務的乘客需要等多久。為了讓題目簡單一點，在本題中電梯的運作方式是受限的，或者說有點單純：

- 首先，如果第 i 臺電梯被分配要去接那個待服務乘客，它會先將那個待分配的外部呼叫加入自己的已分配外部呼叫中，讓自己的已分配外部呼叫增加一個。接著它會面對 m_i^I 個內部呼叫和 $m_i^O + 1$ 個外部呼叫等著它完成，因此它可以知道這 $m_i^I + m_i^O + 1$ 個呼叫都發生在哪些樓層，以

及自己所在的樓層 f_i^E，進而知道這些樓層中的最低樓層 L_i（未必是一樓）與最高樓層 H_i（未必是第 K 樓）。

- 如果電梯 i 現在正在往上，它就會先從目前樓層 f_i^E 往上移動到第 H_i 樓，接著折返往下到 L_i 樓，然後折返往上到第 f_i^E 樓。這些折返都不花費任何時間。如果在第 H_i 樓有個往下的外部呼叫，電梯折返時會轉為往下，讓該外部呼叫對應到的乘客在電梯一抵達該樓層時就進電梯。
- 如果電梯 i 現在正在往下，它就會先從目前樓層 f_i^E 往下移動到第 L_i 樓，接著折返往上到 H_i 樓，然後折返往下到第 f_i^E 樓。這些折返都不花費任何時間。如果在第 L_i 樓有個往上的外部呼叫，電梯折返時會轉為往上，讓該外部呼叫對應到的乘客在電梯一抵達該樓層時就進電梯。
- 如果電梯 i 現在正在靜止中，它會馬上試算往上或往下然後順著走一圈，看哪一個方向會讓自己身上的所有呼叫的等待時間總和最小，然後選擇往該方向前進。
- 電梯在 H_i 或 L_i 時，即使有兩個方向不同的外部呼叫，電梯還是只會開一次門（可以同時服務兩個外部呼叫）。

在給定如上的電梯移動演算法後，每臺電梯都會照這個方式移動，然後沿路服務內部呼叫（讓電梯內的乘客離開電梯）和外部呼叫（讓等候區的乘客進入電梯），直到服務完所有呼叫為止。

為了簡單起見，我們在計算時只需要考慮當下已知的所有呼叫，不考慮未來可能出現的新呼叫，包含乘客走進電梯後新增的內部呼叫。基於這個原則，每個呼叫的「等待時間」只是一個預計的、估計的等待時間，不一定是該呼叫真正的等待時間，但本題中我們不要求大家估計或計算真正的等待時間。

以下是一些例子。假設 $n = 2$、$K = 8$、電梯移動一層需要 $s^F = 5$ 秒，開關門一次需要 $s^S = 10$ 秒：

- 假設電梯 1 現在正在三樓靜止，電梯 2 正在八樓往下；電梯 1 因為靜止，身上自然沒有任何呼叫，而電梯 2 身上有兩個內部呼叫，分別要前往六樓和三樓，還有一個已分配的外部呼叫，要從七樓往下。如果此時有一個待分配的外部呼叫要從六樓往下：
  - 若分配給電梯 1，電梯 1 評估後會發現自己應該往上而非往下，在往上抵達六樓後開門服務此外部呼叫，因此這個待分配的外部呼叫的等待時間是 $5 \times |6 - 3| = 15$ 秒。請注意當電梯抵達六樓開門的瞬間，該外部呼叫就算是被服務了，因此其等待時間不包含電梯開關門所需的時間。電梯 2 要服務完自己身上已經被分配的任務，為此它會持續往下，在七樓、六樓和三樓分別完成服務外部呼叫、內部呼叫和內部呼叫，三者的等待時間分別是 $5 \times |8 - 7| = 5$ 秒、$5 \times (|8 - 7| + |7 - 6|) + 10 \times 1 = 20$ 秒（要包含電梯在七樓開關門一次所花的時間，但不包含在六樓開關門的時間）、$5 \times (|8 - 7| + |7 - 6| + |6 - 3|) + 10 \times 2 = 45$ 秒。所有呼叫的總等待時間是 $(15) + (5 + 20 + 45) = 85$ 秒。
  - 若分配給電梯 2，電梯 2 就會相當於身上被分配了四個任務，為此它會持續往下，在七樓、六樓、六樓和三樓分別完成服務外部呼叫、內部呼叫、外部呼叫和內部呼叫，四者的等待時間分別是 $5 \times |8 - 7| = 5$ 秒、$5 \times (|8 - 7| + |7 - 6|) + 10 \times 1 = 20$ 秒、20 秒（同前一個呼叫的等待時間）、$5 \times (|8 - 7| + |7 - 6| + |6 - 3|) + 10 \times 2 = 45$ 秒。電梯 1 可以繼續維持靜止。所有呼叫的總等待時間是 $(0) + (5 + 20 + 20 + 45) = 90$ 秒。

