# **UVa 10071 - Back to High School Physics 回到高中物**理

A particle has initial velocity and acceleration. If its velocity after certain time is v then what will its displacement be in twice of that time?

一個粒子能有初速度與加速度。若其經過一段時間 t 的速度為 v，則其兩倍時間

(2t)之位移為何？

Input 輸入

The input will contain two integers in each line. Each line makes one set of input. These two integers denote the value of v (−100 ≤ v ≤ 100) and t (0 ≤ t ≤ 200) (t means at the time the particle gains that velocity)

每行輸入包含 2 個整數。每行構成一個輸入集合。這 2 個整數表示速度 v(−100

≤ v ≤ 100)與時間 t (0 ≤ t ≤ 200)（t 為粒子維持速度 v 的時間）。

Output 輸出

For each line of input print a single integer in one line denoting the displacement in

double of that time. 對每行輸入，一行印出一個單一整數表示兩倍時間的位移。

Sample Input 輸入範例

0 0

5 12

Sample Output 輸出範例

0

120

解法：每一行讀入 v 與 t，印出 2\*v\*t。

/\* UVa 10071 - Back to High School Physics

\* by C111193227林佩璇,

\* 2025/09/17

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 判斷是否有資料輸入

while (scanner.hasNext()) {

// 取得 v 值

int v = scanner.nextInt();

// 取得 t 值

int t = scanner.nextInt();

// 印出計算結果

System.out.println(2 \* v \* t);

}

}

}

# 10055 - Hashmat the Brave Warrior 勇士 黑徐梅特

Hashmat is a brave warrior who with his group of young soldiers moves from one place to another to fight against his opponents. Before Fighting he just calculates one thing, the difference between his soldier number and the opponent’s soldier number. From this difference he decides whether to fight or not. Hashmat’s soldier number is never greater than his opponent.

勇士黑徐梅特與他那群年輕士兵們不斷移防以對抗敵人。在戰鬥前，他會掐指算算他的士兵與敵人士兵的數量差距。他從這個差距來決定是否要進行戰鬥。黑徐梅特的士兵不曾比他的敵人更多。

Input 輸入

The input contains two numbers in every line. These two numbers in each line denotes the number soldiers in Hashmat’s army and his opponent’s army or vice versa.

The input numbers are not greater than 232. Input is terminated by ‘End of File’.

每行輸入有 2 個數。這 2 個數分別表示黑徐梅特的士兵數量與敵方的士兵數

量，

也可能是反過來的。輸入的數不大於 232。輸入資料在‘End of File’檔案結束符號時終止。

Output 輸出

For each line of input, print the difference of number of soldiers between Hashmat’s army and his opponent’s army. Each output should be in separate line.

對每行輸入，印出黑徐梅特的士兵與敵方的士兵的數量差距。每組輸出應該不同行。

Sample Input 輸入範例

10 12

10 14

100 200

Sample Output 輸出範例

2

4

100

解法：

1. 注意，要用 long long 儲存以免爆炸。
2. 輸出之差距不可為負數。

/\* UVa 10055 - Hashmat the Brave Warrior

\* by C111193227林佩璇,

\* 2025/09/17

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 偵測是否有資料輸入

while (scanner.hasNextLong()) {

// 取得 a

long a = scanner.nextLong();

// 取得 b

long b = scanner.nextLong();

// 印出 a-b 取絕對值

System.out.println(Math.abs(a - b));

}

}

}

# 10783-Odd Sum 奇數和

Given a range [a, b], you are to find the summation of all the odd integers in this range. For example, the summation of all the odd integers in the range [3, 9] is 3 + 5 + 7 + 9 = 24.

給定一個範圍 [a, b]，您需要找到該範圍內所有奇數整數的總和。 例如，

[3, 9] 範圍內所有奇數的總和為 3 + 5 + 7 + 9 = 24。

Input 輸入

There can be at multiple test cases. The first line of input gives you the number of test cases, T (1 ≤ T ≤ 100). Then T test cases follow. Each test case consists of 2 integers a and b (0 ≤ a ≤ b ≤ 100) in two separate lines.

可以有多個測試用例。 輸入的第一行給出測試案例的數量 T (1 ≤ T ≤ 100)。 然後是 T 測試用例。 每個測試案例由兩個單獨的行中的 2 個整數 a 和 b (0 ≤ a ≤ b

≤ 100) 組成

Output 輸出

For each test case you are to print one line of output – the serial number of the test case followed by the summation of the odd integers in the range [a, b].

對於每個測試案例，您將列印一行輸出 - 測試案例的序號，後面跟著 [a, b] 範圍內奇數整數的總和。

Sample Input 輸入範例

2

1

5

3

5

Sample Output 輸出範例

Case 1: 9 Case 2: 8

解法：

先輸入一個整數 times 代表執行次數(用 for 迴圈)，在輸入兩個整數 a、b，符合條件時開始找範圍裡再將符合條件的數加總。

/\* UVa 10783 - Odd Sum

\* by C111193227林佩璇,

\* 2025/09/17

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 讀取資料筆數

int cases = scanner.nextInt();

// 使用 for 迴圈, 產生 0 ~ 小於資料筆數的索引值, 準備讀取測試資料

for (int i = 0; i < cases; i++) {

// 取得數字 a

int a = scanner.nextInt();

// 取得數字 b

int b = scanner.nextInt();

// 宣告變數, 儲存加總結果

int sum = 0;

// 使用 for 迴圈, 產生數字 a ~ b

for (int number = a; number <= b; number++) {

// 判斷數字是否為奇數, 將數字除以 2 取餘數, 若餘數等於 1 則為奇數

if (number % 2 == 1) {

// 將數字累加

sum = sum + number;

