2017 網際網路程式設計全國大賽
高中組模擬測試賽

- 本次比賽共 8 題，含本封面共 22 頁。
- 全部題目的輸入都來自**標準輸入**。輸入中可能包含多組輸入，以題目敘述為主。
- 全部題目的輸出皆輸出到螢幕 **(標準輸出)**。
  輸出和裁判的答案必須完全一致，英文字母大小寫不同或有多餘字元皆視為答題錯誤。
- 所有題目的時間限制請參考 Kattis 網頁上各題之標示。
- 比賽中上傳之程式碼，使用 C 語言請用 `.c` 為副檔名；使用 C++ 語言則用 `.cpp` 為副檔名。
- 使用 `cin` 輸入速度遠慢於 `scanf` 輸入，若使用需自行承擔 Time Limit Exceeded 的風險。
- 部分題目有浮點數輸出，會採容許部分誤差的方式進行評測。一般來說「相對或絕對誤差小於 ϵ 皆視為正確」，ϵ 值以題目敘述為主。

  舉例來說，假設 $\epsilon = 10^{-6}$ 且 a 是正確答案，b 是你的答案，如果符合 $\frac{|a-b|}{\max(|a|,|b|,1)} \leq 10^{-6}$，就會被評測程式視為正確。

2017 網際網路程式設計全國大賽
輸入輸出範例

C 程式範例：

```c
#include <stdio.h>
int main()
{
    int cases;
    scanf("%d", &cases);
    for (int i = 0; i < cases; ++i)
    {
        long long a, b;
        scanf("%lld %lld", &a, &b);
        printf("%lld\n", a + b);
    }
    return 0;
}
```

C++ 程式範例：

```cpp
#include <iostream>
int main()
{
    int cases;
    std::cin >> cases;
    for (int i = 0; i < cases; ++i)
    {
        long long a, b;
        std::cin >> a >> b;
        std::cout << a + b << std::endl;
    }
    return 0;
}
```

A. 北極熊大遷徙

Problem ID: polarbear

因為全球暖化的關係，北極各處的浮冰正在慢慢融化之中。部份北極熊所在的浮冰已經融化到不堪居住的程度，於是這些北極熊興起遷徙的念頭。

已經融化到不堪居住的浮冰 A 上有 a 隻北極熊，牠們現在打算遷徙到有 b 隻北極熊居住的浮冰 B。你要回答的是：經過北極熊大遷徙以後，浮冰 B 上總共會有多少隻北極熊。

Input

輸入只有一行，有兩個整數 a 和 b，代表有 a 隻北極熊即將從浮冰 A 遷徙到原本有 b 隻北極熊的浮冰 B。

- $1 \leq a, b < 2^{31}$

Output

輸出一行，表示浮冰 B 上最後會有多少隻北極熊。

Sample Input 1	Sample Output 1
24 47	71

Sample Input 2	Sample Output 2
33 20	53

This page is intentionally left blank.

B. 南極企鵝大遷徙

Problem ID: penguin

因為全球暖化的關係，南極各處的浮冰正在慢慢融化之中。部份企鵝居住地的浮冰已經大量融化，導致他們重要的食物來源「磷蝦」數量銳減，已到不堪居住的程度。已經融化到不堪居住的浮冰 A 上有 a 公斤的企鵝，牠們現在打算遷徙到有 b 公斤的企鵝居住的浮冰 B。

你要回答的是：經過企鵝大遷徙以後，浮冰 B 上總共會有多少公斤的企鵝。

Input

輸入只有一行，有兩個浮點數 a 和 b，代表有 a 公斤的企鵝即將從浮冰 A 遷徙到原本有 b 公斤重的企鵝的浮冰 B。

- $0 \leq a, b \leq 50$
- a, b 的小數點後最多有五位。

Output

輸出一行，表示浮冰 B 上最後會有多少公斤的企鵝。

如果你的答案的絕對或相對誤差不超過 10^{-6} 都會被當作正確。

Sample Input 1	Sample Output 1
24.23 47.33	71.56

Sample Input 2	Sample Output 2
24.23000 47.33000	71.5600000000000

Sample Input 3	Sample Output 3
24.230 47.330	71.560001

This page is intentionally left blank.