  綜合以上分析，我們會分配電梯 1 給那個待分配的外部呼叫，因為 $85 < 90$。

- 假設電梯 1 現在正在三樓往上，電梯 2 正在八樓往下；電梯 1 身上有一個內部呼叫要前往五樓，還有一個已分配的外部呼叫要從二樓往下，而電梯 2 身上有兩個內部呼叫，分別要前往六樓和三樓，還有一個已分配的外部呼叫要從七樓往下。如果此時有一個待分配的外部呼叫要從六樓往下：

  - 若分配給電梯 1，電梯 1 就會相當於身上被分配了三個任務，為此它會先往上，先到五樓完成服務內部呼叫，再到六樓完成服務待分配的外部呼叫，然後折返到二樓。三者的等待時間分別是 $5 \times |3-5| = 10$ 秒、$5 \times (|3-5|+|5-6|)+10 \times 1 = 25$ 秒，以及 $5 \times (|3-5|+|5-6|+|6-2|)+10 \times 2 = 55$ 秒。電梯 2 要服務完自己身上已經被分配的任務，為此它會持續往下，在七樓、六樓和三樓分別完成服務外部呼叫、內部呼叫和內部呼叫，三者的等待時間分別是 $5 \times |8-7| = 5$ 秒、$5 \times (|8-7|+|7-6|)+10 \times 1 = 20$ 秒、$5 \times (|8-7|+|7-6|+|6-3|)+10 \times 2 = 45$ 秒。所有呼叫的總等待時間是 $(10+25+55)+(5+20+45) = 160$ 秒。
  - 若分配給電梯 2，電梯 2 就會相當於身上被分配了四個任務，為此它會持續往下，在七樓、六樓、六樓和三樓分別完成服務外部呼叫、內部呼叫、外部呼叫和內部呼叫，四者的等待時間分別是 $5 \times |8-7| = 5$ 秒、$5 \times (|8-7|+|7-6|)+10 \times 1 = 20$ 秒、20 秒、$5 \times (|8-7|+|7-6|+|6-3|)+10 \times 2 = 45$ 秒。電梯 1 會往上，先到五樓完成服務內部呼叫，然後折返到二樓已分配的外部呼叫。兩者的等待時間分別是 $5 \times |3-5| = 10$ 秒以及 $5 \times (|3-5|+|5-2|)+10 \times 1 = 35$ 秒。所有呼叫的總等待時間是 $(10+35)+(5+20+20+45) = 135$ 秒。

  綜合以上分析，我們會分配電梯 2 給那個待分配的外部呼叫，因為 $135 < 160$ 。

輸入輸出格式

系統會提供一共 15 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 $3n+2$ 列：

- 整體資訊：第 1 列裝著四個整數，依序是 n、K、s^F 和 s^S。已知 $2 \le n \le 10$、$2 \le K \le 15$、$5 \le s^F \le 20$、$5 \le s^S \le 20$。
- 電梯資訊：第 2 列到第 $n+1$ 列中，由第 2 列起算的第 i 列裝著兩個整數，依序是 f_i^E 和 d_i^E。已知 $f_i^E \in \{1, ..., K\}$、$d_i^E \in \{-1, 0, 1\}$。
- 內部呼叫資訊：第 $n+2$ 列到第 $2n+1$ 列中，由第 $n+2$ 列起算的第 i 列裝著 m_i^I+1 個整數，第一個整數是 m_i^I，接著依序是 $t_{i,1}^I$、$t_{i,2}^I$ 直到 $t_{i,m_i^I}^I$。已知 $t_{ij}^I \in \{1, ..., K\}$。
- 已分配外部呼叫資訊：第 $2n+2$ 列到第 $3n+1$ 列中，由第 $2n+2$ 列起算的第 i 列裝著 $2m_i^O+1$ 個整數，第一個整數是 m_i^O，接著依序是 $f_{i,1}^O$、$d_{i,1}^O$、$f_{i,2}^O$、$d_{i,2}^O$ 直到 $f_{i,m_i^O}^O$ 和 $d_{i,m_i^O}^O$。已知 $f_{ij}^O \in \{1, ..., K\}$、$d_{ij}^O \in \{-1, 1\}$。
- 待分配外部呼叫資訊：最後一列裝著兩個正整數，依序是 F 和 D。已知 $F \in \{1, ..., K\}$、$D \in \{-1, 1\}$。

每一列的任兩個數字之間被一個空白字元隔開；所有參數的值會遵循題目敘述中提到的不重複、不相同等等規則。

讀入上述資訊後，請決定要分配哪臺電梯給待分配之外部呼叫，並且依序輸出被分配之電梯的編號和所有呼叫的總等待時間，兩個數字之間以一個逗點隔開。舉例來說，如果輸入是

```
2 8 5 10
3 0
8 -1
0
2 6 3
0
1 7 -1
6 -1
```