}

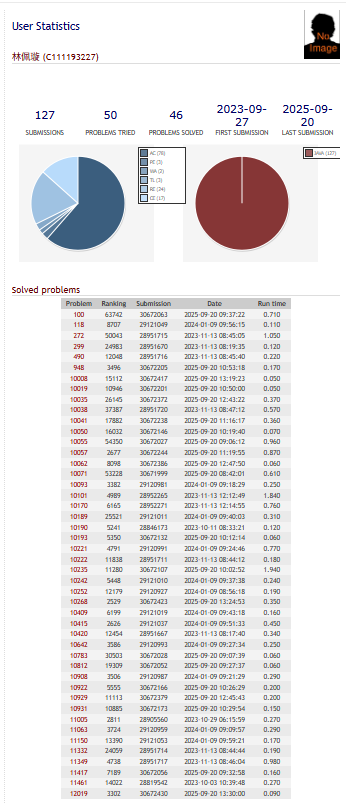
}

// 印出加總結果

System.out.println("Case " + (i + 1) + ": " + sum);

}

}



# 10812-Beat the Spread!擊敗傳播！

The Super Bowl is fast approaching. To pass the time while waiting for the halftime ads and wardrobe malfunctions, a local hacker group has organized a betting pool for the game. Members can place bets on either the sum of the two final scores or the absolute difference between the two scores. Given the winning numbers for each bet type, can you deduce the final scores?

超級盃週日即將到來。 為了打發等待中場廣告和衣櫃故障的時間，當地駭客組織了一場比賽的投注池。 會員將賭注押在兩個最終得分的總和上，或押在兩個得分之間的絕對差值上。 根據每種投注類型的中獎號碼，你能推導出最終的分數嗎？

Input 輸入

The first line of input contains n, the number of test cases. n lines follow, each representing a test case. Each test case gives s and d,non-negative integers representing the sum and (absolute) difference between the two final scores

輸入的第一行包含 n，即測試案例的數量。 接下來有 n 行，每行代表一個測試案例。 每個測試案例給出 s 和 d，非負整數，表示兩個最終分數之間的總和和（絕對）差

Output 輸出

For each test case, output a line giving the two final scores, largest first. If there are no such scores,

output a line containing “impossible”. Recall that football scores are always non-negative integers.

對於每個測試案例，輸出一行給出兩個最終分數，最大的在前。 如果沒有這樣的分數，輸出一行包含「不可能」的內容。 回想一下，足球比分始終是非負整數。

Sample Input 輸入範例

2

40 20

20 40

Sample Output 輸出範例

30 10

Impossible

解法：聯立方程式，假設兩隊為：a 隊、b 隊， a 隊分數 ≥ b 隊分數

/\* UVa 10812-Beat the Spread!

\* by C111193227林佩璇,

\* 2025/09/24

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 讀取資料筆數

int cases = scanner.nextInt();

// 使用 for 迴圈, 產生 0~小於資料筆數的索引值, 準備讀取測試資料

for (int i = 0; i < cases; i++) {

// 取得兩隊相加

int sum = scanner.nextInt();

// 取得兩隊相減

int diff = scanner.nextInt();

// 計算 a 隊與 b 隊的分數

// 根據題目給予的資訊

// sum = 隊伍 a + 隊伍 b, a = sum - b

// diff = 隊伍 a - 隊伍 b, b = diff - a

// --------------------- +

// => 2 a = sum + diff

// => a = (sum + diff) / 2

int a = (sum + diff) / 2;

int b = sum - a;

// 判斷是否為合理情況

// 若 a 隊或 b 隊分數小於 0, 或 sum + diff 無法被 2 整除(餘數為 1), 則為不合理情況

if (a < 0 || b < 0 || (sum + diff) % 2 == 1) {

// 印出題目指定字串

System.out.println("impossible");

} else {

// 印出兩隊各自的分數

System.out.println(a + " " + b);

}

}

}

}

# 11461-Square Numbers 平方數平方

A square number is an integer number whose square root is also an integer. For example 1, 4, 81 are some square numbers. Given two numbers a and b you will have to find out how many square numbers are there between a and b (inclusive).

平方數是一個整數，其平方根也是整數。 例如 1、4、81 是一些平方數。 給定兩個數字 a 和 b，你必須找出 a 和 b（含）之間有多少個平方數。

Input 輸入

The input file contains at most 201 lines of inputs. Each line contains two integers a and b (0 < a ≤ b ≤ 100000). Input is terminated by a line containing two zeroes. This line should not be processed. 輸入檔最多包含 201 行輸入。 每行包含兩個整數 a 和 b (0 < a ≤ b ≤ 100000)。輸入以包含兩個零的行終止。 不應處理該行。

Output 輸出

For each line of input produce one line of output. This line contains an integer which denotes how many square numbers are there between a and b (inclusive).

