2002 年 IOI 試題

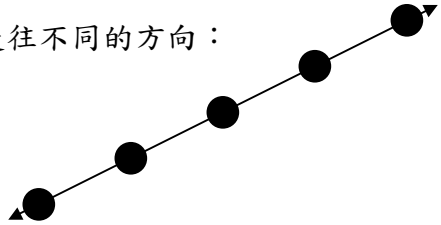
麻煩的青蛙

問題說明

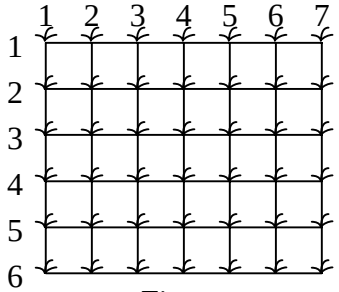
在韓國，*成古利*（一種小青蛙）的頑皮是眾所皆知的，且名符其實，因為這種青蛙在晚上一路跳躍過你的稻田，並踩平稻作物。在早上，看過被踩平的稻作物後，你要找出造成最大傷害的那條青蛙踩過的路徑。青蛙永遠以直線跳躍過稻田，每個跳躍的長度皆相同:

不同青蛙可以有不同的跳躍長度：

或是往不同的方向：



如 Figure 1 所示，稻作物種在格子點上。如 Figure 2 所示，麻煩的青蛙由稻田的一側跳入，一路跳躍至稻田的另一側離開。



Figure

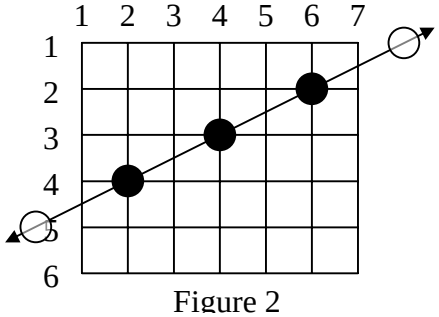
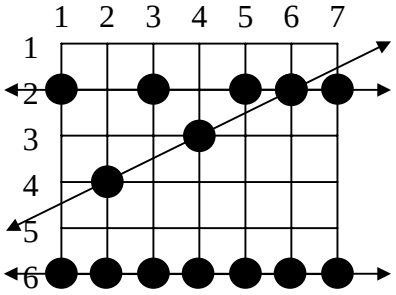
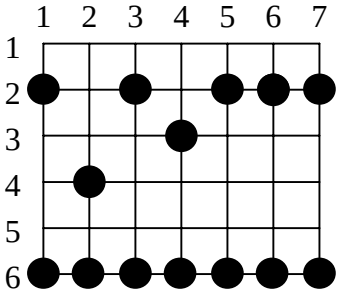


Figure 2

青蛙從一株稻作物跳躍到另一株稻作物。如 Figure 3 所示，每一個跳躍踩平所處之稻作物。請注意，有些稻作物可能被一隻以上的青蛙踩平過。以 Figure 3 為例，你無法看到稻田外之青蛙跳躍路徑，Figure 4 為你所觀察到的現象。



Figure



Figure

從 Figure 4，你將可以重建青蛙跳躍過的所有可能路徑。你有興趣的路徑應該至少有 3 株稻作物被踩平，這種路徑叫做青蛙路徑。Figure 3 標示的三個路徑都是青蛙路徑(當然也有可能還有其他的青蛙路徑)。例如，我們對第一行上的路徑沒有興趣，因為此路徑只有 2 株稻作物被踩平（跳躍長度是 4）; 另外跳躍過（列 2 行 3)，(列 3 行 4）及（列 6 行 7）之斜角路徑，雖然有三株稻作物被踩平，但因為跳躍長度並不規則，所以不是合法路徑。請注意，有些踏平的稻作物無法由任何青蛙路徑來解釋，如 Figure 4 內第二列中踏平之稻作物可以為一個青蛙路徑參雜其他原因所踏平之稻作物，如（列 2 行 6）之稻作物。

你的任務是寫一個程式在所有可能的青蛙路徑中，找到具有被踩平最多稻作物之所屬的青蛙路徑(即引起稻作物最大傷害）。在 Figure 4 中列 6 中之青蛙路徑，產生答案 7。