C. 北極熊大遷徙研究

Problem ID: polarbear2

因為全球暖化的關係，北極各處的浮冰正在慢慢融化之中。部份北極熊所在的浮冰已經融化到不堪居住的程度，於是這些北極熊興起遷徙的念頭。

已經融化到不堪居住的浮冰 A 上有 a 隻北極熊，牠們曾經遷徙到有 b 隻北極熊居住的浮冰 B。你是個學者，你正在研究北極熊的遷徙狀態。已知目前浮冰上已有 x 隻北極熊，你想知道在遷徙時有多少外來的北極熊 a 跟原生的北極熊 b。

你要回答的是：經過北極熊大遷徙之前，浮冰 A, B 上可能分別會有多少隻北極熊，需要一個可能的答案，但你也知道以前的北極熊族群不會太大，不會超過 1000 隻。

Input

輸入只有一行，只有一個整數 x，表示你要研究的浮冰上有 x 隻北極熊。

- $0 \leq x \leq 2000$

Output

輸出一行，有兩個整數 a, b 並以一個空白隔開，分別表示浮冰 A, B 原有可能會有多少隻北極熊。

- $x = a + b$
- $0 \leq a, b \leq 1000$

Sample Input 1	Sample Output 1
4	3 1

Sample Input 2	Sample Output 2
5	4 1

Sample Input 3	Sample Output 3
14	5 9

D. 猜數字

Problem ID: guess

我在心中想了一個介於 1 到 1000 的整數，你有辦法猜到這個數字是多少嗎？每當你猜了一個數字，我可以告訴你猜的過低、過高或正確。但你最多只能猜 10 次，所以你要好好選擇你猜的數字。

互動說明

當你的程式打算要猜數字時，輸出一行且包含一個整數，這個整數必須介於 1 到 1000 之間。當你猜完數字後，記得要清空 (flush) 標準輸出 (standard out)。

當我們收到你的猜測後，會把你猜的結果回覆到你的標準輸入 (standard in)。回覆會是下列三種：

- “`lower`” 如果我想的數字比你猜的數字小
- “`higher`” 如果我想的數字比你猜的數字大
- “`correct`” 如果你猜到了

當你猜到了正確數字後，你的程式必須立刻結束 (exit)。如果你 10 次都猜錯了，你的程式將會被強制中止。

清空 (flush) 標準輸出的範例程式碼請參考題本。

This page is intentionally left blank.

E. Special Judge

Problem ID: judge

你有比過網際網路程式設計大賽，俗稱 NPSC 的比賽嗎？

啊不對，你已經在比了。

有時候 NPSC 會有一些題目的答案不只一組，所以常常會需要寫個程式來幫忙判斷。例如，讓浮點數的評分可以容許小的計算誤差，而不是純粹的字串比對，這樣我們可以讓 0.9999999999999 當作 1.0。或是在有些比賽中，對於一些分隔字元的使用採取較於不嚴格的標準，例如說題目規定要用空白字元隔開，但參賽者寫成用換行字元隔開，那也算對。可惜，NPSC 採嚴格標準。為簡化題目，我們定義分隔字元只有換行跟空白兩種，詳細字元請參考 Input 的說明。

為了方便我們明年有這種評分系統，現在要請你寫一個。

一般來說，我們會先用分隔字元把所有參賽者的輸出切開，這樣會變成一些文字片段。舉例來說 “a bb c” 會被切成 “a”, “bb”, “c”，三組字串。接著把裁判的答案也用一樣的方式切開。如此一來，我們會得到兩個字串陣列，再把這兩個字串陣列一一對應的字串拿出來比對即可。