對於每一行輸入產生一行輸出。 這一行包含一個整數，表示 a 和 b（含）之間有多少個平方數。

Sample Input 輸入範例

1 4

1 10

0 0

Sample Output 輸出範例

2

3

解法：取最小整數開根號的值 i，從 i 開始算平方，直到 i 平方大於 b 時停止，在輸出結果。

/\* UVa 11461-Square Numbers

\* by C111193227林佩璇,

\* 2025/09/24

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 題目說 a b 兩數會介於 1 ~ 100000 之間

// 後續會使用數字作為陣列索引, 如果要訪問索引 100000 的位置, 陣列長度需為 100001

int size = 100001;

// 建立記錄是否為平方數的陣列, 方便後續查詢

boolean table[] = new boolean[size];

// 使用 for 迴圈, 產生從 1 ~ 100000 的數字

for (int i = 1; i \* i < size; i++) {

// 將數字相乘的索引位置設為 true, 表示此索引位置為平方數

table[i \* i] = true;

}

// 判斷是否有資料輸入

while (scanner.hasNext()) {

int a = scanner.nextInt(); // 取得 a

int b = scanner.nextInt(); // 取得 b

// 判斷 a 或 b 是否為 0

if (a == 0 && b == 0) {

// 若為 0, 則結束計算

break;

}

// 記錄平方數的數量

int count = 0;

// 使用 for 迴圈, 產生從 a ~ b 之間的數字

for (int i = a; i <= b; i++) {

// 查表, 用 i 作為索引位置, 查詢陣列[i] 的值是否為 true

if (table[i]) {

// 若為 true, 則 i 為平方數, 平方數的數量+1

count = count + 1;

}

}

// 印出平方數的數量

System.out.println(count);

}

}

}

# 11417-GCD最大公因數

給定 N 的值，您必須找到 G 的值。G 的定義如下：



這裡GCD( i,j)表示整數i和整數j的最大公約數。對於那些難以理解求和符號的人，G 的含義在以下代碼中給出：

G=0； for(i= 1;i <=N;j++) { G+=GCD(i,j); } /\*這裡GCD()是求兩個輸入數字的最大公約數的函數\*/

給定 N 的值，您必須找到 G 的值。G 的定義如下：這裡 GCD(i,j)表示整數 i 和整數 j 的最大公約數。 對於那些難以理解求和符號的人，G 的含義在以下代碼中給出：

G=0；for(i=1;i<=N;j++) { G+=GCD(i,j); } /\*這裡 GCD()是求兩個輸入數字的最大公

因數的函數\*/

Input 輸入

The input file contains at most 100 lines of inputs. Each line contains an integer N (1 < N < 501). The meaning of N is given in the problem statement. Input is terminated by a line containing a single zero. This zero should not be processed.

輸入檔最多包含 100 行輸入。 每行包含一個整數 N (1 < N < 501)。N 的意思在問題陳述中給出。 輸入以包含單一零的行終止。 不應處理該零。

Output 輸出

For each line of input produce one line of output. This line contains the value of G for corresponding N. 對於每一行輸入產生一行輸出。該行包含對應 N 的 G 值。

Sample Input 輸入範例

10

100

500

0

Sample Output 輸出範例

67

13015

442011

解法：使用輾轉相除法求出 GCD，並全部加總。

/\* UVa 11417-GCD

\* by C111193227林佩璇,

\* 2025/09/24

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 偵測是否有資料輸入

while (scanner.hasNextInt()) {

// 取得數字 n

int n = scanner.nextInt();

// 判斷數字 n 是否等於 0

if (n == 0) {

// 若為 0 則結束計算

break;

}

// 按照題目給的程式碼打出來

// 宣告變數 g, 用來儲存加總結果

int g = 0;

// 產生 1 ~ 小於 n 的數字

for (int i = 1; i < n; i++) {

// 產生 i+1 ~ 等於 n 的數字

for (int j = i + 1; j <= n; j++) {

// 累加 i 與 j 的最大公因數

// 呼叫 gcd(i, j) 方法可取得兩數的最大公因數

g += gcd(i, j);

}

}

// 印出總和

System.out.println(g);

}

}

// 宣告 gcd 方法, 回傳是 int 型別, 可傳入兩個 int 參數

// 此方法的邏輯與 CPE 31 題相同

public static int gcd(int i, int j) {

// 使用 while 迴圈持續判斷, 當兩數皆大於 0 時要進行輾轉相除法

// 當其中一個數字為 0 時, while 迴圈就會停止

while (i > 0 && j > 0) {

// 判斷誰比較大, 大的數字要做為被除數, 小的數字要做為除數, 相除取餘數

if (i > j) {

i = i % j;

} else {

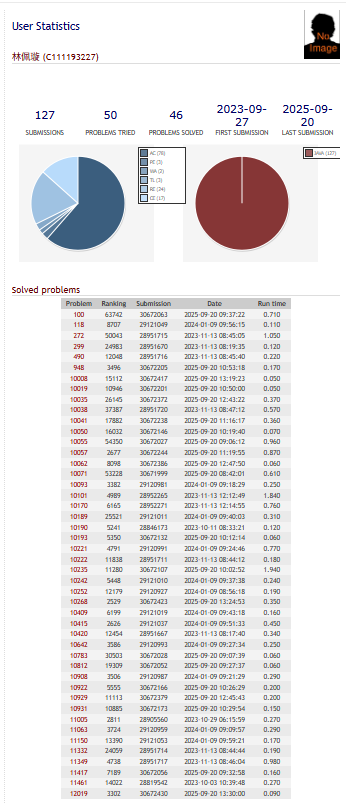
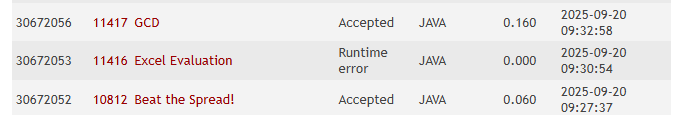
j = j % i;

}

}

// 回傳最大公因數, 兩數字較大者為最大公因數

return Math.max(i, j);



# 100 - The 3n+1 problem 3n+1 問題

計算機科學中的問題通常被歸類為屬於某一類問題（例如，NP 、不可解、遞歸）。在此問題中，您將分析演算法的屬性，該演算法的分類對於所有可能的輸入都是未知的。考慮以下演算法：

