

# 重要的競賽指南!!



請仔細的閱讀以下的指南。它包含了將如何打包和提交你的答案給裁判的重要資訊。如果對於這份指南有任何問題，請在競賽開始前提問。

## 程式輸入/輸出:

需要互動的程式(task)應該要即時在螢幕上(STDOUT)要求使用者輸入，並且從鍵盤/螢幕讀取輸入(STDIN)。

對於不需要互動的程式(task)，你會有二種選擇來讀取輸入資料。在一些題目當中，名為**probXX.txt**的檔案將會被提供來作為樣本輸入，'XX'代表問題的號碼。

你的解答可能會需要有系統地從檔案 **probXX.txt** 讀取輸入。也就是說使用程式語言中的File I/O架構。

大部分的問題會接受直接由鍵盤來輸入(STDIN) 來取代檔案運作。對於需要很多輸入的程式，這樣就變得很冗長乏味。然而，一個簡單的方式是在執行的時候借由導入檔案的內容資料到你的程式中。例如：一個名為 **prob01.txt** 的檔案可以用下面的語法直接從STDIN輸入到你的程式當中：

```
%> java prob01 < prob01.txt
%> java -jar js.jar prob01.js < prob01.txt
%> python prob01.py < prob01.txt
%> prob01.exe < prob01.txt
```

在這個例題裡，你正執行著 **prob01** 以及把 **prob01.txt**檔案中的資料傳送到你程式中的STDIN。你的程式行為就會如同你從鍵盤鍵入你的輸入一樣。

*提示：當使用鍵盤直接輸入時，請使用'Ctrl-Z' <return> 來結束你的程式輸入。*

所有程式的輸出應該都要傳送到螢幕上(STDOUT)。

## 提交你的程式

**Interpreted Programs (JAVA, JavaScript, Python)**。你的程式一定要用 **probXX.java / probXX.js / probXX.py**來命名，'XX'代表問題的號碼。請僅提交原始碼(.java、.js 或 .py)。Java的主要類別一定要命名為 **probXX**。請注意大小寫。所有主要和支援類別都應該包含在預設的(或匿名的)套件裡。

**Native Programs (C, C++... etc)**。你的程式應該用 **probXX.exe**來命名，'XX'代表問題的號碼。

**強烈地建議你在開始競賽之前，**

**先提交問題#0 (在下一頁上) 以確保你的編譯環境和裁判的是相容的。**

## 前言

這個題目的主要目的是讓每一個參賽團隊試著遞交一個測試程式，以確保產生出來的結果可以在評審的電腦上正確執行。強烈建議每一個參賽團隊先遞交這題。

這題改編自經典的「Hello World!」程式，請印出「Hello HP CodeWars 2015 Taipei!」。

## 輸入

[這題沒有輸入。]

## 輸出

Hello HP CodeWars 2015 Taipei!

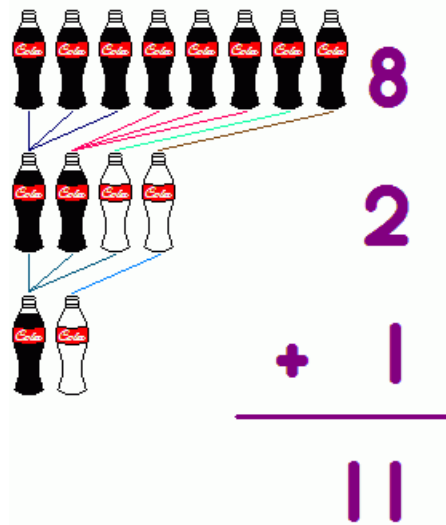
問題 1  
可樂活動  
3 分

前言

可樂最多喝幾瓶。

可樂優惠活動如下：「3 瓶空可樂罐換一瓶可樂。」

下圖說明當一開始拿到 8 瓶可樂的情況。在喝了 8 瓶可樂之後，有 8 個空罐子，把其中 6 瓶空罐子換成 2 瓶新的，喝完之後你就有 4 個空罐子，所以又可以拿 3 瓶換 1 瓶新的。最後，只剩下 2 個空罐子，所以你再也不能再換新的可樂了。所以最多總共可以喝到  $8 + 2 + 1 = 11$  瓶可樂。



## 輸入

輸入為一開始拿到的可樂，最多 100 瓶。

**實例1:** 7

**實例2:** 8

**實例3:** 9

## 輸出

請輸出最多可以喝到幾瓶可樂。

**實例1:** 10

**實例2:** 11

**實例3:** 13

## 問題 2

# 天干和地支

3 分

### 前言

天干和地支合稱干支。

十天干分別是甲(0)、乙(1)、丙(2)、丁(3)、戊(4)、己(5)、庚(6)、辛(7)、壬(8)、癸(9)。

十二地支是子(A)、丑(B)、寅(C)、卯(D)、辰(E)、巳(F)、午(G)、未(H)、申(I)、酉(J)、戌(K)、亥(L)。

天干和地支組合便成為以「甲子」為首的六十干支循環。

1	2	3	4	5	6	7	8	9	10	11	12
甲子	乙丑	丙寅	丁卯	戊辰	己巳	庚午	辛未	壬申	癸酉	甲戌	乙亥
0A	1B	2C	3D	4E	5F	6G	7H	8I	9J	0K	1L
13	14	15	16	17	18	19	20	21	22	23	24
丙子	丁丑	戊寅	己卯	庚辰	辛巳	壬午	癸未	甲申	乙酉	丙戌	丁亥
2A	3B	4C	5D	6E	7F	8G	9H	0I	1J	2K	3L
25	26	27	28	29	30	31	32	33	34	35	36
戊子	己丑	庚寅	辛卯	壬辰	癸巳	甲午	乙未	丙申	丁酉	戊戌	己亥
4A	5B	6C	7D	8E	9F	0G	1H	2I	3J	4K	5L
37	38	39	40	41	42	43	44	45	46	47	48
庚子	辛丑	壬寅	癸卯	甲辰	乙巳	丙午	丁未	戊申	己酉	庚戌	辛亥
6A	7B	8C	9D	0E	1F	2G	3H	4I	5J	6K	7L
49	50	51	52	53	54	55	56	57	58	59	60
壬子	癸丑	甲寅	乙卯	丙辰	丁巳	戊午	己未	庚申	辛酉	壬戌	癸亥
8A	9B	0C	1D	2E	3F	4G	5H	6I	7J	8K	9L