接下來，我們要來實作比對的方式，如果發現浮點數要特別處理比對。首先我們先定義一些名詞：

- 數字：字元 ‘0’ 到 ‘9’ 之一。
- 數字字串：只有數字的字串，不能是空字串。
- 浮點數：開頭可以帶有一個正負號，再由數字字串組成。並可以帶有一個小數點 ‘.‘，小數點可以位於這串數字的任何位置（包含最前面或最後面），並把這串數字切成兩段（第一段或第二段可能爲空字串）。
  緊接著也可以接上科學記號的部分。科學記號的部分會是由一個字元 ‘e’ 或 ‘E’ 開始，接著可以接上一個正負號，再接上一個數字字串。
  舉例來說 “.2”, “1.”, “+10.e-10” 都是浮點數。
- 整數：開頭可以帶有一個正負號，再接上數字字串組成。

為簡化題目，我們先不考慮 C99/C11 有定義的 16 進位整數跟 16 進位浮點數，以及 INF, NaN 等特殊數值。

當比對兩個字串時，如果裁判方的字串是浮點數但非整數，且參賽者方的答案是整數或浮點數時，則採用浮點數判斷。在其他情況都採純粹的字串比對。至於實際浮點數誤差的部分，由於有太多邊界情況，而且你應該會抱怨你已經寫太多程式碼了，所以我們就簡化掉吧。

你的程式在解析後，需要對每一對字串依序輸出你得到的是哪兩個字串，以及他們適用哪一種比對法（字串比對或是浮點數比對）。如果切出來的字串數量不相同，你則需要在輸出到該對的時候，在缺少的那一邊做標示。詳細的輸出請參考 Output 的說明。

Input

測試資料的第一行會有兩個整數 N, M，分別表示參賽者的輸出有幾行，以及裁判的輸出有幾行。

接下來 N 行是參賽者的輸出，緊接著 M 行是裁判的輸出。

- 保證參賽者跟裁判的輸出**分別**不超過 2000 個字元 (包含換行字元)
- 保證所有字元只會有以下幾種：換行字元 '\n' (ASCII 值 10)、空白字元 ' ' (ASCII 值 32)、正號 '+' (ASCII 值 43)、負號 '-' (ASCII 值 45)、小數點 '.' (ASCII 值 46)、數字 '0' 到 '9' 以及英文大小寫字母
- 為簡化題目，保證參賽者跟裁判的輸出最後一個字元都會是換行字元

Output

假設參賽者的輸出切出 n 個字串 $a_1, a_2, \ldots, a_n$，而裁判的輸出切出 m 個字串 $b_1, b_2, \ldots, b_m$，則總共輸出 $\max(n, m)$ 行。

第 i 行輸出 a_i 跟 b_i 以及比對的方式，之間都用一個空白隔開。如果是用字串比對則輸出 "`str`" (不含引號)，如果是用浮點數比對則輸出 "`float`" (不含引號)。

如果 a_i 或 b_i 有缺少，則缺少的那方輸出 "`<missing>`" (不含引號)。另外，如果有任何一方缺少，則一定使用字串比對法。

Sample Input 1

```
3 15
IamStr
1 2  3  4 5 6   7  8  9  10 11
12           13          14 15
0.0
1.1
+2.2
-3.
+.4
+5.e+5
-.6E-6
StringBegin
8
+
.
-.
+.e+12
.e-13
+14.14e14.14
```

Sample Output 1

```
IamStr 0.0 str
1 1.1 float
2 +2.2 float
3 -3. float
4 +.4 float
5 +5.e+5 float
6 -.6E-6 float
7 StringBegin str
8 8 str
9 + str
10 . str
11 -. str
12 +.e+12 str
13 .e-13 str
14 +14.14e14.14 str
15 <missing> str
```

This page is intentionally left blank.