1. 輸入n
2. 列印 n
3. 如果 n = 1 則停止
4. 如果 n 是奇數則 n ← − 3n + 1
5. 否則 n ← − n/2
6. 轉到2

給定輸入 22，將列印以下數字序列22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1 據推測，對於任何整數輸入值，上述演算法都將終止（當列印1時）。儘管演算法很簡單，但尚不清楚這個猜想是否正確。然而，對於所有滿足 0 < n < 1, 000, 000 的整數n已經得到驗證（事實上，對於比這更多的數字）。給定輸入 n，可以確定數字的數量在打印1 之前打印（包括列印1）。對於給定的 n，這稱為 n的循環長度。在上面的例子中，22的循環長度是16。對於任意兩個數字i和j，你要確定i和j之間（包括兩者）的所有數字的最大循環長度。計算機科學中的問題通常被澄清為屬於其中類別問題（例如， NP ，不可解，遞回）。

在本題中，您將分析演算法的一個屬性，該演算法的不知道所有可能的輸入的分類。

考慮以下演算法：

1.輸入 n

1. 列印 n
2. 如果 n = 1 則停止
3. 如果 n 是奇數則 n ←− 3n + 1
4. 否則 n ←− n/2 6. 轉到 2 給定輸入 22，將列印以下數字序列 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1 據推測，對於任何整數輸入，上述演算法都會終止（當列印 1 時）價值。 儘管演算法很簡單，但尚不清楚這個猜想是否正確。它有然而，對於所有滿足 0 < n < 1, 000, 000 的整數 n（事實上，對於更多比這個數字。）給定輸入 n，可以確定先前列印的數字數量，包括 1 被印出來。 對於給定的 n，這稱為 n 的循環長度。 在上面的例子中，循環 22 的長度是 16。

對於任兩個數字 i 和 j，您需要確定所有數字的最大循環長度

介於 i 和 j 之間（包括 i 和 j）。

Input 輸入

The input will consist of a series of pairs of integers i and j, one pair of integers per line. All integers will be less than 10,000 and greater than 0.

You should process all pairs of integers and for each pair determine the maximum cycle length over all integers between and including i and j.

You can assume that no operation overflows a 32-bit integer.

輸入將由一系列整數對 i 和 j 組成，每行一對整數。 所有整數將小於 10,000 且大於 0。

您應該處理所有整數對，並為每對確定最大循環長度 i 和 j 之間（含）的所有整數。

您可以假設沒有任何操作會溢出 32 位元整數

Output 輸出

對於每對輸入整數 i 和 j，您應該輸出 i、j 以及 i 和 j 之間（含 i 和 j）整數的最大循環長度。這三個數字應至少用一個空格分隔，所有三個數字都在一行上，並且每一行輸入對應一行輸出。整數 I 和 j 必須按照它們在輸入中出現的順序出現在輸出中，並且後面應該跟最大循環長度（在同一行上）。

對於每對輸入整數 i 和 j，您應該輸出 i、j 以及最大循環長度 i 和 j 之間（包括 i 和 j）的整數。 這三個數字應至少用一個空格分隔所有三個數字都在一行上，每一行輸入都有一行輸出。整數 i 和 j 必須按照它們在輸入中出現的順序出現在輸出中，並且應該是接下來是最大周期長度（在同一行）。

Sample Input 輸入範例

1 10

100 200

201 210

900 1000

Sample Output 輸出範例

1 10 20

100 200 125

201 210 89

900 1000 174

解法：依題目給的判別方式去判別，並輸出最大值即可。

/\* UVa 100 - The 3n+1 problem

\* by C111193227林佩璇,

\* 2025/10/01

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 偵測是否有資料輸入

while (scanner.hasNextInt()) {

int i = scanner.nextInt(); // 取得數字 i

int j = scanner.nextInt(); // 取得數字 j

// 因為後續 i j 可能會交換, 所以先將 i 與 j 印出來

System.out.print(i + " " + j + " ");

// 判斷 i 是否大於 j

if (i > j) {

// 若 i 大於 j, 則將 i 與 j 進行交換

// 宣告一個變數, 用來儲存 i 的原始值

int tempI = i;

// 將 j 的數值指派給 i 變數

i = j;

// 將 i 的原始數值指派給 j

j = tempI;

}

// 用來儲存最長數列的個數

int max = 0;

// 使用 for 迴圈, 產生介於 i 與 j 之間的所有整數

for (int n = i; n <= j; n++) {

int number = n; // 取得數值

// 記錄數字帶入公式的數列包含幾個數字

// 因為題目會將原始數字也算進去, 所以初始個數從 1 開始

int count = 1;

// 套用公式, 計數到數值等於 1 為止

while (number > 1) {

// 判斷是否為偶數

if (number % 2 == 0) {

// 套用特定公式

number = number / 2;

} else {

// 套用特定公式

number = 3 \* number + 1;

}

// 數列個數+1

count = count + 1;

}

// 判斷目前的數列個數是否大於最長數列個數

if (count > max) {

// 若大於則更新最長數列個數

max = count;

}

}

// 印出最大數列個數

System.out.println(max);

}

}

}

# 10190 - Divide,But Not Quite Conquer 分裂，但還沒有完全征服

您在此問題中的目標是將某個整數 n 除以另一個整數 m，直到 n = 1，以獲得數字序列。我們將這個序列中的每個數字稱為 a[i]，假設它有 k 個數字（即您必須進行 k − 1 次連續除法才能達到 n = 1）。僅當滿足以下限制時才能擁有此序列：