提示：已知西元年 2015 為農曆乙未 (1H)

### 輸入

輸入一個參數。代表西元年，範圍從 0 到 3000 年。

實例1： 2015

實例2： 1874

實例3： 1900

### 輸出

請輸出農曆年，使用阿拉伯數字及英文大寫表示。

實例1： 1H

實例2： 0K

實例3： 6A

# 問題 3

## 員工號編碼原則

3 分

### 前言

#### 簡易編碼

由於 HP 的員工人數在全世界已經高達 30 萬人，為了更有效率的管理員工，HP 推出一個新的機制來配發員工編號，這個機制是由 8 個數字和一個英文字母所組成 (EX: A12345678)，此英文字母為所在的區域，如 A 表示 America，E 表示 Europe，P 表示 Asia-Pacific, M 表示 Middle East，而後面的 8 個數字並也不是隨便配發的，其中最後一個數字為檢查碼，此檢查碼主要為了驗證前面七個數字經過運算後是否符合運算原則，請同學們根據此運算原則幫 HP 寫出一個簡易的認證員工編號的程式，如果輸入的編碼是符合運算原則，程式將輸出 Success，反之，若輸入的員工編碼格式錯誤或是不符合運算原則，程式將輸出 Fail。

下面為員工編號的運算原則：

1. 英文代號請根據以下列表轉成數字。  
A (America) = 10  
E (Europe) = 11  
P (Asia-Pacific) = 12  
M (Middle East) = 13
2. 將英文文轉成數字，個位數乘8加上十位數。
3. 個數字左倒右依次乘1,2,3,4,5,6,7。
4. 將第2步驟的數字和第3步驟的數字相加。
5. 將第4步驟所得到的和除以10得到餘數，再用10減去餘數，結果就為檢察碼，若餘數為0，則檢查碼就是0。

例如員工編號為：P23273626

$$P = 12$$

$$2 \times 8 + 1 = 17$$

$$2 \times 1 + 3 \times 2 + 2 \times 3 + 7 \times 4 + 3 \times 5 + 6 \times 6 + 2 \times 7 = 107$$

$$107 + 17 = 124 \text{ (數字加英文數字的總和)}$$

$$124 \% 10 = 4 \text{ (餘數)}$$

$$10 - 4 = 6 \text{ (驗證碼)}$$

## 輸入

程式必須輸入一組員工編號，第一個字為英文字母，後面八碼為數字。

**實例1:** A12343046

**實例2:** P39290493

**實例3:** K17367820

## 輸出

程式必須驗證所輸入的員工編號，如果驗證成功則印出 Success，若失敗則印出 Fail。

**實例1:** Success

**實例2:** Success

**實例3:** Fail

# 問題 4

## 三角形與點

3 分

### 前言

點是否在三角形內

在不考慮退化三角形 (面積為 0) 的情況下，3 個座標點可以組成一個三角形，而若有一第 4 個座標點，它必然會在三角形之內或之外。請利用使用者輸入的 4 個座標點，判斷第 4 個座標點是否存在於前 3 個座標點組成的三角形內。若前 3 點無法組成三角形，請輸出此三角形不存在。若第 4 個座標點在三角形的邊上，則判斷此點不在三角形內。

### 輸入

4 個座標，座標格式為(\*,\*)，每個座標以空格分開，座標值皆為正整數或 0。

**實例1:** (0,0) (6,0) (3,4) (3,3)

**實例2:** (5,5) (2,2) (6,0) (7,0)

**實例3:** (3,3) (6,6) (9,9) (1,1)

### 輸出

第 4 個點是否在前 3 個點組成的三角形內，若在三角形內，顯示” Point is within the Triangle.” 若不在三角形內，顯示” Point is NOT within the Triangle.” 若前 3 點無法組成三角形，顯示” Triangle doesn't exist.”

**實例1:** Point is within the Triangle.

**實例2:** Point is NOT within the Triangle.

**實例3:** Triangle doesn't exist.



# 問題 5

## 機密文件處理

3 分

### 前言

請提供一個程式幫助使用者自動塗改選定字串。  
使用者可以自己輸入 1 到 10 個想要塗改的字串。  
這個程式會讀取原始文字檔案內容並新增指定字串以星號 "\*\*\*\*" 隱藏的文字檔。

注意事項:

1. 本程式只需要處理英文文章。
2. 無論隱藏字串的長度請都以三個星號 \*\*\* 來取代…。
3. 隱藏字串有分大小寫。
4. 指定原始檔案與新存檔案皆為 .txt 檔。
5. 新存檔案名稱指定為原始檔名加上 "\_mod" 例如 sample.txt 變成 sample\_mod.txt。
6. Input 格式: 原始檔案名 指定字串總數 指定字串 (以空格分割)。

## 輸入

### 實例1:

Input:  
Sample.txt 1 John

使用者輸入一個隱藏字串為 John

原始內文:  
Mary is upset about what John did.

### 實例2:

Input:  
S2.txt 4 HP ProBook performance comprehensive  
使用者輸入四個隱藏字串HP, ProBook, performance, comprehensive

原始內文:  
The sleek HP ProBook 450 with optional 10-point touchscreen and the latest generation technologies delivers powerful performance in the office or on the go. Proven reliability and comprehensive HP security features help keep your data and investment protected.

