九十五學年度高級中學資訊學科能力競賽決賽

上機程式設計題

作答注意事項：

一、 對考題有任何疑義，請於考試開始後二個小時之內填寫「問題單」，交付監考人員轉送命題委員提出問題，逾時不予回覆。

二、 第一題到第四題每題 16 分，第五題和第六題各 18 分，共 100 分。

三、 可選擇指定解題語言中任何一種語言解題。

四、 最後繳交編譯後之執行檔限定在 Windows XP 的命令提示字元下執行。

五、 各題執行檔檔名請設定如下：
考生編號_題號.exe
例如：101_1.exe

六、 各題原始碼檔名請設定如下：
考生編號_題號.解題語言附屬檔名
例如：101_1.c

七、 各題輸入資料檔名如下：
in_題號.txt
例如：in_1.txt

八、 各題輸入方式以讀檔方式為之，請以目前工作目錄（Current Working Directory）下的檔案名稱為讀取路徑。

九、 各題輸出方式為標準輸出（螢幕）。

十、 考試結束後，將不再允許更動及重新編譯程式。

十一、所有發展的程式必須在 10 秒以內於試場內的電腦輸出結果，否則不予計分。

1. 影像像素轉換問題

問題敘述：

一個 $n*n$ 影像檔包含 n^2 個像素。每個像素包含紅(R)綠(G)藍(B)三個顏色。寫一個程式，讀入一個 $n*n$ 影像檔，根據下列式子，將紅綠藍三個顏色轉換成 XYZ 表色系統，其中 Y 爲影像亮度，將轉換成的 XYZ 根據輸入順序輸出，並算出影像平均亮度，將答案輸出。

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.5149 & 0.3244 & 0.1607 \\ 0.2654 & 0.6704 & 0.0642 \\ 0.0248 & 0.1248 & 0.8504 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

表示

$X = 0.5149 * R + 0.3244 * G + 0.1607 * B$
$Y = 0.2654 * R + 0.6704 * G + 0.0642 * B$
$Z = 0.0248 * R + 0.1248 * G + 0.8504 * B$

輸入說明：

每一個像素皆由 RGB 三個顏色組成，每個顏色的值是介於 0 到 255 之間(含)的整數，若一個像素的組成爲 $R = 255$、$G = 3$、$B = 192$，則此像素表示爲 255 3 192 (整數間以一個空白字元分隔)。

輸入檔案的第一行爲一整數 n ($0 < n \leq 256$)，之後有 n 行，每一行代表 n 個像素，每個像素都是由上述的方式表示 (每個像素間以一個空白字元隔開)。

輸出說明：

對每一個像素請根據輸入順序，由左而右，由上而下輸出，每一行請輸出 1 個像素，共 n^2 行，每一個像素依序輸出 X、Y、Z 的值，並且三個值之間都以一個空白字元分隔，並在最後一行輸出影像平均亮度 (請印出 “The average of Y is”)。其中 X、Y、Z 的值與影像平均亮度的值請表示到小數點後第 4 位 (以下四捨五入)。

註：每一個數字誤差在 0.0001 之內就算正確。

輸入範例：

```
2
255 3 192 254 16 171
224 51 167 160 34 8
```

輸出範例：

```
163.1271 82.0146 169.9752
163.4547 89.1162 153.7144
158.7189 104.3614 153.9368
94.6992 65.7712 15.0144
The average of Y is 85.3159
```

2. 聖誕老人的馴鹿

問題敘述：

乖小孩都可以在聖誕夜收到聖誕老人送的禮物。但是乖小孩這麼多，聖誕老人能夠在時間內將禮物送完，是有訣竅的。聖誕老人藉由他的法力，不但可以讓馴鹿拉著裝滿禮物的雪橇飛上天際，還可以將空間扭曲，將乖小孩們的住處排列在一直線上，然後將雪橇加速到光速，很快的將一整雪橇的禮物送完。