輸入

你的程式將由標準輸入讀入。第一行包括兩個整數 R 和 C，分別是你的稻田的列和行數，
1 <= R, C <= 5000。第二行包括單一個整數 N，其為被踩平稻作的個數，3 <= N <= 5000。剩下的 N 行，每行包括兩個整數，即為被踩平稻作物的列號碼 (1 <= 列號碼 <= R) 和行號碼 (1 <= 行號碼 <= C)，用一個空白間隔。每株被踩平的稻作物只列出一次。

輸出

你的程式將寫到標準輸出。輸出單一個整數在一行。如果存在任何合法之青蛙路徑，找出具有最大傷害之青蛙路徑，並輸出此青蛙路徑所屬的踩平之稻作物數；否則輸出 0。

輸入及輸出範例

範例 1:

輸入
(圖四的例子)

```
6 7
14
2 1
6 6
4 2
2 5
2 6
2 7
3 4
6 1
6 2
2 3
6 3
6 4
6 5
6 7
```

輸出

```
7
```

範例 2:　　輸入(Figure 5 的例子)

```
6 7
18
1 1
6 2
3 5
1 5
4 7
1 2
1 4
1 6
1 7
2 1
2 3
2 6
4 2
4 4
4 5
5 4
5 5
6 6
```

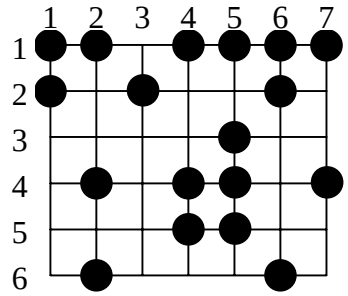


Figure 5

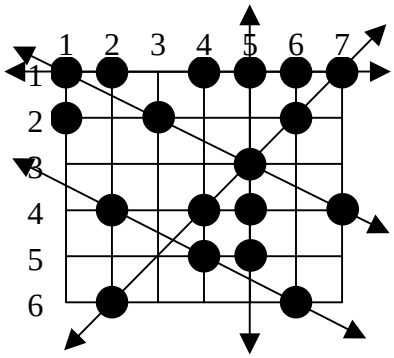


Figure 6:青蛙踩平的最大稻作物數是 4

輸出

```
4
```

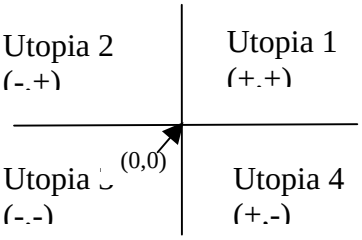
評分方式

如果你的程式在時間限制內輸出一個測試狀況的正確答案，則你得到那個測試狀況的全部分數， 否則你得到 0 分。

分割的烏托邦

問題說明

烏托邦的美麗國土曾遭戰火摧殘。當敵對結束的時候，國土被一條經線（由北至南）及一條緯線（由東至西）分割為四個區域。這兩條線的交點為原點（０，０）。這四個區域在一開始的時候都自稱為烏托邦。但隨著時間過去，這四個區域開始被稱為烏托邦１（東北區域），烏托邦２（西北區域），烏托邦３（西南區域），及烏托邦４（東南區域）。在這四個區域中的任何一個地點都可以依它在原點以東及以北之距離來標示。這些距離可以是負數，因此烏托邦２內的任一地點之座標為（負數，正數），烏托邦３內的任一地點之座標為（負數，負數），烏托邦４內的任一地點之座標為（正數，負數），而烏托邦１內的任一地點之座標為（正數，正數）。



在這四個區域中的公民不能跨越經線或緯線。很幸運的是，烏托邦的ＩＯＩ參賽者發明了安全的電子傳送機器。這個機器需要指示碼來操作。每一個指示碼只能被使用一次。現在你的挑戰是指揮這個電子傳送機器將人員由原點（０，０）出發依序經過所指定的區域。你將會有Ｎ個需依序經過的區域，但至於到達區域中的哪一個地點並不重要。在依序經過的區域中可能被要求由一個區域傳送到同一個區域。在離開原點（０，０）之後你不能停留在經線或緯線上。

你將會被提供２Ｎ個正整數做為輸入，並將這２Ｎ個正整數加上正號或負號成為Ｎ對整數做為指示碼來操作。如果你現在在地點（ｘ，ｙ），且使用指示碼（＋ｕ，－ｖ），則你將被傳送至（ｘ＋ｕ，ｙ－ｖ）。你可以採用任意順序來使用這２Ｎ個正整數，並任意加上正號或負號。