## 輸出

### 實例1:

Output:  
Success: Sample\_mod.txt

修改後內文:  
Mary is upset about what \*\*\* did.

### 實例2:

Output:  
Success: S2\_mod.txt

修改後內文:  
The sleek \*\*\* \*\* 450 with optional 10-point touchscreen and the latest generation technologies delivers powerful \*\*\* in the office or on the go. Proven reliability and \*\*\* \*\* security features help keep your data and investment protected.

前言

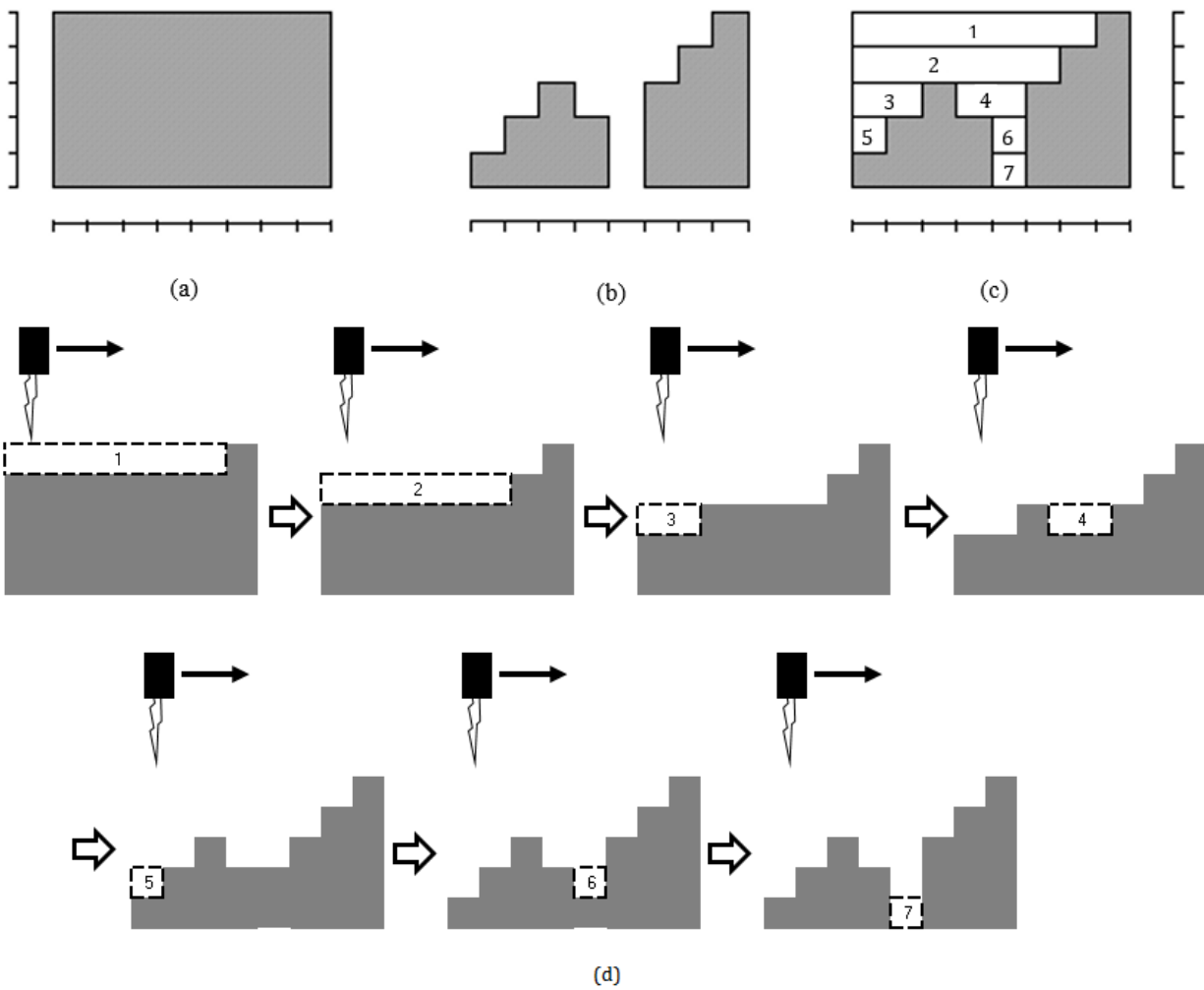
雷射最少被開幾次。

雷射頭向左右移動。在移動的過程中，若雷射光被開啟，則雷射會除去物體上，一層單位高度的物質。

注意：每次雷射光開啟僅能除去同一水平層的物質。

下面的圖則顯示出雷射雕刻的過程：

1. 圖 (a) 有一高 5 mm、長 8 mm 待雕刻的物體方塊。
2. 圖 (b) 則是我們最後要雕刻出來的樣式。
3. 圖 (c) 則展示出物體被雕刻的全部過程。  
首先標示 1 的被移除，接下來標示 2 的被移除，接下來標示 3 的被移除，依此進行下去。
4. 圖 (d) 分解 7 次雷射雕刻過程。在這個過程中，雷射光共被打開 7 次。



## 輸入

每筆測資的一開始有 2 個整數，分別是高(H)及長(W)。(1 <= H <= 10, 1 <= W <= 10)  
接下來是 N 個整數，對應雷射結果每一個長度最後的高度(由左至右)。

**實例1(如圖b):** 5 8 1 2 3 2 0 3 4 5

**實例2:** 3 3 1 0 2

**實例3:** 4 3 4 4 1

## 輸出

請輸出雷射最少要被開幾次，可以達到最後指定的高度。

**實例1:** 7

**實例2:** 3

**實例3:** 3

# 問題 7

## 數字遊戲

5 points

### 前言

數字遊戲程式可輸入兩組號碼(每一組號碼的數字不會重複)，第一組號碼是對照組，第二組號碼是實驗組。程式需要拿實驗組的號碼比對對照組。A 與 B 的初始值為 0，假如數字存在且位置一致則 A 變數值加一，假如數字存在但位置不一致則 B 變數值加一。當所有數字比完後把 A 變數值和 B 變數以十進制顯示出來(如實例)。