F. 愛玩的普遜

Problem ID: numbers

普遜最喜歡數字了，一堆數字照一定順序接起來的「數列」當然更讓他喜歡了。但是，傳說中最美麗的數列：「對稱數列」更是普遜追求的極致！

我們稱一個非空數列（項數大於零）$< a_i >_{i=1}^{n}$ 為「對稱數列」，若是對於所有的 k 從 1 到 n 都滿足 $a_k = a_{n+1-k}$ 。

現在普遜的好友，史冬咪，送給普遜一個「數列」來做為生日禮物。普遜為了表達對史冬咪的感謝，決定把這個「數列」經過數次操作變成一個「對稱數列」！所有操作都必須是「加入」操作，也就是在數列的某個位置加入一個**正整數**，加入後，其後方的所有數字都會往後面順移。舉例來說，$< 1, 2, 3, 4 >$ 可以經由一次「加入」操作變成 $< 1, 2, 5, 3, 4 >$、$< 1, 2, 3, 4, 514 >$ 或 $< 1126, 1, 2, 3, 4 >$，但是沒辦法變成 $< 1, 2, 4, 3 >$、$< 1, 2, 5, 4 >$ 或 $< 1, 2, 4 >$。

不難發現，能夠經由數次「加入」操作變成的「對稱數列」其實不只一個，於是普遜打算讓它變成最「好」的一個「對稱數列」。

我們比較兩個數列好壞的方法如下：

- 若兩個數列的項數不一樣多，那麼項數比較小的數列比較好。
- 若前項比較無法完成比較，則看兩個數列的第一項（即 a_1），第一項比較小的數列比較好。
- 若前項比較無法完成比較，則看兩個數列的第二項（如果存在的話），第二項比較小的數列比較好。
- 若前項比較無法完成比較，則看兩個數列的第三項（如果存在的話），第三項比較小的數列比較好。
- 以此類推比較下去，直到完成比較為止。(可以發現如果到結束都無法比較出好壞關係，表示兩個數列確實相等。)

現在給你史冬咪送給普遜的數列，請你求出普遜能藉由數次「加入」操作可以得到最「好」的「對稱數列」。

為了避免輸出過大，令計算出所求的數列為 $< b_i >_{i=1}^{m}$，請輸出 $(b_1+1) \bigoplus (b_2+2) \bigoplus \cdots (b_i+i) \bigoplus \cdots (b_m+m)$，其中 $\bigoplus$ 為位元互斥或（Bitwise XOR），即為 C/C++ 中的運算子 "^"

例如，範例測試資料第二筆，所求的數列為 $< 1, 2, 3, 2, 1 >$，上述需輸出的值即為 $(1+$

$1) \bigoplus (2+2) \bigoplus (3+3) \bigoplus (2+4) \bigoplus (1+5) = 0$；另外，範例測試資料第一筆中，所求的數列為 $< 4, 1, 5, 1, 4 >$，可以用同樣方法計算出輸出應為 2。

Input

測試資料共包含 2 行。

第一行包含一個正整數 n，表示史冬咪送給普遜的數列的項數。

第二行包含 n 個正整數，依序史冬咪送給普遜的數列的第一項到第 n 項。

- $1 \leq n \leq 2000$
- $1 \leq$ 史冬咪送給普遜的數列的每一項 ≤ 2000

Output

輸出一行，包含一個整數，即為題目中所要求的值。

Sample Input 1	Sample Output 1
3 5 1 4	2

Sample Input 2	Sample Output 2
5 1 2 3 2 1	0

G. 密碼鎖

Problem ID: lock

頗旺是臺灣大學的學生，每天都開心的騎着他心愛的腳踏車上學。

然而，在臺灣大學中，有許多關於腳踏車的不可思議現象。像是，「腳踏車的稠密性」：你永遠能在兩輛腳踏車之間停入你的腳踏車；「車籃聚寶盆」：當你腳踏車停了一段時間之後，腳踏車車籃上便會自動聚集若干寶特瓶；「薛丁格的腳踏車」：在你回去牽你的腳踏車之前，你永遠不會知道你的腳踏車依然停在腳踏車位上，還是被拖吊走了。當然，這些都比不上，最可怕的「消失的腳踏車」，沒錯，就是你的腳踏車就憑空消失了。