給定７，５，６，１，３，２，４，８等指示碼，來指示電子傳送機器依照４，１，２，１區域的順序，進而產生下列指示碼（＋７，－１），（－５，＋２），（－４，＋３），（＋８，＋６）。按照這一個指示碼順序，你將由原點（０，０）依序到達下列地點（７，－１），（２，１），（－２，４），及（６，１０）。這些地點分別位於烏托邦４，烏托邦１，烏托邦２，烏托邦１。

工作

給定２Ｎ個不同的正整數及依序應經過之Ｎ個區域號碼，你需要將所給定之２Ｎ個正整數轉換成一序列之指示碼，讓電子傳送機器將人員由原點（０，０）出發依序經過所指定的區域。

輸入

你的程式將由標準輸入讀取資料。第一行有一個正整數Ｎ（１＜＝Ｎ＜＝１００００），第二行有２Ｎ個正整數（這些正整數介於１及１０００００之間，包括１及１０００００），這２Ｎ個正整數都不相同，且由空白隔開。最後一行為Ｎ個應依序經過之區域號碼（可為 1, 2, 3, 或 4）。

輸出

你的程式將由標準輸出印出結果。輸出將有Ｎ行，每一行有一個指示碼。每一個指示碼包含一對具有正負號的整數，來指揮電子傳送機器將人員由原點（０，０）出發依序經過所指定的區域。注意：在正負號和數字之間不應有任何空格。但每一個指示碼中的兩個整數之間須有一空格。

如果存在多組解，你的程式只需輸出一組解。如果不存在任何解，你的程式需輸出一個整數０。

輸出入範例

Example 1: 輸入　　輸出

```
4
7 5 6 1 3 2 4 8
4 1 2 1
```

```
+7 -1
-5 +2
-4 +3
+8 +6
```

Example 2: 輸入　　輸出

```
4
2 5 4 1 7 8 6 3
4 2 2 1
```

```
+3 -2
-4 +5
-6 +1
+8 +7
```

計分

如果你的程式在時間限制內正確回答一個測試資料，則你可以得到該測試資料的全部分數，否則得到０分。

XOR

問題說明

假設你在發展一個行動電話的應用程式，正如下列圖示，其行動電話有一個黑白螢幕，其中 X 軸由螢幕的左方算起；Y 軸由螢幕的上方算起。在此應用程式中，你需要使用各種大小的影像。在程式中你不用儲存影像，而必須使用行動電話專用的繪圖程式庫。你可以假設一開始繪圖時，螢幕的所有像素為白點。行動電話專用的繪圖程式庫只有一個繪圖指令 XOR(L,R,T,B)，其動作為將左上角座標為(L,T)、右下角座標為(R,B)的長方形內的所有像素的值由白變黑或由黑變白，亦即”反白”。其中 L 代表長方形的左端座標；T 代表長方形的上端座標；R 代表長方形的右端座標；B 代表長方形的下端座標。

試舉一例說明，考慮 Figure-3 的影像結果，一開始繪圖時，一個全白的影像因使用 XOR(2,4,2,6)指令得到 Figure-1，接著使用 XOR(3,6,4,7)指令於 Figure-1 可得 Figure-2，最後使用 XOR(1,3,3,5)指令於 Figure-2 可得 Figure-3。

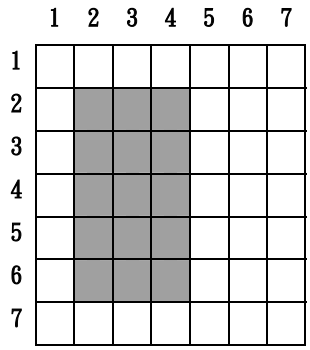


Figure-1

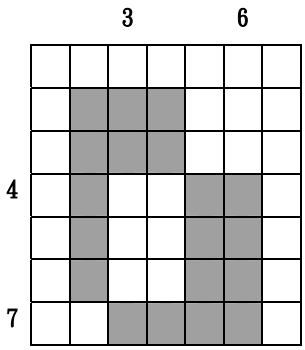


Figure-2

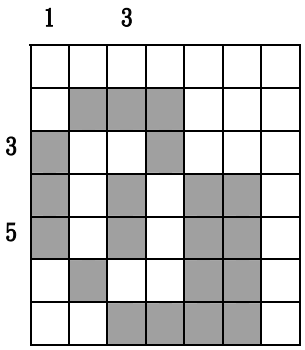


Figure-3

給定一組黑白影像，你的工作是從起始的全白螢幕開始，產生最少的 XOR 指令來繪出給定的影像。我們會提供輸入檔描述給定的黑白影像，你必須產生輸出檔包括需要的 XOR 指令及其參數。本題不需要提供產生這些輸出檔的程式。

