競賽網站：http://algorithm.csie.ncku.edu.tw/ITSA/

第29次

教育部資訊軟體人才培育計畫
ITSA線上程式設計競賽

競賽題目

日期	時間	活動內容
103/02/26	17：50~18：00	報到、機器測試
星期三	18：00~21：00	比賽

主辦單位：成功大學 ITSA 教育部資訊軟體人才培育 程式設計e-tutor推廣

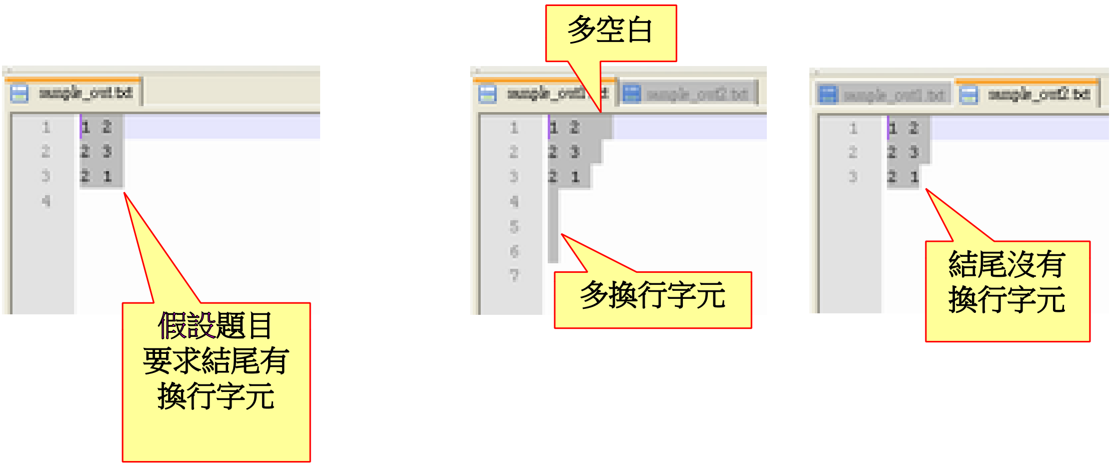
注意事項

1 本比賽系統採用PC²，所使用的 I/O 是標準輸出輸入裝置，所以可以使用C語言的scanf ( ) 、printf ( ) ，或是C++語言上的 cin、cout來讀入及輸出資料，比較要注意的是：本系統並不是用人工方式來keyin資料，所以不必在意使用者界面的問題，也就是說不用印出像是 "Please enter a number" 或 "The answer is" ．．．之類的文字；此外，有些題目是以讀到 EOF 為 input 結束，有些是讀入0結束等等的，必需善用I/O函式。上傳檔案的檔名請勿使用中文以免發生不必要的錯誤。

2 比賽用的編譯器版本：gcc 3.4.4 、 g++ 3.4.4 、 jdk 1.6.0_23 、 Microsoft (R) Visual C# 2010 Compiler version 4.0.30319.1 、 Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 16.00.30319.01。若出現Compilation Error，可能是某些函式不支援。

3 PC²系統判定錯誤可能原因：

正確答案　　　　錯誤答案



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

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

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

特別注意題目範例是否有換行字元。

4 PC²系統判定結果說明：

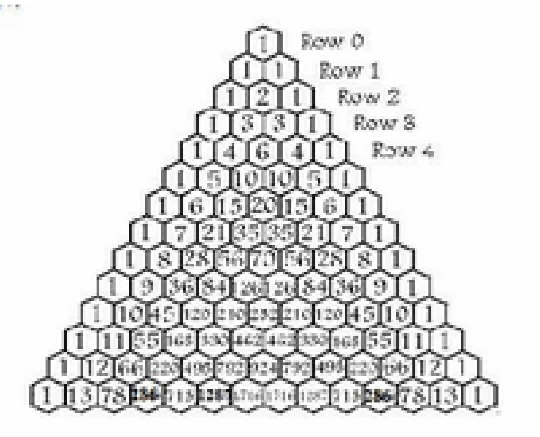
結果	**說明**
Yes	解題正確
No - Compilation Error	錯誤：編譯錯誤
No - Run-time Error	錯誤：程序運行錯誤
No - Time-limit Exceeded	錯誤：運行超時（每道題都有運行時間限制）
No - Wrong Answer	錯誤：運行結果與標準答案不一致
No - Excessive Output	錯誤：程序運行佔用內存空間超出要求
No - Output Format Error	錯誤：輸出格式錯誤
No - Other - Contact Staff	未知錯誤