* a[ 1] = n，a[i] = a[i − 1] ÷ m，對於所有 1 < i ≤ k
* 對所有 1 ≤ i < k，a[i] 可被 m 整除（即，a[i] mod m = 0）
* a[ 1] > a[2] > a[3] > 。 。 。 > 一[k]

例如，如果 n = 125 且 m = 5，則有 125、25、5 和 1（您進行了 3 個除法：125/5、25/5 和 5/5）。因此，k = 4、 a[ 1] = 125、a[2] = 25、a[3] = 5 且 a[4] = 1。如果 n = 30 且 m = 3，則有 30, 10, 3 1. 但是a[2] = 10，並且10 mod 3 = 1，所以不存在序列，因為它違反了限制2. 當序列不存在時，我們認為它不好玩，因此非常無聊

此問題的目標是將某個整數 n 除以另一個整數 m，直到 n = 1，得到一系列數字。 讓我們稱這個序列的每個數字為 a[i]，假設它有 k 個數字（即您必須連續進行 k − 1 次除法才能達到 n = 1)。 只有在滿足以下條件時，您才能擁有此序列滿足限制：

* a[1] = n，a[i] = a[i − 1] ÷ m，對於所有 1 < i ≤ k
* 對於所有 1 ≤ i < k，a[i] 可被 m 整除（即，a[i] mod m = 0） • a[1] > a[2] > a[3] > 。 。 。 > 一[k]

例如，如果 n = 125 且 m = 5，則有 125、25、5 和 1（您進行了 3 個除法：125/5、

25/5 和 5/5)。 因此，k = 4、a[1] = 125、a[2] = 25、a[3] = 5 且 a[4] = 1。如果 n = 30 且 m = 3，則有 30、10、3 和 1。但是 a[2] = 10，且 10 mod 3 = 1，因此不存在序列，因為它違反了限制 2。當序列不存在時，我們認為它不有趣，並且，因此，非常無聊

Input 輸入

The input will consist on an arbitrary number of lines. Each line will consist of two nonnegative integers n, m which are both less than 2000000000. You must read until you reach the end of file.

輸入將包含任意數量的行。 每行由兩個非負數組成整數 n、m 均小於 2000000000。您必須讀取直到到達檔案末端。

Output 輸出

For each pair n, m you must print the correspondent sequence a (as defined above) in a single line, with each adjacent numbers of the sequence separated by a single space. In the case the sequence doesn’t exist because it violates some restriction, just print the phrase ‘Boring!’ in a single line (without the quotes).

對於每一對 n, m，您必須在一行中列印對應的序列 a（如上所定義），序列中每個相鄰的數字都以一個空格分隔。 如果序列因違反某些限製而不存在，只需在一行中列印短語“Boring！”（不含引號）。

Sample Input 輸入範例

125 5

30 3

1. 2
2. 3

Sample Output 輸出範例

125 25 5 1

Boring! Boring!

81 27 9 3 1

解法：

1. 數字超過 int 範圍，故需要使用 long。
2. 從 1 開始乘上 m，並記錄每次的結果，直到等於或超過 n。
3. 如果 Vector 裡面是空的，或者最後一位數不等於 n，則為 Boring，反之就印出題目所需要的結果。

/\* UVa 10190 - Divide,But Not Quite Conquer

\* by C111193227林佩璇,

\* 2025/10/01

\*/

import java.util.Scanner;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 偵測是否有資料輸入

while (scanner.hasNextLine()) {

int n = scanner.nextInt(); // 取得數字 n

int m = scanner.nextInt(); // 取得數字 m

// 判斷 m 是否等於 1

if (m == 1) {

// 若等於 1, 直接印出特定字串, 因為 n 除以 1, 商數都會等於 n, 無法收斂至 1

System.out.println("Boring!");

} else {

// 宣告串列, 用來儲存 n 除以 m 取商數的結果

ArrayList<Integer> numbers = new ArrayList<Integer>();

// 當 n 大於 1 且 n 可以被 m 整除時進入 while 迴圈

// 例如 n = 125, m = 5, 125 > 1 且 125 可以被 5 整除

while (n > 1 && n % m == 0) {

// 將數字 n 存入串列中

// 例如: 將 125 存入串列中

numbers.add(n);

// 將 n 更新為 n 除以 m 取商數, 準備下一次進入 while 時再繼續除以 m

// 例如: 將 n = 125 更新為 5 (125/5)

n = n / m;

}

// 判斷 n 是否等於 1

if (n == 1) {

// 將運算過程印出

for (Integer number : numbers) {

System.out.print(number + " ");

}

// 因為進入 while 迴圈的條件須滿足 n 大於 1

// 當 n=1時就無法進入 while 迴圈, 在這邊要手動輸出的字串

System.out.println("1");

} else {

// 若 n 無法收斂到 1, 印出特定字串

System.out.println("Boring!");