雖然聖誕老人有很強的法力，但麻煩的是馴鹿要吃草。在連續飛行x公里後，馴鹿必須吃x^2公斤的牧草，才願意繼續飛行。萬一在送完禮物之前的某次著陸時，沒有足夠的牧草給馴鹿吃，馴鹿是會罷工的，那麼有些乖小孩就收不到聖誕禮物了。所以精靈們必須將乖小孩所在的位置經過空間扭曲之後的座標清單算好，長度單位以公里計，並且計算好每次降落後要準備多少牧草給馴鹿吃，再將這些資料交給聖誕老人。

你現在是精靈工廠的電腦工程師，負責計算聖誕老人必須準備多少牧草，才能從北極出發，而順利送完禮物給所有在清單上的乖小孩。假設空間扭曲成直線後，聖誕老人出發的座標是 0，接著聖誕老人會選擇路程最短的方式送禮物，也就是座標小的先送，座標大的後送。例如：如果乖小孩有 2 位，座標分別是 5 和 17，那麼聖誕老人會先送給座標 5 的乖小孩，再送給座標 17 的乖小孩，一共需要準備 $5^2 + (17-5)^2 = 169$ 公斤的牧草。

因爲資料量很大，就算作爲一個精靈，也無法拿紙筆計算，你必須寫一個程式來計算聖誕老人需要準備多少牧草給馴鹿吃。

輸入說明：

輸入的第一行爲一個小於 10000 正整數 n，代表這一雪橇的禮物一共要送給 n 個乖小孩。緊接的有 n 行，每行都恰有一個正整數，代表某個乖小孩的家經過空間扭曲後在直線上的座標（假設這 n 個正整數皆不相同）。在直線座標上，每個乖小孩與距他最近的乖小孩不會超過 100 公里，而距出發點最近的乖小孩也在 100 公里內。

輸出說明：

輸出聖誕老人需要準備多少公斤的牧草。

輸入範例 1：

```
5
50
150
199
99
225
```

輸出範例 1：

```
10579
```

輸入範例 2：

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

輸出範例 2：

```
5
```

輸入範例 3：

```
5
100
50
98
2
1
```

輸出範例 3：

```
4614
```

3. 關鍵邏輯閘

問題敘述：

一個組合電路(combinational circuit)由線路(wires)連接一組邏輯閘(logic gates)而成，並且不包含有向迴路(directed cycle)。一個組合電路的效能決定於最後一個主要輸出(primary output)被算出來的時間。假設每一個邏輯閘所需的運算時間都是固定並且是已知的，而線路的延遲(delay)是 0。我們希望把一個組合電路所有的關鍵邏輯閘找出來。若一個邏輯閘的運算時間有任何延長，便會影響到整個電路的效能，它就被稱爲關鍵邏輯閘。以圖一的組合電路爲例，I_1、I_2、I_3是主要輸入；O_1、O_2是主要輸出；圓圈代表邏輯閘，箭頭代表線路；每個邏輯閘有自己的編號以及固定的延遲時間。在圖一的例子當中，該組合電路中的O_1會因爲邏輯閘 2、4、5 的延遲，在時間 400 才會收到運算結果；而O_2會因爲邏輯閘 2、4、6 的延遲，在時間 600 才收到運算結果，因此O_2是最後一個被算出來的主要輸出。關鍵邏輯閘分別是 2、4 以及 6 號邏輯閘，當其中一個關鍵邏輯閘的運算時間有任何延長，O_2被算出來的時間也會再往後延遲。相反地，就算 1 號邏輯閘的運算時間從 150 延長到 160，也不會影響到O_2被算出來的時間，因此 1 號邏輯閘並不是關鍵邏輯閘。