儘管頗旺對於這些鄉野謠傳的不可思議現象不感興趣，但倘若哪天真的遇上的「消失的腳踏車」，就算一把鼻涕、一把眼淚也是找不回來的。因此，頗旺決定爲他心愛的腳踏車配上一個密碼鎖，如此一來，腳踏車就不會輕易的消失了！

但是，記憶密碼對於頗旺來說實在是太困難了，因此，他決定買一個特別製作的密碼鎖，讓他就算忘記密碼，也有機會把密碼找回來！

準確來說，頗旺所買的密碼鎖，總共有 N 位，每位分別包含數字 $1 \sim R$。而要打開這個密碼鎖，必須將每個位子都轉到對應的數字，例如，第一位是 5、第二位是 1、第三位是 4、$\cdots$。當然每個位子的數字都是環狀的，也就是說，當我把數字 1 往前轉一格時會轉到數字 2、數字 2 往後轉一格會到數字 1、數字 R 往前轉一格會轉到數字 1，數字 1 往後轉一格會轉到數字 R，以此類推。

而這個密碼鎖的特別之處在於，當你轉到不是正確的密碼時，他不僅僅只是鎖解不開來，他還會告訴你最少需要轉動多少格才能轉到正確密碼。舉例來說，這個密碼鎖有 3 位，每位分別包含數字 $1 \sim 5$。而對應的密碼是第一位爲 5、第二位爲 1、第三位爲 4，而若當你轉到第一位爲 1、第二位爲 3、第三位爲 3 時，密碼鎖會告訴你最少需要轉動 4 格才能轉到正確密碼（第一位往後轉一格、第二位往後轉兩格、第三位往前轉一格）。

頗旺拿到這個密碼鎖之後非常開心，認爲有這麼方便的功能就不用辛苦的去記憶密碼了。然而，事情總不是都那麼順利，頗旺第一天就忘記密碼了！看著心愛的腳踏車被密碼鎖鎖著無法動彈，頗旺心急如焚的想要把他解開。這時，頗旺才發現，這功能一點都不實用呀！

頗旺試了半天都毫無斬獲。剛好，身爲頗旺的好友，你看見頗旺正看著他心愛的腳踏車焦慮著，你決定來幫助他解開這個密碼鎖，看著頗旺心急的樣子，你答應他會在 $\lfloor N \lg(\lg(\lg(R))) \rfloor$ 次內把密碼解開。事不宜遲，這就開始解密碼鎖吧！($\lg(x) = \log_2(x)$，表示 2 的多少次方爲 x，e.g. $\log_2(1) = 0, \log_2(4) = 2, \log_2(5) = 2.3219\cdots, \log_2(5.14) = 2.3617\cdots$)（$\lfloor x \rfloor$ 代表小於等於 x 最大的整數）

互動說明

一開始，你的程式會從標準輸入 (standard input) 讀入兩個正整數 N、R 代表頗旺的密碼鎖有 N 位，每位分別有數字 $1 \sim R$。

而你便可以開始猜測頗旺的密碼：

當你的程式決定要猜密碼時，輸出一行包含 N 個整數，每個整數必須介於 1 到 R 之間。第一個數字代表第一位密碼、第二個數字代表第二位密碼、$\cdots$。當你輸出完成後，記得要清空 (flush) 標準輸出 (standard output)。

當我們收到你的猜測後，會把密碼鎖的結果回覆到你的標準輸入 (standard input)。回覆會包含一個非負整數，代表你最少需要轉幾格能轉到正確的密碼。

特別的是，當你猜到正確的密碼時，密碼鎖會回覆 0，你的程式必須立刻結束 (exit)。如果你在 $\lfloor N \lg(\lg(\lg(R))) \rfloor$ 次內無法找到正確的密碼，你的程式將會被強制終止。