}

}

}

}

}

# 10235 - 簡單的酋長簡單的酋長

如果大於 1 的整數唯一的正因數（因數）是 1 和它本身，則該整數稱為素數。多年來，許多數學家一直在研究素數。

素數的應用出現在密碼學和編碼理論等領域。

Have you tried reversing a prime? For most primes, you get a composite (43 becomes 34). An Emirp (Prime spelt backwards) is a Prime that gives you a different Prime when its digits are reversed.For example, 17 is Emirp because 17 as well as 71 are Prime.In this problem, you have to decide whether a number N is Non-prime or Prime or Emirp. Assume that 1 < N < 1000000.Interestingly, Emirps are not new to NTU students. We have been boarding 199 and 179 buses forquite a long time! 如果大於 1 的整數唯一的正因數（因數）是 1 和它本身，則該整數稱為質數。多年來，許多數學家都對素數進行了研究。 素數的應用數字出現在密碼學和編碼理論等領域。您是否嘗試過反轉素數？ 對於大多數質數，您會得到一個合數（43 變成 34）。 一個 Emirp（Prime 反向拼字）是素數，當其數字顛倒時，它會為您提供不同的質數。例如，17 是 Emirp，因為 17 和 71 都是 Prime。在此問題中，您必須確定數字 N 是非素數、質數還是 Emirp。 認為即 1 < N < 1000000。有趣的是，Emirps 對南洋理工大學的學生來說並不陌生。 我們已經搭乘 199 和 179 公

車了相當長的一段時間了！

Input 輸入

Input consists of several lines specifying values for N. 輸入由指定 N 值的幾行組成。

Output 輸出

For each N given in the input, output should contain one of the following:

1. ‘N is not prime.’, if N is not a Prime number.
2. ‘N is prime.’, if N is Prime and N is not Emirp.
3. ‘N is emirp.’, if N is Emirp.

對於輸入中給出的每個 N，輸出應包含以下內容之一：

1. 如果 N 不是素數，則「N 不是素數」。

2.“N 是質數。”，如果 N 是質數且 N 不是 Emirp。

3. 如果 N 是 Emirp，則「N 是 emirp」。

Sample Input 輸入範例

17

18

19

179

199

Sample Output 輸出範例

1. is emirp.
2. is not prime.
3. is prime.

179 is emirp.

199 is emirp.

解法：

1. 原本數字、反轉後數字皆為質數且原本數字不等於反轉後的數字=emirp。
2. 原本數字為質數或者原本數字、反轉後數字皆為質數但原本數字等於反轉後的數字 =prime。
3. 非質數。

/\* UVa 10235 - Simply Emirp235

\* by C111193227林佩璇,

\* 2025/10/01

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 判斷是否有輸入

while (scanner.hasNextInt()) {

int n = scanner.nextInt(); // 取得數字 n

// 將數字 n 轉字串

String nText = Integer.toString(n);

// 宣告反轉字串

String reverseNText = "";

// 由大到小(由右而左)讀取字串 n 的每一個字元

for (int i = nText.length() - 1; i >= 0; i--) {

// 將字元加到反轉字串中

reverseNText = reverseNText + nText.charAt(i);

}

// 將反轉字串轉成數字

int reverseN = Integer.parseInt(reverseNText);

// 宣告布林值, 記錄 n 是否為質數, 預設值為 true

boolean nIsPrime = true;

// 尋找數字 n, 是否有 2 ~ 小於 n 的因數

// 1跟n不用判斷, 因為每個數字的因數都包含 1 跟 n

for (int i = 2; i < n; i++) {

// 判斷數字 n 是否可被 i 整除

if (n % i == 0) {

// 若可被整除, 表示因數數量不只 2 個, 所以是質數

nIsPrime = false;

// 結束 for 迴圈

break;

}

}

// 宣告布林值, 記錄反轉的 n 是否為質數, 預設值為 true

boolean reserveNisPrime = true;

// 尋找反轉數字, 是否有 2 ~ 小於 n 的因數

for (int i = 2; i < reverseN; i++) {

// 判斷反轉數字是否可被 i 整除

if (reverseN % i == 0) {

// 若可被整除, 表示因數數量不只 2 個, 所以不是質數

reserveNisPrime = false;

// 結束 for 迴圈

break;

}

}

// 印出數字 n

System.out.print(n);

// 若原始數字是質數、反轉數字也是質數且原始數字不等於反轉數字, 即為反質數

// 例子: 原始數字 17, 反轉後的數字一樣是 71, 這樣有 2 個數字, 所以是反質數

// 例子: 原始數字 11, 反轉後的數字一樣是 11, 這樣只有 1 個數字, 所以不是反質數

if (nIsPrime && reserveNisPrime && n != reverseN) {

// 印出反質數

System.out.println(" is emirp.");

} else if (nIsPrime) {

// 印出質數

System.out.println(" is prime.");

} else {

// 印出非質數

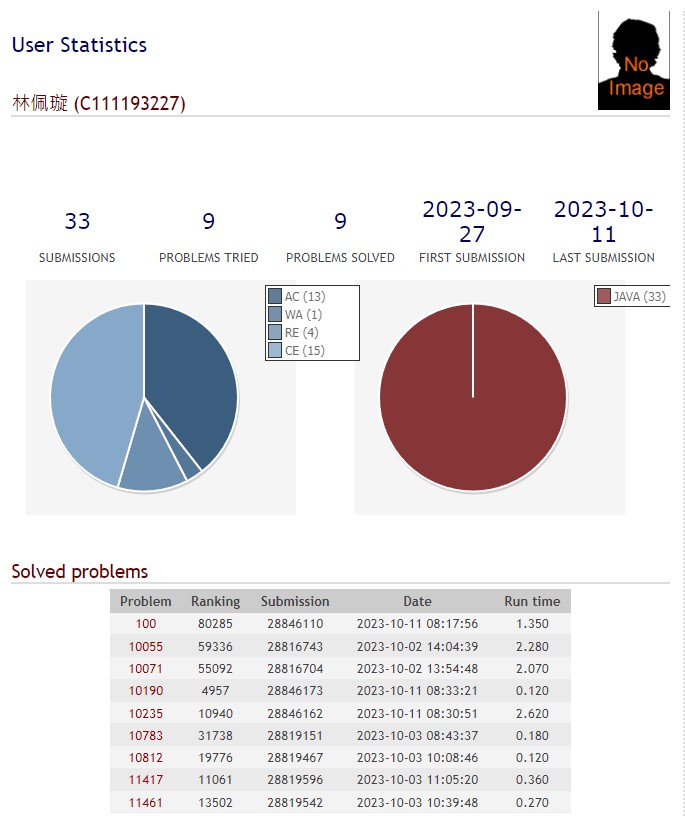
System.out.println(" is not prime.");