輸入

我們會提供 10 個程式輸入檔，檔名為 xor1.in 到 xor10.in。每個輸入檔的格式如后：輸入檔的第一行含一個整數 N，其範圍為 5≦N≦2000，表示給定的影像有 N 行和 N 列。第二行以後為給定影像由上到下的像素值， 每一行含有 N 個整數，對應到影像由左到右的像素值，其值為 0 或 1，其中 0 代表白的像素；1 代表黑的像素。

輸出

你應提供 10 個輸出檔來對應到各個輸入檔。

輸出檔的第一行必須包含以下文字：

#FILE xor I

其中整數 I 代表本輸出檔對應的輸入檔編號。輸出檔的第二行含一個整數 K，代表本輸出檔所含的 XOR 指令個數。接著 K 行代表從頭到尾應執行的 K 個 XOR 指令。每一行含有 4 個整數值：依序代表一個 XOR 指令的 L, R, T, B 參數。

輸入輸出範例

Example: xor0.in

```
7
0 0 0 0 0 0 0
0 1 1 1 0 0 0
1 0 0 1 0 0 0
1 0 1 0 1 1 0
1 0 1 0 1 1 0
0 1 0 0 1 1 0
0 0 1 1 1 1 0
```

xor0.out

```
#FILE xor 0
3
2 4 2 6
3 6 4 7
1 3 3 5
```

評分方式

若

- 你的輸出檔所指定的 XOR 指令不能產生所要求的影像，或
- 你的輸出檔所指定的 XOR 指令個數不是 K，或
- 你的輸出檔的 K 值，K>40000，或
- 你的輸出檔所指定的 XOR 指令內的參數值為 L>R 或 T>B，或
- 你的輸出檔所指定的 XOR 指令內的參數值不是正數，或
- 你的輸出檔所指定的 XOR 指令內的參數值超過 N，

則 你將得到 0 分。否則 你將得到以下分數

1 + 9xCallsInBestAnswerOfAllContestants/CallsInYourAnswer

每個測試題的成績以四捨五入取小數點第一位；整題的成績將以四捨五入取到最接近的整數。

若你提出的輸出檔的解有 121 個 XOR 指令。而且此為所有競賽者的最佳解，你的成績為 10 分。若所有競賽者的最佳解為 98 個 XOR 指令，你的成績為 1+9x98/121(= 8.289 . . .)，其可四捨五入為 8. 3 分。

批次排程

問題說明

在一部機器上有 N 個工作要處理，這些工作從 1 到 N 編號；也即其順序為 1, 2, … , N。這些工作必須分為一個或數個批次來處理；每個批次內的工作具有連續的編號。機器從時間 0 開始處理工作，由第一批次開始，一批次接著一批次的處理方式如下：如果批次 b 所含的工作編號小於在批次 c 的工作編號，則會先處理 b。在每個批次內，機器依照工作編號依序處理。當該批次的所有工作全部處理完後，機器一次輸出該批次中所有工作的結果。工作 j 的輸出時間即為包含 j 之批次的結束時間。

每一批次執行前需要時間 S 來準備機器。對每一個工作 i，我們知道它的處理成本因素 F_i 和所需處理時間 T_i。如果一個批次包含工作 $x, x+1, \cdots, y-1, y$，且從時間點 t 開始處理，則那個批次中的每個工作的輸出時間是 $t+S+(T_x+T_{x+1}+\cdots+T_{y-1}+T_y)$。注意！機器會在同一時間同時輸出該批次的所有工作結果。如果工作 i 的輸出時間是 O_i，則它的處理成本是 $O_i \times F_i$。例如，假設有 5 個工作，準備時間為 $S = 1$，且 $(T_1, T_2, T_3, T_4, T_5) = (1, 3, 4, 2, 1)$，而 $(F_1, F_2, F_3, F_4, F_5) = (3, 2, 3, 3, 4)$。如果這些工作被分割成三個批次排程 {1, 2}, {3}, {4, 5}，則輸出時間 $(O_1, O_2, O_3, O_4, O_5) = (5, 5, 10, 14, 14)$，而工作的處理成本分別是 (15, 10, 30, 42, 56)。一組工作分割的總成本是所有工作的處理成本的總和。因此，上述分割例子中工作處理的總成本是 153 。