- $10 \leq N \leq 514$
- $10^6 \leq R \leq 10^9$

以下是 C 程式 flush 的範例：

```c
#include <stdio.h>
int main(){
  int n, r, i;
  scanf( "%d%d" , &n , &r );
  for( i = 1 ; i < n ; i ++ )
    printf( "1 " );
  printf( "1\n" );
  fflush( stdout );
  long long ret;
  scanf( "%lld" , &ret );
}
```

以下是 C++ 程式 flush 的範例：

```cpp
#include <iostream>
int main(){
  int n, r;
  std::cin >> n >> r;
  for( int i = 1 ; i < n ; i ++ )
    std::cout << "1 ";
  std::cout << "1\n";
  std::cout << std::flush;
  long long ret;
  std::cin >> ret;
}
```

This page is intentionally left blank.

H. 吃吃為吃吃，是吃也

Problem ID: cook

喜歡做點心餵食小蘿莉的蘿莉農為了精進他的廚藝，特地從世界各地收集了 n 種特級食材。他打算從中挑選一些食材製作成特級餅乾，之後裝進他親手製作的精美餅乾盒中。但要挑選哪些食材好呢？蘿莉農覺得他必須親自嚐過所有可能的非空食材組合，才有辦法從這 $2^n - 1$ 種組合中挑出最適合蘿莉食用的餅乾。

我們知道，有些食材是天生絕配，搭在一起會有額外的好吃度加成；而有些食材如果混在一起的話反而會有反效果，甚至是食物中毒。更精確地來說，一個食材搭配是一個集合 s 以及一個好吃度影響值 v，如果挑選的食材組合包含 s 的話，其好吃度就會加上 v。也就是說一個食材組合的好吃度，就是其所有包含的食材搭配的好吃度總和。舉例來說，如果食材搭配 $\{1, 2\}$ 的好吃度影響值是 3，而食材搭配 $\{2, 3\}$ 的好吃度影響值是 -1，那麼食材組合 $\{1, 2, 3\}$ 的好吃度就是 $3 + (-1) = 2$。

蘿莉農嘗試各種組合的同時，他的廚藝熟練度也會逐漸上升。因此若第 i 次的食材組合好吃度是 d，則製作出來的餅乾好吃度會是 $i \times d$。為了讓這個試嚐餅乾的過程盡量愉悅而不要從此對餅乾有心理陰影，他希望找一個最好的組合嘗試順序，讓做出來的餅乾總好吃度最大。以上面的例子來說，一種可能的最好嘗試順序為 $\{2, 3\}, \{1\}, \{2\}, \{3\}, \{1, 3\}, \{1, 2, 3\}, \{1, 2\}$，其總好吃度為 $1 \times -1 + 2 \times 0 + 3 \times 0 + 4 \times 0 + 5 \times 0 + 6 \times 2 + 7 \times 3 = 32$。

Input

測試資料第一行有兩個正整數 n, m，分別代表食材數跟食材搭配數。接下來 m 行，每行會有一個字串 s_i 跟一個整數 v_i。如果 s_i 的第 j 個字元是 1 的話代表此搭配中包含第 j 種食材；反之若為 0 的話則代表不包含。而 v_i 則為此搭配的好吃度影響值。

- $1 \leq n \leq 22$
- $1 \leq m \leq 10^5$
- $|s_i| = n$
- s_i 中至少有一個字元為 1
- 所有的 s_i 皆相異
- $-100 \leq v_i \leq +100$

Output

請輸出一個整數於一行，代表最好嘗試順序的總好吃度。

Sample Input 1

```
2 2
01 1
10 -1
```

Sample Output 1

```
2
```

Sample Input 2

```
3 2
110 3
011 -1
```

Sample Output 2

```
32
```