輸入說明：

第一行爲一個整數 n ($0 < n < 10000$)，代表一個組合電路的邏輯閘總數，每個邏輯閘的編號都不同，且範圍是介於 1~n 之間的整數。第二行爲一個整數 m ($m < 50000$)，代表線路的總數。接下來的 n 行，依序列出每個邏輯閘的運算時間；每個運算時間的值是介於 0 到 600 之間(含)的整數。最後 m 行，分別列出每一條線路由某個邏輯閘的輸出接到另一個邏輯閘的輸入。

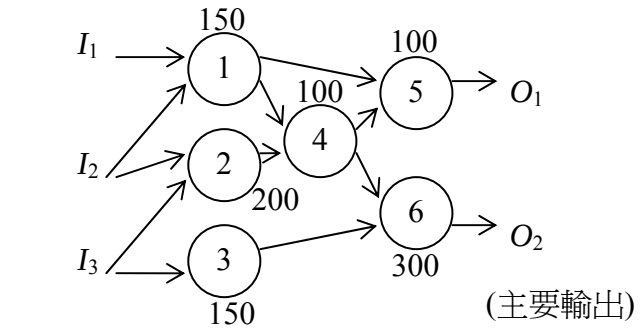
注意：爲簡化輸入，我們把主要輸入(primary inputs)與邏輯閘之間的線路，以及邏輯閘與主要輸出(primary outputs)之間的線路省略。(每一個邏輯閘至少都含有一個線路輸出到另一個邏輯閘或主要輸出)

輸出說明：

列出關鍵邏輯閘的個數。

輸入範例 1：

```
6
6
150
200
150
100
100
300
1 5
1 4
2 4
4 5
4 6
3 6
```



(主要輸入)

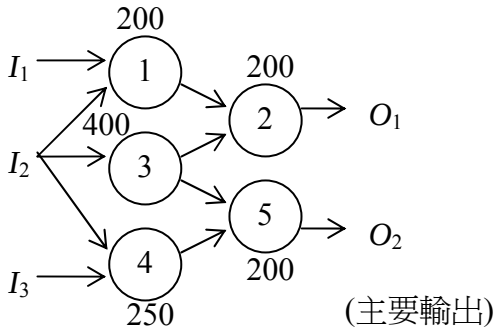
圖一

輸出範例 1：

```
3
```

輸入範例 2：

```
5
4
200
200
400
250
200
1 2
3 2
3 5
4 5
```



(主要輸入)

圖二

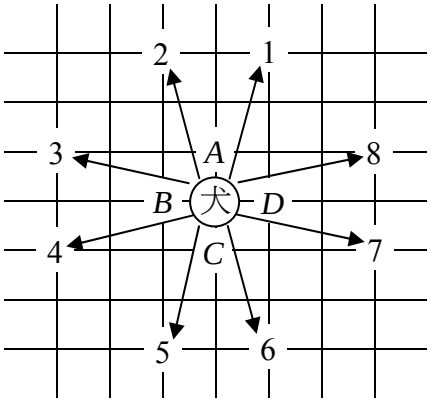
輸出範例 2：

```
3
```

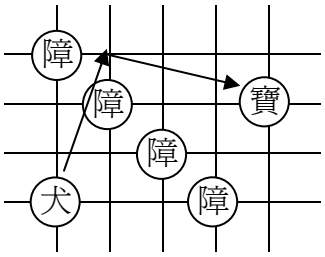
4. 靈犬尋寶

問題敘述：

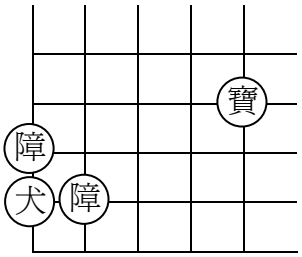
正方形(左下角座標爲(0,0)，右上角座標爲(99,99))的格網上，有一隻靈犬要尋找一個寶物，格網上除了靈犬與寶物之外，還有一些障礙物。一般情況下，只要不超出格網的邊界，靈犬的每一步最多有 8 個方向可供選擇，如圖一；但是必須注意，只有在 *A* 點沒有障礙物時才可以選擇方向 1 或方向 2，只有在 *B* 點沒有障礙物時才可以選擇方向 3 或方向 4，只有在 *C* 點沒有障礙物時才可以選擇方向 5 或方向 6，只有在 *D* 點沒有障礙物時才可以選擇方向 7 或方向 8。如果靈犬可以從出發的位置走到寶物的位置，其總共使用的步數，理論上應有一個最小值；但是靈犬也有可能無法走到寶物的位置。過程中，靈犬不可以走到障礙物的位置上。