### 輸入

輸入兩組位數相同之整數(十進制)。第一個值代表對照組號碼，第二個值代表實驗組號碼。

**實例1:** 12345 01324

**實例2:** 987654 197324

**實例3:** 1234567890 0987654321

### 輸出

程式必須印出 AB 變數值。

**實例1:** 1A3B

**實例2:** 2A1B

**實例3:** 0A10B

## 前言

HP 團隊的同事們想要進行交換禮物派對，他們打算用鬼腳圖決定每個人的禮物。鬼腳圖的玩法如下：每個人選一起點開始往下走，遇到橫線則沿著橫線走到隔壁的縱線，最後到達終點就是所抽中的項目。如圖 1 所示，C 同事最後拿到了 4 號禮物。

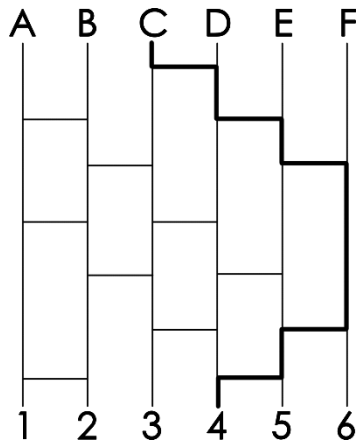


圖 1、六人的鬼腳圖

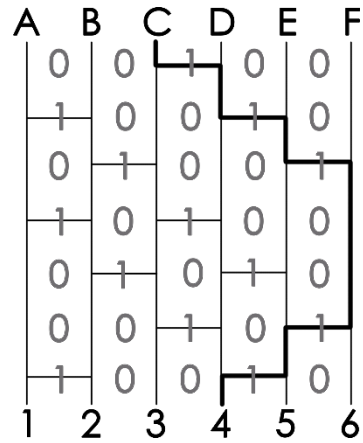


圖 2、鬼腳圖與輸入範例

## 輸入

輸入六個不同字母，鬼腳圖長度，以及一張鬼腳圖，其格式如下：0 代表兩條縱線之間沒有橫線，1 代表兩條縱線之間有橫線。不會有三條縱線間有兩條相鄰橫線的情形，以圖 1 的鬼腳圖為例，其長度為 7，輸入為：

```
00100
10010
01001
10100
01010
00101
10010
```

鬼腳圖輸入與圖 1 的對應關係請見圖 2。

### 實例 1:

```
ABCDEF
1
10100
```

### 實例 2:

```
QWERTY
6
10101
01010
10101
```

00101  
01000  
01010

**實例3:**

ASDFGH  
7  
00100  
10010  
01001  
10100  
01010  
00101  
10010

**輸出**

鬼腳圖底部由左至右的字母順序。

**實例1:** BADCEF

**實例2:** RWQEYT

**實例3:** GFHDSA

# 問題 9

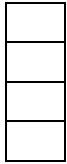
## 惠普之塔

8 分

### 前言

要建一個高度為  $n$  個磚塊的惠普之塔，建設工人一次可以堆 1 個、2 個、3 個磚塊。請問，辛苦的建设工人總共有多少種可能的堆疊方法？請實作程式計算出來。

以高度為 4 個磚塊的惠普之塔為例



可能的堆疊方式如下所示：

- 第 1 種：1, 1, 1, 1 (分成四次，每次堆 1 個。)
- 第 2 種：2, 2 (分成兩次，每次堆 2 個。)
- 第 3 種：1, 1, 2 (分成三次，第一次堆 1 個，第二次堆 1 個，第三次堆 2 個。)
- 第 4 種：2, 1, 1 (分成三次，第一次堆 2 個，第二次堆 1 個，第三次堆 1 個。)
- 第 5 種：1, 2, 1 (分成三次，第一次堆 1 個，第二次堆 2 個，第三次堆 1 個。)
- 第 6 種：1, 3 (分成兩次，第一次堆 1 個，第二次堆 3 個。)
- 第 7 種：3, 1 (分成兩次，第一次堆 3 個，第二次堆 1 個。)