則輸出應該是

```
1,85
```

如果輸入是

```
2 8 5 10
3 1
8 -1
1 5
2 6 3
1 2 -1
1 7 -1
6 -1
```

則輸出應該是

```
2,135
```

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：

- 確定可以使用的語法包含 `if-else`、`for`、`while`、陣列、函數、指標、動態記憶體配置、call by reference、`<climits>` 裡面所有的東西、`<iomanip>` 裡面所有的東西、`<cmath>` 裡面的 `abs()` 和 `sqrt()`、`sizeof()`、`static_cast()`、constants 等。
- 確定不可以使用的語法包含 `printf`、`scanf`、`max`、`min`、`<cmath>` 裡面除了 `abs()` 和 `sqrt()` 以外的函數等等。

請注意正面表列的固然是都確定可以用，但沒有被負面表列的不表示可以用喔！

評分原則

- 這一題的其中 30 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結構、運算邏輯、可讀性（包含排版、變數命名、註解等等）、可擴充性、模組化程度，以及是否使用了還沒教過的語法。請寫一個「好」的程式吧！

第四題

說明：本題和作業四的第二題基本上是一樣的，只是因為規模特別大，所以有一些新的要求。為了讓大家不用一直翻之前的題目，底下我們還是完整地把題目敘述一次。

（20 分）有一個物流公司每週會配送食材到各個市場，我們令 $I = \{1, ..., m\}$ 為市場的集合。每個市場都有三個參數，而市場 i 的參數為 X_i、Y_i 和 q_i，其中 X_i 和 Y_i 是市場 i 在笛卡兒座標系的位置座標，q_i 是市場 i 每週的需求量。

食材配送是從物流中心出發的。我們令 $J = \{1, ..., n\}$ 為候選地點的集合，每個物流中心候選地點也有三個參數，而候選地點 j 的參數為 X_j、Y_j、f_j，其中 X_j 和 Y_j 是候選地點 j 的位置座標，f_j 是在該地點設置物流中心後每週的營運成本。從市場 i 到物流中心候選地點 j 的距離用曼哈頓距離計算，亦即兩地點間的距離為 $d_{ij} = |X_i - X_j| + |Y_i - Y_j|$。已知從物流中心運送一單位食材到市場時，一單位距離的成本是 C。從總部到物流中心的補貨成本因為是很有效率的大宗運補，所以可以忽略不計。

該公司原本就已經有若干個早已建設好、已經營運中的物流中心了。我們用 A_1、A_2 到 A_n 表示各候選地點上是否已經有物流中心了，如果 $A_j = 1$ 表示候選地點 j 上已經有物流中心，$A_j = 0$ 則表示沒有。公司現在希望能在還沒有物流中心的候選地點中，選擇一個地點去新建一個物流中心，以最小化每週的總成本，如果有複數個地點都可以最小化總成本，就選編號最小的。請注意因為新增物流中心會帶來額外的營運成本，不無可能在任意一個可行的候選地點上新建物流中心都讓總成本上升。若是如此，公司就應該選擇不新建物流中心。

舉例來說，假設有 $m = 3$ 個市場，市場 1 位在 $(2, 3)$ 且單週需求量為 20，市場 2 位在 $(5, 6)$ 且單週需求量為 30，市場 3 位在 $(8, 9)$ 且單週需求量為 25；有 $n = 3$ 個物流中心候選地點，地點 1 位在 $(1, 2)$ 且單週營運成本為 100，地點 2 位在 $(4, 5)$ 且單週營運成本為 120，地點 3 位在 $(7, 8)$ 且單週營運成本為 150。如果目前沒有已經建設好的物流中心（亦即 $A_1 = A_2 = A_3 = 0$），且 $C = 5$，則公司可以依序考慮三個地點。如果在地點 1 建立物流中心，單週補貨總成本是 $5 \times (20 \times 2 + 30 \times 8 + 25 \times 14) = 3150$，單週營運成本為 100，因此單週總成本為 3250；如果在地點 2 建立物流中心，單週總成本為 1820；如果在地點 3 建立物流中心，單週總成本為 2000。結論是在地點 2 新建物流中心可以讓總成本最小，單週總成本為 1820。

前一個例子中原本是沒有已經建好的物流中心的，所以讓我們再看一個例子。假設還是那三個市場和那三個候選地點，但是在候選地點 2 已經有物流中心了。那麼公司的考量將會是：