}

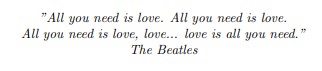
}

}

}



# 10193 - All You Need Is Love 你最需要的是愛



國際美麗機器公司發明了一種新的強大小玩意，叫做愛情機器！給定一個由二進制數字組成的字串，愛情機器會回答它是否僅由愛組成，也就是說，如果您只需要愛情來構建該字串。愛情機器的愛情定義是另一串二進制數字，由人類操作員給出。假設我們有一個代表「愛」的字串 L，並且我們為愛機器提供了一個字串 S。我們說，如果我們可以重複從S 中減去L，直到達到L，那麼您所需要的就是熱愛構建S。這裡定義的減法與以2 為底的算術減法相同。根據這個定義，很容易看出如果L > S （二進位），那麼 S 不是由愛組成的。如果S = L ，那麼S顯然是由愛組成的。讓我們來看一個例子。假設 S =“11011”且 L =“11”。如果我們不斷地從 S 中減去 L，我們得到： 11011, 11000, 10101, 10010, 1111, 1100, 1001, 110, 11。所以，給定這個 L，你所需要的就是建造 S。由於愛，不能有帶有前導零的字串。即「0010101」、「01110101」、「011111」等都是無效字串。只有一位數字的字串也是無效的（這是另一個限制）。此問題中您的任務是：給定兩個有效的二進位字串S1 和S2，查找是否可能有有效的字串L，使得S1 和S2 都只能由L 組成（即給定兩個有效的字串S1 和S2，找出如果存在至少一個有效字串L（例如S1和S2都僅由L組成）。例如，對於 S1 = 11011 和 S2 = 11000，我們可以讓 L = 11，這樣 S1 和 S2 都僅由 L 組成（如上例所示）。

國際美麗機器公司發明了一種新的強大小玩意，叫做愛情機器！ 給定一個由二進制數字組成的字串，愛情機器會回答它是否僅由愛組成，也就是說，如果您只需要愛情來構建該字串。 愛情機器的愛情定義是另一串二進制數字，由人類操作員給出。 假設我們有一個代表「愛」的字串 L，並且我們為愛機器提供了一個字串 S。我們說，如果我們可以重複從 S 中減去 L，直到達到 L，那麼您所需要的就是熱愛構建 S。這裡定義的減法與以 2 為底的算術減法相同。根據這個定義，很容易看出如果 L > S （二進位），那麼 S 不是由愛組成的。 如果 S = L，那麼 S 顯然是由愛組成的。 讓我們來看一個例子。 假設 S =“11011”且 L =“11”。 如果我們不斷地從 S 中減去 L，我們得到： 11011, 11000, 10101, 10010, 1111, 1100, 1001,

110, 11。所以，給定這個 L，你所需要的就是建造 S。由於愛，不能有帶有前導零的字串。 即「0010101」、「01110101」、「011111」等都是無效字串。 只有一位數字的字串也是無效的（這是另一個限制）。 此問題中您的任務是：給定兩個有效的二進位字串 S1 和 S2，查找是否可能有有效的字串 L，使得 S1 和 S2 都只能由 L 組成（即給定兩個有效的字串 S1 和 S2，找出如果存在至少一個有效字串 L（例如 S1 和 S2 都僅由 L 組成）。 例如，對於 S1 = 11011 和 S2 = 11000，我們可以讓 L = 11，這樣 S1 和 S2 都僅由 L 組成（如上例所示）。

輸入輸入

第一行輸入是一個正整數 N < 10000，代表測試案例的數量。然後，將有 2 N 行。每一對線由一個測試盒組成。該對中的每一行代表要作為愛情機器輸入的每個字串（S1 和 S2）。任何字串都不會超過 30 個字元。您可以假設輸入中的所有字串根據上述規則都是有效的。

第一行輸入是一個正整數 N < 10000，代表測試案例的數量。 然後，將有 2 \* N 行。 每一對線由一個測試盒組成。 該對中的每一行代表要作為愛情機器輸入的每個字串（S1 和 S2）。 任何字串都不會超過 30 個字元。 您可以假設輸入中的所有字串根據上述規則都是有效的。

Output 輸出

For each string pair, you must print one of the following messages:

Pair #p: All you need is love!

Pair #p: Love is not all you need!

Where p stands for the pair number (starting from 1). You should print the first message if there exists at least one valid string L such as both S1 and S2 can be made only of L.

Otherwise, print the second line.