給定批次準備時間和一系列的工作，以及它們的處理時間和處理成本因素，請寫一個程式計算最小可能的總成本。

輸入

你的程式將由標準輸入讀入。第一行包含工作數 N ， $1 <= N <= 10000$ 。第二行有一整數，即為準備時間 S，且 $0 <= S <= 50$ 。隨後 N 行包含工作 1, 2, …, N 的資訊如下：首先在每一行有一個整數 T_i， $1 <= T_i <= 100$ ，為那個工作的處理時間。隨後，有一個整數 F_i ， $1 <= F_i <= 100$ ，此為那個工作的處理成本因素。

輸出

你的程式應寫到標準輸出。輸出只有一行，為一個整數，即最小可能總成本。

輸入及輸出範例

範例 1：　輸入　　輸出

```
2
50
100 100
100 100
```

```
45000
```

範例 2：　輸入　　輸出

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

```
153
```

範例 2 即為問題說明中所舉的例子。

註解

本題所有的測試資料中，任何分割的總成本不會超過 $2^{31} - 1$ 。

評分

如果你的程式在時間限制內輸出正確答案，則你得到那個測試組的滿分，否則得到 0 分。

公車站

問題說明

龍仁市計畫建造含有N個公車站的公車網路。每一個公車站都在街角。龍仁市是一個先進的城市，城市是由許多大小相同的方形區塊所組成，在所有的公車站中，有兩個公車站將會被選為轉運站 H_1 及 H_2。轉運站間將以直接公車連結，其他 N－2 個公車站將直接連到轉運站 H_1 或 H_2 （但不會同時直接連接到兩個轉運站)。這 N－2 個公車站彼此互不相連。

兩個公車站間的距離是沿著街道走的最短路徑。換句話說，如果一個公車站以其座標點來表示，則兩個公車站 （x_1，y_1） 與 （x_2，y_2） 的距離是 $|x_1-x_2|+|y_1-y_2|$。如果兩個公車站A及B都連到同一個轉運站（如H_1），則A與B之距離為A到此轉運站H_1的距離加上B到此轉運站H_1的距離。如果兩個公車站A及B連到不同的轉運站（例如：A連到H_1，B連到H_2），則A與B之距離為A到 H_1 的距離，加上 B 到 H_2 的距離，再加上 H_1 到 H_2 的距離。

龍仁市規劃當局希望確保每一個市民可以在最短的時間內到達任何一個地點。因此龍仁市規劃委員希望在 N 個公車站中挑選兩個站作為轉運站，目標是讓兩個距離最遠的公車站之間的距離越短越好。

給定兩組轉運站的選擇P及Q。如果在P中兩個距離最遠的公車站之間的距離小於在Q中兩個距離最遠的公車站之間的距離，則P優於Q。你的工作是要寫一個程式，在所有可能的轉運站選擇中，找出最優的轉運站選擇。

輸入

你的程式將由標準輸入讀取資料。第一行有一個正整數N（2 <= N <= 500)，代表公車站數目。再接下來的N行，每一行包含兩個小於或等於 5000 的正整數，第一個正整數代表公車站的x座標，第二個正整數代表公車站的y座標。沒有兩個公車站位於相同座標。

輸出

你的程式將由標準輸出印出結果。輸出將只有一行，此行只有一個正整數，其為在所有可能的選擇中，找出兩個站作為轉運站，讓兩個距離最遠的公車站之間的距離為最短。並印出此距離。

輸入與輸出範例

範例 1：

輸入

```
6
1 7
16 6
12 4
4 4
1 1
11 1
```

輸出

```
20
```

範例 2：

輸入

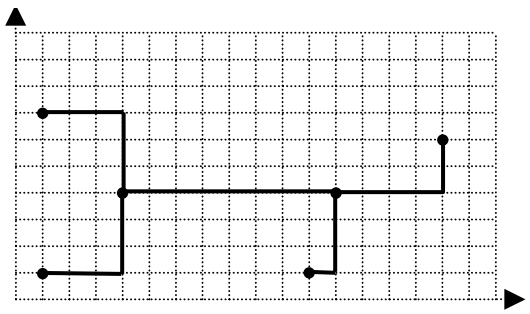
```
7
7 9
10 9
5 3
1 1
7 2
15 6
17 7
```