共有 7 種堆疊方式。

### 輸入

輸入  $n$ ， $n$  為一介於 0 到 30 之間的正整數。

**實例 1:** 8  
**實例 2:** 29

### 輸出

輸出為可能的堆疊方法數。

**實例 1:** 81  
**實例 2:** 29249425



# 問題 10

## 最大整數計算

8 分

### 前言

給定  $N$  個正整數。請利用串接的方式使這  $N$  個正整數連接而成的數字最大。

### 輸入

輸入包括數個正整數。第一個整數代表正整數的數量，爾後分別代表各個正整數。

**實例1:** 2 123 124

**實例2:** 3 54 595 599

**實例3:** 5 1761 176 1769 17 175

### 輸出

程式計算後所得由正整數串接而成的最大數。

**實例1:** 124123

**實例2:** 59959554

**實例3:** 1769176117617517

# 問題 11

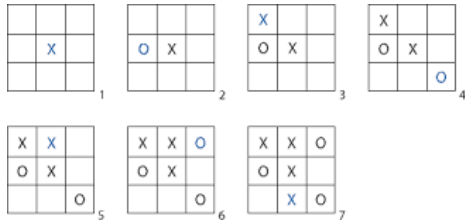
## 井字遊戲

8 分

### 前言

井字遊戲相信大家都知道怎麼玩，不知道的請參考下面這段介紹：

兩個玩家，一個打圈(O)，一個打叉(X)，  
輪流在 3 乘 3 的格上打自己的符號，  
最先以橫、直、斜連成一線則為勝。



我方先下而且使用的符號為 O，對手後下並且使用的符號為 X。  
給一個進行到一半的棋局。請判斷是否有機會在我方下兩步棋內獲勝。

### 輸入

輸入為九個字元，前三個代表第一行，中間三個代表第二行，最後三個代表第三行。O 代表我方下的棋子，X 代表對手棋子，A 表示尚未下過的位置。

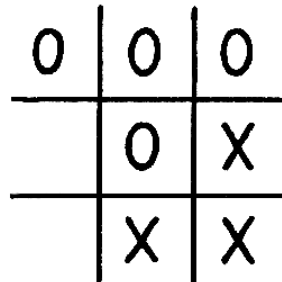
右圖的棋局等於 OOOAOXAXX

若有機會能夠獲勝則輸出為 yes  
其他則輸出為 no

實例 1: AXOAOAXOX

實例 2: OXOXXAOAO

實例 3: AOAAXXOOX



### 輸出

實例 1: yes

實例 2: no

實例 3: no

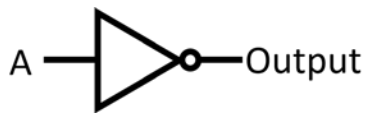
前言

基礎知識建立：

電腦運算皆採用 0 與 1 兩個數字訊號來傳遞與運算，我們稱該傳遞方式為數位訊號。

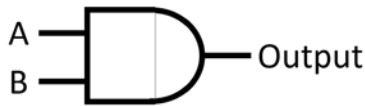
筆記型電腦中的硬體電路是由許多電路元件來設計結合而成，如反閘 (Not Gate)、集閘 (AND Gate) 與或閘 (OR Gate) 等元件皆是常見數位電路基礎元件。

1. 反閘 (NOT Gate)：假設輸入訊號為 0，則輸出訊號就為 1。
2. 集閘 (AND Gate)：將兩輸入訊號做 AND 運算，必須要兩輸入訊號皆為 1，輸出訊號才會為 1，其他情況皆為 0。
3. 或閘 (OR Gate)：將兩輸入訊號做 OR 運算，只要兩輸入訊號有一者為 1，則輸出訊號就為 1。只有當兩輸入訊號皆為 0，輸出訊號才會為 0。



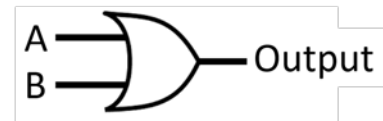
1. 反閘(Not Gate)真值表

輸入	輸出
A	Output
0	1
1	0



2. 集閘(AND Gate)真值表

輸入		輸出
A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



3. 或閘(OR Gate)真值表

輸入		輸出
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

舉例：

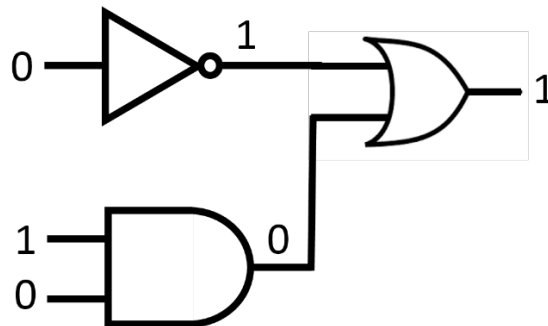


圖 1、範例電路圖

問題介紹：

HP 工程師將一台筆電內部電路進行硬體升級更新，該電路需要進行功能測試。  
現在想要請你透過程式將該電路實現出來，並將測試值輸入(ABCDEFGH)，測得該電路所對應之輸出訊號(abcd)。詳細電路圖如下圖 2 所示。

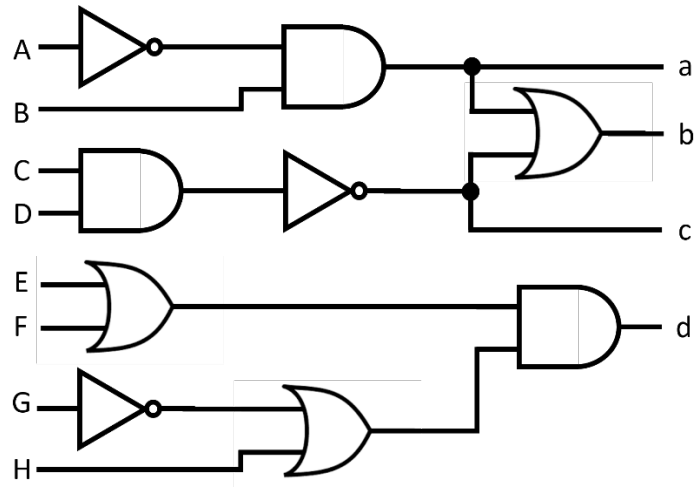


圖 2、待測電路圖

## 輸入

輸入一行八位元的二進制數值(從左至右為 ABCDEFGH)。

ABCDEFGH

**實例1:** 01001010

**實例2:** 11110010

**實例3:** 10100011

## 輸出

程式必須輸出電路測試結果，該值為四位元的二進制值(從左至右為 abcd)。

abcd

**實例1:** 1111

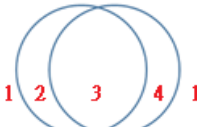

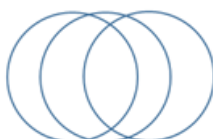
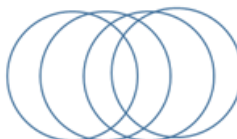
**實例2:** 0000