圖一



圖二



圖三

以圖二爲例，有多達 4 個障礙物介於靈犬與寶物之間，但是靈犬最快只要 2 步就可以到達寶物的位置。圖三是另一個例子，靈犬的位置靠近角落，雖然只有 2 個障礙物，靈犬卻無法到達寶物的位置。

請撰寫一個程式，若靈犬可以從最初位置走到寶物的位置時，請列印出其使用之最少步數；若靈犬無法到達寶物的位置，請列印出單字『impossible』。

輸入說明：

第一行爲一個整數 n，代表障礙物的個數，$0 \le n \le 1000$。接下來的 n 行，每行表示一個障礙物的座標，其橫座標值與縱座標值間以一個空白隔開。

再下來的一行，表示靈犬的最初位置，其橫座標值與縱座標值間以一個空白隔開。

最後一行，代表寶物的位置，其橫座標值與縱座標值間以一個空白隔開。

注意：輸入之障礙物、靈犬、寶物皆在不同位置。所有橫、縱座標值均爲介於 0(含)至 99(含)之間的整數。

輸出說明：

依行走之規則，若靈犬可以從其位置走到寶物的位置時，請列印出其使用之最少步數；若靈犬無法到達寶物的位置，請列印出單字『impossible』。

輸入範例 1：

```
4
3 6
4 5
5 4
6 3
3 3
7 5
```

輸出範例 1：

```
2
```

輸入範例 2：

```
2
1 1
0 2
0 1
4 3
```

輸出範例 2：

```
impossible
```

5. 快遞服務

問題敘述：

「收的快」快遞公司成立之後，已經分別與市內許多中小企業公司簽訂郵件收送服務契約。由於有些公司是在同一棟大樓內，所以「收的快」需要收件的地點（收件點）最多只有 m 點 $(1, 2, \ldots, m)$，因此「收的快」僅先行採購了三輛貨車並聘用了三名司機，每天早上分別從收件地點 「1」, 「2」 及 「3」出發。而在與客戶的服務契約中有明訂「收的快」必須在客戶提出郵件寄送要求的隔天派人至該公司（地點）收件。爲了能更有效率的服務客戶並節省收件時間，該公司設立了收件服務登記網站，客戶如有郵件需要寄送，必須在需要收件的前一天就先上網登記。因此「收的快」就可以利用晚上先行安排三位司機隔天的收件路線。每位司機至各地點收件的順序應與各公司上網登記的順序相符且必須能在最省油的情況下完成當天所有的收件服務。因此每位司機有可能需要在不同時間重複到同一地點收件，或不同的司機有可能需在不同的時間點前往同一地點收件。如下面範例二(收件公司地點依序爲： 4 2 4 1 5 4 3 2 1)所示，雖然司機一一開始就已經在收件地點「1」了，但是他卻不能先把後面第四個登記的公司（地點「1」）郵件先收了再前往第一、第二、或第三個登記收件地點（地點「4」,「2」,「4」）收件。但是如果前三個登記收件的服務是由司機二或三來負責，則司機一就可以在地點「1」收了第四個登記的郵件後再前往後面所登記的地點收件。此外，在某些情況下，不一定每輛車都要收到貨，也就是說，最佳收件方式也有可能是只需出動一或兩輛車去收貨。請寫一個程式來幫「收的快」公司計算每天依預約順序至各收件地點收件的最少總耗油量。