輸出

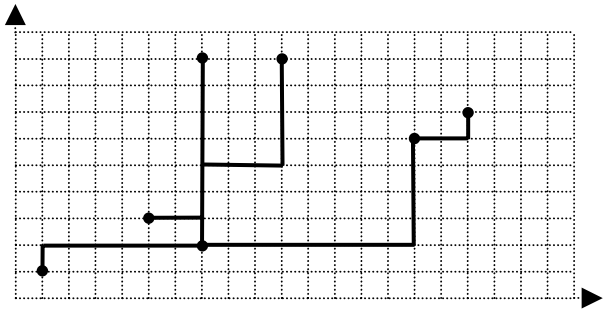
```
25
```

下列圖形顯示以上輸入範例之公車網路。如果範例１選擇公車站３及４作為轉運站，則公車站２及５為距離最遠的兩個站，同理公車站２及１亦為距離最遠的兩個站。在此並無更佳的轉運站選擇，所以本題應輸出２０。

以範例２而言，如果選擇公車站５及６作為轉運站，則公車站２及７為距離最遠的兩個站，在此並無更佳的轉運站選擇，所以本題應輸出２５。



範例１之公車網路



範例２之公車網路

計分

如果你的程式在時間限制內正確回答一個測試資料，則你可以得到該測試資料的全部分數，否則得到０分。

兩段繩索

問題說明

假設一段繩索(rod)為在水平或垂直方向的至少連續兩個方格所構成。在一個 NxN 的棋盤上有兩段繩索，一段為水平方向，另一段為垂直方向。在 Figure-1 中的兩段繩索皆以 X 表示之。兩段繩索可能長度相同，也可能長度不同，而且也可以同時含同一個方格。如 Figure-1 中的(4,4)方格，此方格可能被解釋為包含在一段繩索中也可能被解釋為包含在兩段繩索中。本題採取的解釋為(4,4)包含在兩段繩索中，因此垂直繩索的上端方格是(4,4)而非(5,4)。

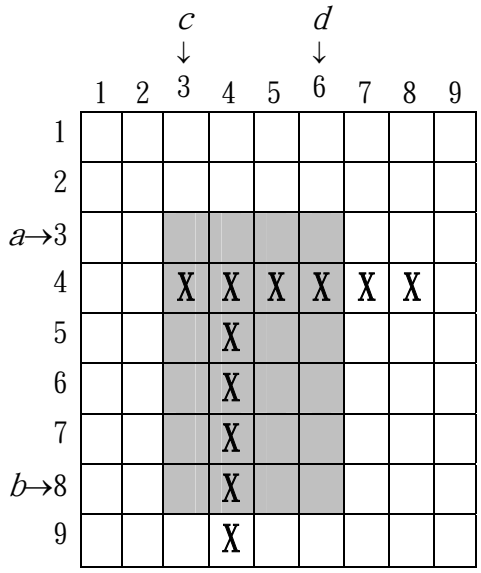


Figure-1

一開始我們並不知道兩段繩索的位置，你的工作是寫一個程式來找出他們的位置。我們稱水平繩索為ROD1，垂直繩索為ROD2。棋盤上每一個方格皆以一個列(row)/行(column)對(r_1, c_1)表示之，棋盤的左上方為(1,1)。每段繩索以兩個方格表示之，<(r_1, c_1), (r_2, c_2)>。在Figure-1 中ROD1 為<(4,3),(4,8)>，ROD2 為<(4,4),(9,4)>。

本題需要用函式庫來做輸入、解題計算、和輸出工作。棋盤的一邊長度可以用函式gridsize取得，你的程式在每個測試題一開始時必須先使用此函式。在找繩索的位置時，你只能使用函式rect(a, b, c, d)，其功能為檢測範圍為$[a, b]\times[c, d]$的長方形區域(在Figure-1的灰色區域)，其中$a \le b$且$c \le d$，若在$[a, b]\times[c, d]$ 檢測範圍內有至少有一個X(註：任一繩索的一部份)，則rect回傳1；否則回傳0。你必須寫一個程式，利用有限次數的rect呼叫，來找出兩段繩索的正確位置。