Problem 1. 帕斯卡三角形

(Time Limit: 5 seconds)

問題描述：

帕斯卡三角形（Pascal's Triangle）的頂端是1，視為row 0。第1列 (row 1) 為兩個1，這兩個1是由他們上頭左右兩數之和（不在三角形內的數視為0）。依此類推產生第2列 (row 2): 0+1= 1 ; 1+1= 2 ; 1+0= 1 。第 3 列 (row 3): 0+1= 1 ; 1+2= 3 ; 2+1= 3 ; 1+0= 1。循此法可以產生以下諸列，如下圖所示。



每列由左而右各數，分別命名為第 0 元素，第 1 元素， … ，如此第n列第r元素是$\frac{n!}{r!(n-r)!}$。

請寫一個程式將帕斯卡三角形中之指定列的指定元素印出來。

輸入說明:

輸入檔中第一行為一個正整數N（$0 < N < 100$），代表共有N組測試資料。之後接下來有N行，每行有兩個整數，第一個整數n代表 帕斯卡三角形中之第n列$(0 \le n \le 20)$，第二個整數m代表該列由左而右第m個元素 。

輸出說明:

每筆測試資料結果輸出於一行，最後必須有換行字元。

範例:

Sample Input:	Sample Output:

3
0 0
3 1
10 4
	1
	3
	210

Problem 2. 增重問題

(Time Limit: 5 seconds)

問題描述：

拳擊手在比賽前經常耗費體力進行減重，在拳賽舉辦前日秤重合格後，選手們將有一天的時間可補充營養盡量恢復體力，然而飲食過度將造成反效果。身為一個拳擊教練必須調整好選手狀況以贏得勝利。現假設選手比賽前一日最多能吃重量為 W 的食物。拳擊教練提供了三種食物，一份和牛燒肉全餐重 w_1，全部吃完能獲得體力恢復 p_1；一份黑豬肉燒肉全餐重 w_2，全部吃完能獲得體力恢復 p_2；一份羔羊全餐重 w_3，全部吃完能獲得體力恢復 p_3。身為一個選手想要贏得勝利，當然希望能在有限的胃口容量下恢復最大體力。請問，在輸入 W、w_1、w_2、w_3、p_1、p_2 與 p_3 的情形下，如何食用這些餐點以恢復最大體力？

輸入說明:

輸入W、w_1、w_2、w_3、p_1、p_2 與 p_3值（可以為浮點數），所有值的範圍介在0到10之間。

輸出說明:

顯示和牛、黑豬肉與羔羊的食用量與最後獲得體力 (至小數點以下三位四捨五入)，若無解則顯示 0，最後必須有換行字元。

範例:

Sample Input:	**Sample Output:**
2.5 3.2 3.4 2.0 3 2 1	2.500 0.000 0.000 2.344

Problem 3. What Day is Today?

(Time Limit: 5 seconds)

Problem Description

Write a program to determine what day today is. We know that the first of January in 2000 is a Saturday. Now give the year (at least 2000), the month, and the day, your program must determine what day it is.

Note that we need consider leap year in order to do this right. A year y is a leap year if it is dividable by 400, or it is dividable by 4, but not 100.

The days of a week are: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.

Input File Format

The first line of the input is a number of dates ($0 < n < 100$). Each of the following n line has year, month, and day.

Output Format

For each date you must output what day it is. If the date is invalid, including the case where the year is less than 2000, you should output "invalid date".