輸入說明：

輸入檔案第一行有一個整數 m，$3 \le m \le 200$，代表「收的快」公司收件的地點數，以 1 至 m 之間的整數代號來表示每個地點。接下來的 m 行（第 2 到第 m+1 行），每行有 m 個整數，代表一個矩陣 D。第 i+1 行的第 j 個整數是 $D(i, j)$，$D(i, j)$ 代表司機開車從收件點 i 到收件點 j 所需耗油量。最後有一行數串，數串之數字依序爲前一天上網登記要求收件的公司地點代號，最多會有 1000 個收件請求。輸入檔案中任兩個相鄰的整數都以一個空白隔開。

注意：油量矩陣 D 滿足三角不等式，也就是說 $D(i, j) \le D(i, k) + D(k, j)$，$1 \le i, j, k \le m$。因此，每輛車前往下一個收件地點時一定是直接前往，不必先繞道至其它地點再抵達下個收件地點。

輸出說明：

請輸出一個整數，代表收件所需最少總耗油量。

輸入範例 1：

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

輸出範例 1：

6 ← 到每個請求收件地點的司機分別爲 1 1 1 1 3 3 2 2 2 1，因此司機 1 只需從起使點 1 移動到地點 3，司機 2 只需停留在地點 2，司機 3 從起始點 3 移動到地點 4。

輸入範例 2：

```
5
0 1 1 1 1
1 0 2 2 2
1 1 0 2 1
2 1 3 0 1
3 2 3 4 0
4 2 4 1 5 4 3 2 1
```

輸出範例 2：

5 ← 到每個請求收件地點的司機分別爲 1 2 1 2 2 1 3 1 3，因此司機 1 只需從起使點 1 移動到地點 4，再到地點 2，司機 2 從起始點 2 移動到地點 1，再到地點 5，司機 3 只需從起始點 3 移動到地點 1。

6. 糊塗情報員

問題敘述：

有一位間諜，依他所屬情報單位要求編碼方式，將他所收集到情報全部編成數字碼。但他認爲這樣還是不夠安全，因此他再將這些數字字串，隨意切割成好幾個整數，然後將每個整數用一個數學算式來表示。這些算式只用了加、減、乘三種運算子，而且每個運算元都是正整數。最後，他爲了讓他自己更爲心安，他將整個密碼分成兩本密碼簿儲存。密碼本 A 存放這些數學算式，但他將算式內的所有括號全部拿掉，然後再將這些拿掉的括號資訊記錄在密碼本 B 裡面。

過了不久他返國述職，發現他把密碼本 B 弄丟了。再加上他的記憶力不好，很多情報內容根本記不得，所以現在沒了密碼本 B 幾乎束手無策。在不得已的情況下他的情報單位派了幾位心理與腦神經生理專家詢問他，希望能喚起他腦海內的記憶。這些專家試了好幾天，用盡各種心理生理辦法後，終於承認他的記憶力果真很差，怎麼也問不出情報內容。倒是心理學專家有一發現，即這位情報員在寫密碼算式時，傾向於將括號加在那些會讓算式得最大值的位置。例如 5*7+2 這個算式，有兩種括法：((5*7)+2)以及(5*(7+2))，第二種括法所得的值較大。請寫一程式，算出這些算式的可能最大值。

輸入說明：

每一筆輸入資料爲一行算式，運算子只有三種即一般的加、減、乘三種二元運算子，分別以符號 "+"、"-"、"*" 表示。

每一個運算元都是一個正整數(≤ 100)，運算元和運算子之間不會有空白，一行算式不會有超過 50 個運算元。

輸出說明：

相對於每一輸入算式，輸出所有可能運算結果的最大值。該值都會是一個正整數，而且不會超過 2147483647。

輸入範例 1：

```
5*7+2
```

輸出範例 1：

```
45
```

輸入範例 2：

```
6*3-9*3
```

輸出範例 2：

```
27
```

輸入範例 3：

```
5+2-7*2-3
```

輸出範例 3：

```
14
```