你必須呼叫另一個函式report(r_1, c_1, r_2, c_2, p_1, q_1, p_2, q_2)來產生輸出，其中ROD1 為<(r_1, c_1), (r_2, c_2)>，ROD2 為<(p_1, q_1),(p_2, q_2)>.。呼叫report可結束你的程式。請注意ROD1 為水平繩索，ROD2 為垂直繩索，且(r_1, c_1)為水平繩索ROD1 的左端點方格，(p_1, q_1) 為垂直繩索ROD2 的上端點方格。因此$r_1 = r_2$， $c_1 < c_2$，$p_1 < p_2$，且 $q_1 = q_2$。若你的report參數不符合以上的限制，你將會在標準輸出(standard output)上看到錯誤訊息。

限制

- 你只能用內建函式 gridsize 和 rect 來取得輸入資料。
- N，棋盤的最大行數(列數)滿足 $5 \le N \le 10000$ 的範圍。
- 在每個測試題中，至多呼叫 rect 函式 400 次。若你的程式呼叫 rect 超過 400 次，系統會自動結束你的程式。
- 你的程式必須呼叫 rect 至少一次，且呼叫 report 正好一次。
- 若呼叫 rect 不正確(譬如，查詢的範圍超出棋盤範圍)，此呼叫會結束你的程式。
- 你的程式不能讀入或寫出任何檔案，且不能使用任何標準輸入/輸出。

函式庫

你可使用下述的函式庫:

FreePascal Library (prectlib.ppu, prectlib.o)

```
function gridsize: LongInt;
function rect(a,b,c,d : LongInt) : LongInt;
procedure report(r1, c1, r2, c2, p1, q1, p2, q2 : LongInt);
```

指示: 要編譯你的 rods.pas,可包含此 import 敘述：

```
 uses prectlib;
```

放在原始碼(source code)內，並以下述指令編譯你的原始碼：

```
 fpc -So -O2 -XS rods.pas
```

下列程式 prodstool.pas 顯示使用 FreePascal 函式庫的範例：

GNU C/C++ Library (crectlib.h, crectlib.o)

```
int gridsize();
int rect(int a, int b, int c, int d);
void report(int r1, int c1, int r2, int c2, int p1, int q1,
            int p2, int q2);
```

指示: 要編譯你的 rods.c，使用

```
 #include “crectlib.h”
```

放在原始碼(source code)內，並以下述指令編譯你的原始碼：

```
 gcc -O2 -static rods.c crectlib.o -lm
 g++ -O2 -static rods.cpp crectlib.o -lm
```

下列程式 crodstool.c 顯示使用 GNU C/C++ 函式庫的範例：

在 RHIDE 環境使用 C/C++

要確定你將 Option->Linker configuration 設定為 crectlib.o .

實驗

要實驗函式庫，你必須建一個文字檔rod.in。此檔案必須含三行：第一行含一個整數：N，棋盤的大小；第二行含ROD1 的位置座標r_1 c_1 r_2 c_2，其中(r_1, c_1)為ROD1 的左端點方格；第三行含ROD2 的位置座標p_1 q_1 p_2 q_2，其中(p_1, q_1) 為ROD2 的上端點方格。

你的程式在執行呼叫過 report 後，可得到 rods.out 輸出檔。此檔含有你的程式呼叫 rect 的次數和你在呼叫 report 函式時所用的兩段繩索的位置值。若在呼叫函式時，違反上述的限制要求，則 rods.out 會含相對應的錯誤訊息。

你的程式和函式庫的對話記錄在 rods.log 檔案。此記錄檔 rods.log 依序顯示你的程式所做的呼叫，其格式為" *k* : rect(*a, b, c, d*) = *ans*" ，表示第 k 個 rect(*a, b, c, d*) 函式呼叫回復為 *ans*。

輸入及輸出範例

Example: rods.in

```
9
4 3 4 8
4 4 9 4
```

rods.out

```
20
4 3 4 8
4 4 9 4
```

評分方式

若你的程式違反任何上述的限制(譬如，超過 400 次函式呼叫)，或者你的程式輸出(兩段繩索的位置)不正確，則你將得到 0 分。

若你的程式輸出正確，則你的成績會依照每個測試題的函式呼叫個數來判斷。在每個測試題中，若函式呼叫個數至多 100 次，則你可得到 5 分。若你的程式使用 101 到 200 次函式呼叫，則你可得到 3 分。若你的程式使用 201 到 400 次函式呼叫，則你可得到 1 分。