Example

Sample Input:	**Sample Output:**

8
2000 1 1
2001 2 29
2006 3 13
2000 1 -4
2000 14 4
1999 1 1
2004 2 29
2006 1 9
	Saturday
	invalid date
	Monday
	invalid date
	invalid date
	invalid date
	Sunday
	Monday

Problem 4. 接龍可出牌數

(Time Limit: 5 seconds)

問題描述：

接龍是一個多人的撲克牌遊戲，其規則為，玩家擁有♥7（也有人以♠7或♣7）者必須先出牌，之後玩家輪流出牌；出牌方式必須接續同花色且依照數字順序（往上由 7 至K，往下由 6 至A），或者發出其他花色的7，直到所有人用完手牌為止。舉例而言，有人出♥7，下一位玩家必須發出♥6　或♥8，也可發出♠7、♣7或♦7；當有人沒辦法出牌時，必須選擇從手上覆蓋一張牌來略過這次出牌機會，此牌到遊戲結束前不可明示。遊戲結束時計算覆蓋牌點數的總和，最多的人最輸，但是從頭到尾都覆蓋牌的人算最贏。請寫一個程式，根據手上的牌，以及牌桌上已經出過的牌，判斷目前自己手上有哪些牌可以出。

輸入說明:

在本問題中，input是牌桌上已經出過的牌的數量、已經出過的每一張牌、自己手牌的數量、自己手上的每一張牌。每一張牌均用數字表示，表示方式如下所示：

梅花♣ : 1~13 = 0~12
方塊♦ :1~13 = 13~25
紅心♥ : 1~13 = 26~38
黑桃♠ : 1~13 = 39~51

例如，目前出過的牌為♣7 ♦7 ♥7 ♠7手牌為♣1 ♦2 ♦6 ♦12 ♥11 ♠4 ♠8，則input為

```
4
6 19 32 45
7
0 14 18 24 36 42 46
```

輸出說明:

output是所有可出的牌的數量 ，最後必須有換行字元。

範例:

Sample Input:	Sample Output:

4
6 19 32 45
7
0 14 18 24 36 42 46
	2

Problem 5. Typesetting

(Time Limit: 5 seconds)

Problem Description

Write a program to line up text. You are given a number *C* that indicates the number of characters that can appear in single line, and a set of words that must be typeset so that the text occupies exactly *C* columns. If the last word cannot be fit into a line, it must be broken up into pieces so that the most of it can appear in this line.

Note that sometimes we cannot put a whole word within the column limit, we need to break it up into pieces and only print as much as we could. Including the hyphen '-' at the end. It is a very difficult job to hyphenate a word so we will be very sloppy about this. We will use the following rules.

First find all vowel combinations.

Consecutive **'a', 'e', 'i', 'o', 'u'.**

Ending **'y'** is considered a vowel.

Ending **'e'** is ***not*** considered a vowel.

We simply assume that we can hyphen a word before the consonant that is before every vowel combination except the first one. This is usually the first letter other than vowel, with the following exception.

ch , sh , ph.

Now we have 'di-vide', 'cha-rac-ter ', ' fol-lo-wing', ' oc-cu-py ', 'ex-change', 'for-tu-na-te-ly '. ' const-ruc-tion ', 'hyphen', 'ma-chine'.

This rule is far from perfect but I just do not want to spend too much time in it. If you think that you have a simple and better algorithm for hyphenating words please do let me know.

Input File Format

The input consists of only letters, comma, period, and white space. Every string you get from **scanf** is either a English word, or a English word with comma or period at the end. To simplify things, you can assume that the period or the comma is part of the word, so the hyphenation of " example, " is " e-xamp-le, ", and " candy, " is "can-dy, ".

The number *C* will be at least three times of the length of the longest word, so any word or part of it will not occupy a whole line. ($0 \leq C \leq 100$)

You can use the return value of **scanf** to test if there is any more words to process. This program reads the input as a series of strings, and print them out one at a time. The total number of characters is no more than 1500.

Output Format

Now after finding all the places you can break up a word, you need to put as many characters into a line as possible, without breaking the column limit. Also in order to use up all *C* columns, you need to add extra space before those words near the end of the line. For example, in the following example, we need to put 3 extra space chars in the sixth line in the output, so we simply make the last three spacing 2 char in width (in " before those words").

Example

Sample Input:	**Sample Output:**
38 Now after finding all the places you can break up a word, you need to put as many characters into a line as poshible, without breaking the column limit. Also inorder to use up all C columns, you need to add extra space before those words near the end of the line. For example, in the following example, we need to put three extra space chars in the second line, so we simply make the last three spacing two char in width.	Now after finding all the places you can break up a word, you need to put as many characters into a line as po- shible, without breaking the column limit. Also in order to use up all C columns, you need to add extra space before those words near the end of the line. For example, in the following e- xample, we need to put three extra space chars in the second line, so we simply make the last three spacing two char in width.