- 如果在地點 1 建立物流中心，市場 1、2、3 應該分別由在地點 1、地點 2、地點 2 的物流中心補貨，三個市場的單週補貨成本將依序是 $5 \times 20 \times 2 = 200$、$5 \times 30 \times 2 = 300$、$5 \times 25 \times 8 = 1000$，合計 1500，而單週營運成本為 $100 + 120 = 220$，因此單週總成本為 1720，比原本的 1820 小，是一個可以考慮的選擇。
- 如果在地點 3 建立物流中心，市場 1、2、3 應該分別由在地點 2、地點 2、地點 3 的物流中心補貨，三個市場的單週補貨成本將依序是 $5 \times 20 \times 4 = 400$、$5 \times 30 \times 2 = 300$、$5 \times 25 \times 2 = 250$，合計 950，而單週營運成本為 $120 + 150 = 270$，因此單週總成本為 1220，也比原本的 1820 小，是一個可以考慮的選擇。

結論是在地點 2 已經有一個物流中心的情況下，在地點 3 再新建一個物流中心可以讓總成本最小，新的總成本是 1220。

在本題中，請輸出能最小化總成本的新建一個物流中心的地點編號。如果新建任何一個物流中心都比不新建帶來更高的成本，則輸出 0。

特別說明一：相較於作業四第二題，本題新增的挑戰是 n 跟 m 可能都很大，例如好幾千，因此如果像當時那樣先算好一個距離矩陣然後反覆用，則該距離矩陣可能本身就有幾千萬個值，一方面是浪費空間，二方面是浪費時間（如果真的花時間算出這幾千萬個值，其實絕大部分都不需要被用到）。所以在本題中，助教會在 PDOGS 上設定記憶體用量和運算時間上限，如果大家先把距離矩陣完整地算出來，會無法得到分數的。因此我們建議大家寫一個給定兩點的座標後可以計算距離的函數（當然這個函數相當簡單），然後當需要計算兩點間的距離時就呼叫這個函數。此外，我們用這個只佔 20 分的題目小小地給大家思考演算法時間複雜度的挑戰；給定任何一個演算法，大家得要想想有沒有不需要的步驟，進而改進演算法去只做必要的計算。舉例來說，當我要評估一個還沒有物流中心的候選地點 j_1 時，我可能會計算「加上 j_1 後的總成本」，但事實上計算「加上 j_1 後省下的成本」可能更快，因為大部分的市場都會維持由原本就最近的物流中心繼續補貨，只有少數市場會改由 j_1 補貨。提示就到這裡，總之請大家在面對運算時間限制時，試著設計更有效率的演算法吧！

特別說明二：我們也透過這個題目讓大家知道兩個進階的效率議題。首先，大家可能知道呼叫函數本身（不包含執行函數內的運算）也要花掉一些時間，雖然通常就是一點點而已，但如果一個函數要被呼叫個幾十萬次，那省下這幾十萬次呼叫可能也是值得的；再加上我們這個函數非常陽春，或許根本就不要寫這個函數，程式可以執行得更快。其次，不先算好距離矩陣而是每次遇到才算，有可能會重複地幫兩個地點做多次距離計算（可能會，可能不會，大家可以想想看）。由於只是基礎課程，對這兩個進階議題我們就點到為止就好。

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。在每個檔案中會有八列，第一列裝著三個正整數，依序是 n、m 和 C；第二列存了 n 個整數，依序是第 1 個、第 2 個直到第 n 個候選地點的 x 座標；第三列存了 n 個整數，依序是第 1 個、第 2 個直到第 n 個候選地點的 y 座標；第四列存了 m 個整數，依序是第 1 個、第 2 個直到第 m 個市場的 x 座標；第五列存了 m 個整數，依序是第 1 個、第 2 個直到第 m 個市場的 y 座標；第六列存了 n 個整數，依序是 f_1、f_2 直到 f_n，分別是各候選地點上營運物流中心的單週營運成本；第七列存了 m 個整數，依序是 q_1、q_2 直到 q_m，代表每個市場的單週需求量；第八列存了 n 個非零則一的整數，依序是 A_1、A_2 直到 A_n，代表候選地點是否已經有物流中心了，1 表示有，0 表示沒有。已知 $1 \le n \le 10000$、$1 \le m \le 10000$、$1 \le C \le 20$、所有的座標值都落在 0 和 100 之間（包含 0 和 100）、$1 \le f_j \le 10000$、$1 \le q_i \le 100$、$A_j \in \{0, 1\}$。

舉例來說，如果輸入是

```
3 4 5
34 48 68
45 37 78
34 45 68 35
57 34 58 29
36 37 38
22 84 37 36
```

```
0 0 0
```

則輸出應該是

```
2
```

如果輸入是

```
12 3 20
37 48 59 36 48 38 39 27 57 38 37 58
45 57 24 58 24 46 68 42 47 57 26 47
46 58 68
34 57 63
66 55 48 36 45 58 45 57 68 84 27 26
37 58 96
0 0 1 0 1 1 0 0 0 1 0 0
```

則輸出應該是

```
12
```

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

這一題的所有分數都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。