**實例3:** 0110

問題 13  
算圓切割區域  
12 分

前言

給定  $n$  個圓，請求出這  $n$  個圓最多可切割出多少區域數  $R$ 。

$N=2$ $R=4$ 	$N=5$ $R=22$ 
$N=3$ $R=8$ 	
$N=4$ $R=14$ 	

舉例 1：2 個圓，可最多切割出 4 個區域。

舉例 2：3 個圓，可最多切割出 8 個區域。

請寫一程式計算  $n$  個圓最多可切割出多少區域數  $R$ 。

輸入

輸入包括一個整數。代表圓的個數。

實例 1： 2

實例 2： 3

實例 3： 4

輸出

程式必須印出一個整數代表最多可切割出的區域數。

實例 1： 4

實例 2： 8

實例 3： 14

## 前言

### 系統驅動程式安裝移除檢查器

電腦系統的驅動程式與應用程式經常伴隨著相依性。於服務發揮功能之前，絕大多數的系統元件或驅動程式必須要預先正常安裝。系統中某些部分的驅動程式經常是共享的，例如 EMAIL 與 HTTP 網路程式需要先安裝 TCP/IP 驅動程式。假如 EMAIL 與 HTTP 服務程式已經安裝完成，當你要使用 HTTP 服務程式的時候，就不需要安裝 TCP/IP 驅動程式了。能夠移除暫時無用的原件對電腦系統是很有幫助的，除了可留下更多磁碟空間，釋放記憶體與其他資源。

然而任何系統的元件要被解除安裝，前提是相關的驅動程式也都移除。例如移除 HTTP 網路程式以及 TCP/IP 驅動程式，連帶地 EMAIL 將也一併無法正常發揮作用，因為 TCP/IP 是 HTTP 與 E-MAIL 所共用的通訊協定。同樣地，移除 TCP/IP 本身也會造成 EMAIL 與 HTTP 無法發揮功能。

現在有一個情況發生了，假如電腦使用者僅移除 EMAIL 程式以節省電腦資源，但是也使得 TCP/IP 驅動程式一併消失，如此很多其他共用 TCP/IP 的網路程式受影響。實際上我們並不想讓 TCP/IP 被意外移除。我們必須設計一個程式能夠自動地處理安裝與移除應用程式或驅動程式，並且能夠判斷應用程式或驅動程式之間的相依性，以避免意外地被刪除。

## 輸入

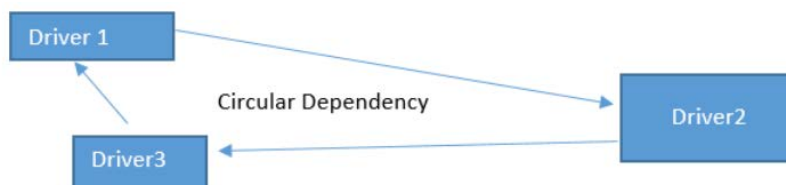
能夠輸入一連串的命令，一行總共不能超過 80 個字元。一行內的每一個指令或項目都是最多 16 個字元組成的字串。電腦系統支援的指令(皆大寫)如下，必須出現在每一行輸入的第一個欄位，

### DEPEND, INSTALL, REMOVE, LIST

每一個項目或指令可以用一個或一個以上的空白字元隔開。

所有 **DEPEND** 指令必須在 **INSTALL** 指令執行之前完成。

### DEPEND 指令應避免造成 Circular Dependency



**END** 指令代表輸入結束並完成程式運作。

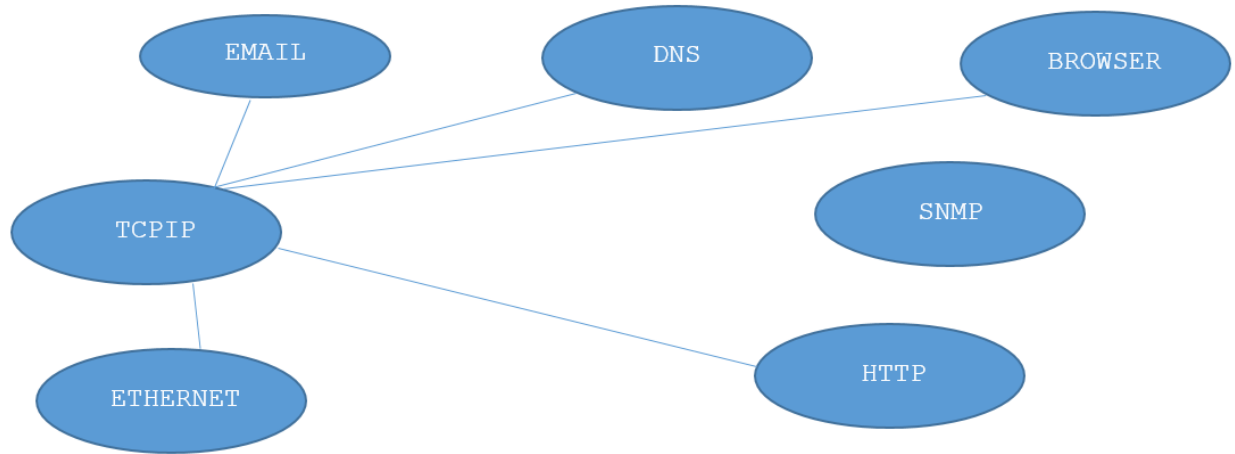
指令語法	描述/反應
<b>DEPEND</b> <u>item1</u> <u>item2</u> <u>item3</u> ...	<u>item1</u> 依賴 <u>item2</u> (and <u>item3</u> ...)
<b>INSTALL</b> <u>item1</u>	<p>安裝 <u>item1</u> 以及其所需要的各樣驅動程式 如有遇到重複安裝情況且尚未移除該元件，則將忽略重複的<b>INSTALL</b>指令。</p> <p>例: 下列第二個<b>INSTALL</b>指令將被忽略</p> <pre><b>INSTALL XYZ</b> ... ... I <b>INSTALL XYZ</b></pre>
<b>REMOVE</b> <u>item1</u>	<p>1. 移除 <u>item1</u> 以及其所需要用的驅動程式，如有牴觸相依性，則保留且不移除。</p> <p>例: 下列<b>REMOVE DRIVER2</b>指令將被忽略</p> <pre><b>DEPEND DRIVER1 DRIVER2</b> <b>DEPEND DRIVER3 DRIVER4 DRIVER2</b> ... ... <b>REMOVE DRIVER2</b> ...</pre> <p>2. 移除不存在的驅動程式或應用程式，也將被忽略。</p>
<b>LIST</b>	列出目前系統中安裝的元件

## 輸出

每一行指令最後須有換行。在每一個 **INSTALL** 或 **REMOVE** 指令之後，系統必須反應執行指令。系統必須能夠辨識出例外條件(參考下列範例)。 **LIST** 指令必須印出目前系統已經安裝的元件(如驅動程式或應用程式)。

**DEPEND** 與 **END**，除了換行指令之外，系統不需任何印出反應。

## 範例



## 輸入範例

DEPEND EMAIL TCPIP ETHERNET

DEPEND TCPIP ETHERNET

DEPEND DNS TCPIP ETHERNET

DEPEND BROWSER TCPIP HTTP

INSTALL ETHERNET

INSTALL EMAIL

INSTALL SNMP

REMOVE ETHERNET

INSTALL BROWSER

INSTALL DNS

REMOVE EMAIL

REMOVE ETHERNET

REMOVE DNS

REMOVE ETHERNET

INSTALL ETHERNET

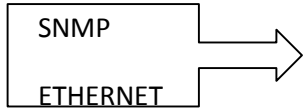
REMOVE TCPIP



REMOVE BROWSER

REMOVE TCPIP

LIST



輸出範例

END

## 前言

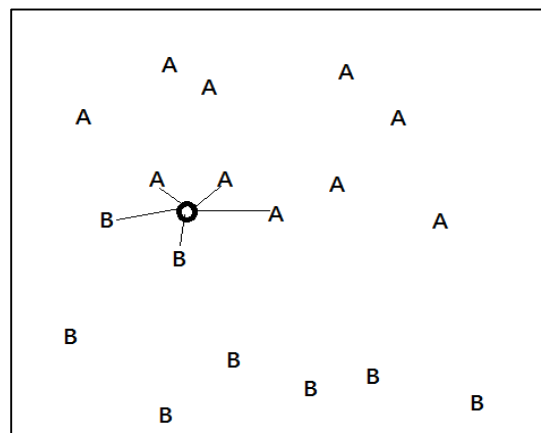
在人類的生活中已經有許多用電腦來進行預測或是分類的例子。例如說天氣預測，垃圾信辨別，證券交易或是人臉辨識等等。這個領域的研究叫做機器學習。今天我們就來試試一個基本的機器學習的方法。進行預測之前我們會先拿到一組含有多筆資料的樣本。再基於樣本來進行預測與分類。

我們要用的方法叫做”第 k 位最近的鄰居”演算法。

請參考下圖，在一個座標系中有許多的點。

這些點分為兩個組別 (A 跟 B)。

今天加入了一個**新的圓點**，而我們該如何預測此新的點應該為 A 還是 B 呢？



圖中的例子我們搜尋了離**新的圓點**直線距離最近的 5 位鄰居 ( $k = 5$ ) 並且統計在這些鄰居中屬於哪個組別的人最多。右圖中有三位 A 兩位 B，因此我們判定新的圓點為 A 組。

兩個座標的直線距離公式為  $d = \sqrt{(\Delta x)^2 + (\Delta y)^2}$ 。

其中  $\Delta x$  為兩座標之間的 x 座標值的差， $\Delta y$  為兩座標之間的 y 座標值的差。

d 為最後求出的距離。

## 輸入

輸入為多個參數且數量不定。每一組輸入包含三個值  $x, y, z$ 。  $x$  跟  $y$  代表座標，  $z$  的值為此座標的已知分類，可能的值有  $a$  或  $b$ 。最後會有一組座標沒有  $z$  的值，請求出此座標的分類。所有座標的  $x, y$  值皆為正整數，而且當  $k = N$  的時候，不會有因為多個點距離相同導致需要考慮大於  $N$  個點的情況。

請依據題目輸出  $k = 3$  和  $k = 5$  時的分類為何。

### 實例1:

```
1 2 a
2 1 a
98 99 b
24 56 b
75 34 b
3 3
```

### 實例2:

```
0 5 a
2 5 a
3 5 a
11 b
1 2 b
2 2 b
3 3 b
2 3
```

### 實例3:

```
2 4 a
3 3 a
4 2 b
3 4 b
4 3 b
6 6 a
5 5
```

## 輸出

### 實例1:

```
ab
a代表(k=3的解答), b代表(k=5 的解答)
```

### 實例2:

```
bb
b代表(k=3的解答), b代表(k=5 的解答)
```

### 實例3:

```
ba
b代表(k=3的解答), a代表(k=5 的解答)
```

## 前言

格雷碼 (Gray Code) 是一種二進位的數列，其規則為任相鄰兩數之間只會有一個位元不同。

3 位元的格雷碼：

```
000 001 011 010 110 111 101 100
```

4 位元的格雷碼：

```
0000 0001 0011 0010 0110 0111 0101 0100  
1100 1101 1111 1110 1010 1011 1001 1000
```

我們要求數列從最小位數開始跳位，也就是說底下數列雖然相鄰兩數間只有一位元不同，但是並不符合我們的要求。

```
000 010 011 001 101 111 110 100
```

再假設此數列從 0 開始，並且不會有重覆的數，因此，每一位元的數列只會有一組解。

## 輸入

輸入一個 3 至 10 位元的二進位數  $X$ ， $X$  不會是數列的頭或尾。

**實例1：** 010

**實例2：** 1111

**實例3：** 10101

## 輸出

輸出該長度之數列上  $X$  的前一個數、 $X$ 、 $X$  的下一個數。數與數之間空一格。

**實例1：** 011 010 110

**實例2：** 1101 1111 1110

**實例3：** 10100 10101 10111

## 前言

如果一個正數 (positive integer) 在某一個基底 (進位)，能夠被從左邊讀到右邊，或者右邊讀到左邊都是相同的情況下，這一個正數，它就能夠被稱為”迴文數” (palindrome)。

”迴文數”的型式如下所示：

給定一個數，反轉它的每一個數字，並將反轉結果加上原始的數。若結果不是一個迴文數，重覆以上步驟。例如，從一個數 87，基底為 10 (10 進位)開始。

步驟如下：

$$87 + 78 = 165$$

$$165 + 561 = 726$$

$$726 + 627 = 1353$$

$$1353 + 3531 = 4884, \text{ 一個迴文數}$$

尚未證明所有的數經由重覆以上步驟後最終都會成為一個迴文數。但是，所有以 10 為基底並小於 10,000 的數都已經被証實可以藉由有限的次數找到迴文數，除了 196 之外。雖然沒有證明存在來證明 196 沒有迴文數，但是 196 已經由上述方式產生到 200 萬個數字的數仍然無法找到迴文數。

## 輸入

每一組數都會由正整數 (positive integer) 數字和基底 (進位) 組成。其中，正整數不會超過 10 個數字。基底的範圍在 10 - 16。

(若基底大於 10，10 請用 A 表示，11 請用 B 表示，以此類推...15 請用 F 表示。)

**實例1:** A23 16

**實例2:** A345 12

**實例3:** 196 10

## 輸出

如果此迴文數能在十次之內(包括十次)被產生的話，就印出迴文數。若超過 10 次 (> 10) 仍然無法產生迴文數，請印出文字 “NONE”。

**實例1:** D4D

**實例2:** 9B4B9

**實例3:** NONE

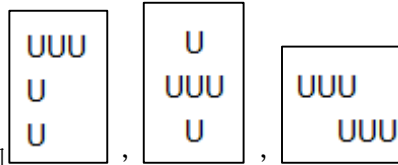
## 前言

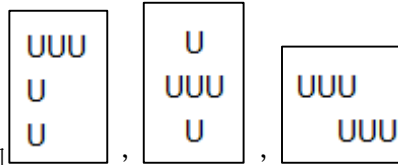
從傳統的「三個同樣圖案就消除」的基本玩法演變而來的遊戲眾多，複雜的盤面如何排出理想的配置總是讓人絞盡腦汁。本題希望各位寫一個程式用來計算已經排好的盤面可以消除多少組。

我們使用 R, U, G, Y, B, P 代表盤面上六種不同顏色的珠子，盤面的大小是常見的 6\*5，總計三十顆珠子，最大可以消除十組。

消除珠子的基本規則如下：

1. 不論橫向縱向，只要三個以上同樣顏色的珠子連在一起就可以消除一組。例如：UUU 是一組、UUUUUU 也只算作一組。



2. 多組交叉或相黏的時候，例如  此三種情形都視為一組消除。

在消除完盤面上所有可消除的珠子後會產生空格，此時上方的珠子會向下落到下層的珠子或是盤面底部。落下後可能產生新的可消除組數，比照先前的邏輯再進行處理，直到盤面上沒有可消除的珠子為止。只需要考慮盤面上現有的珠子，不會有新珠子進入盤面的情況發生。

## 輸入

每筆輸入是五列，每列六個字母代表盤面，例如：

```
GPBRRR
GPBYYY
GPBUUU
UUUYYY
RRRGGG
```

或是

```
RBRBRB
BRBRBR
RBUURB
BRGGUU
GGYYYY
```

## 輸出

對每筆輸入，請輸出各色珠子可消除的組數，依照 RUGYBP 的順序一色一列，以空格分隔。範例輸入的輸出如下：

```
R 2  
U 2  
G 2  
Y 2  
B 1  
P 1
```

```
R 0  
U 1  
G 1  
Y 1  
B 0  
P 0
```

## 前言

歐巴桑計畫到歐洲旅遊，他想要搭乘火車在各個城市間遊蕩。在旅遊服務處他取得了一張來往各城市間的火車票價圖。歐巴桑為了貪小便宜，想要知道從目前所在城市抵達地圖上其他城市，最少各需要多少錢。

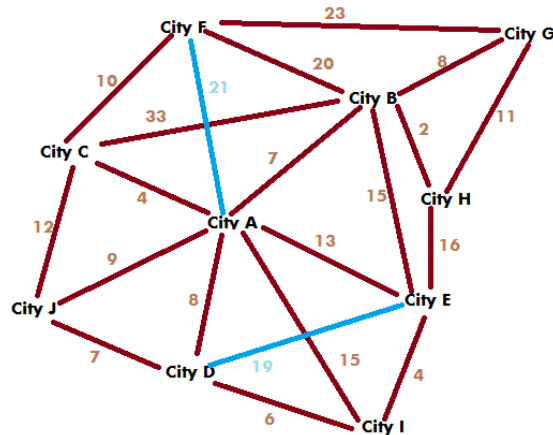


圖 1、各城市間的票價

## 輸入

輸入一個數字代表城市間的鐵路總數。接下來每行分別輸入兩個城市的代碼，以及串連兩城市鐵路的票價。輸入完所有代碼與票價後，最後一行為出發城市與目的城市。城市數最大為 20，不會有兩條不同路徑卻有相同票價的情況。

### 實例1:

```
5
A B 3
B C 7
C A 11
B D 6
B E 25
A E
```

### 實例2:

```
7
A B 6
D A 7
D C 11
E D 8
C A 10
A E 6
B D 5
C E
```



### 實例3:

```
9
A B 10
A C 7
A D 10
B C 3
B F 25
C D 5
C E 9
D E 8
D F 2
A F
```

### 輸出

出發城市與目的城市之間，搭乘鐵路所需的最少票價，以及從出發城市到目的城市間會經過的城市。

**實例1:** 28 A B E

**實例2:** 16 C A E

**實例3:** 12 A D F