對於每個字串對，您必須列印以下訊息之一：配對#p：你所需要的只是愛！ Pair #p：愛並不是你所需要的全部！

其中 p 代表對數（從 1 開始）。 如果存在至少一個有效字串 L（例如 S1 和 S2 只能由 L 組成），則應列印第一行。否則，列印第二行。

Sample Input 輸入範例

5

11011 11000 11011 11001

111111

100

1000000000

110

1010

100

範例輸出輸出範例

第一對：你所需要的就是愛！

第二對：愛情不是你所需要的全部！

第三對：愛情並不是你所需要的全部！

第四對：你所需要的就是愛！第5對：你所需要的就是愛！

解法：

1. 把 binary 轉成整數，因為想法是一樣的。Ex binary 減去 1，換成整數也是減去 2. 此題就是要找出 2 個數字的最大公因數。如果有找到則為" All you need is love!" 反之為 Love is not all you need!。

/\* UVa 10193 - All You Need Is Love

\* by C111193227林佩璇,

\* 2025/10/08

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 取得資料筆數

int cases = scanner.nextInt();

// 根據資料筆數, 產生 0 ~ 小於資料筆數的索引

for (int caseIndex = 0; caseIndex < cases; caseIndex++) {

String s1 = scanner.next(); // 取得字串 1

String s2 = scanner.next(); // 取得字串 2

// 將二進制的字串轉成十進制的數字

int n1 = Integer.parseInt(s1, 2);

int n2 = Integer.parseInt(s2, 2);

// 使用 while 迴圈持續判斷, 當兩數皆大於 0 時要進行輾轉相除法

// 當其中一個數字為 0 時, while 迴圈就會停止

while (n1 > 0 && n2 > 0) {

// 判斷誰比較大, 大的數字要做為被除數, 小的數字要做為除數, 相除取餘數

if (n1 > n2) {

n1 = n1 % n2;

} else {

n2 = n2 % n1;

}

}

// 取最大數字, 也就是兩者最大公因數

int max = Integer.max(n1, n2);

// 印出測試資料號碼

System.out.print("Pair #" + (caseIndex + 1) + ": ");

// 判斷最大公因數是否大於 1, 若大於 1 則表示有找告數字 L 可以同時組成 s1 與 s2

if (max > 1) {

System.out.println("All you need is love!");

} else {

// 若最大公因數為 1, 根據題目規定是不合法的公因數

System.out.println("Love is not all you need!");

}

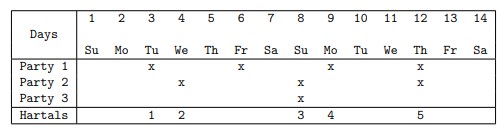
}

}

}

# 10050 –Hartals哈塔爾斯

一個社會研究組織確定了一組簡單的參數來模擬我國政黨的行為。其中一個參數是正整數 h（稱為 hartal 參數），表示對應方呼叫的兩次連續 hartal（罷工）之間的平均天數。儘管該參數過於簡單而無法完美無缺，但它仍然可以用來預測 Hartals 造成的損失。下面的例子會給你一個清晰的概念：考慮三個政黨。假設 h1 = 3、h2 = 4 和 h3 = 8，其中 hi 為第 i 方的 Hartal 參數 (i = 1, 2, 3)。現在，我們將在 N = 14天中模擬這三方的行為。必須始終在星期日開始模擬，並假設每週假期（星期五和星期六）不會有 hartals。



一個社會研究組織確定了一組簡單的參數來模擬我國政黨的行為。 其中一個參數是正整數 h（稱為 hartal 參數），表示對應方呼叫的兩次連續 hartal（罷工）之間的平均天數。 儘管該參數過於簡單而無法完美無缺，但它仍然可以用來預測 Hartals 造成的損失。 下面的例子會給你一個清晰的概念：考慮三個政黨。 假設 h1 = 3、h2 = 4 和 h3 = 8，其中 hi 為第 i 方的 Hartal 參數 (i = 1, 2, 3)。 現在，我們將在 N = 14 天中模擬這三方的行為。必須始終在星期日開始模擬，並假設每週假期（星期五和星期六）不會有 hartals。

Input 輸入

The first line of the input consists of a single integer T giving the number of test cases to follow.The first line of each test case contains an integer N (7 ≤ N ≤ 3650) giving the number of days over which the simulation must be run. The next line contains another integer P (1 ≤ P ≤ 100) representing the number of political parties in this case. The ith of the next P lines contains a positive integer hi (which will never be a multiple of 7) giving the hartal parameter for party i (1 ≤ i ≤ P).

輸入的第一行由一個整數 T 組成，給出了要遵循的測試案例的數量。每個測試

案例的第一行包含一個整數 N (7 ≤ N ≤ 3650)，給出了模擬必須進行的天數跑步。 下一行包含另一個整數 P (1 ≤ P ≤ 100)，表示本例中的政黨數量。 接下來 P 行的第 i 行包含一個正整數 hi（永遠不會是 7 的倍數），給出第 i 方的 hartal 參數 (1 ≤ i ≤ P)。

Output 輸出

For each test case in the input output the number of working days we lose. Each output must be on a separate line.

對於輸入輸出中的每個測試案例，我們損失的工作天數。 每個輸出必須位於單獨的線。

Sample Input 輸入範例

2

14

3

3

4

8 100

4

12

15

25

40

Sample Output 輸出範例

5

15

解法：

1. 使用陣列紀錄哪些天是罷工。
2. 以 7 取餘數，餘 6、餘 0(星期五、星期六)則不需要計算罷工天數。

/\* UVa 10050 - Hartals

\* by C111193227林佩璇,

\* 2025/10/08

\*/

import java.util.Scanner;

import java.util.HashSet;

import java.util.TreeSet;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 取得資料筆數

int cases = scanner.nextInt();

// 根據資料筆數取得相對應測試資料

for (int i = 0; i < cases; i++) {

// 取得天數

int day = scanner.nextInt();

// 取得政黨數量

int groups = scanner.nextInt();

// 宣告串列, 用來記錄罷工日

HashSet<Integer> set = new HashSet<Integer>();

// TreeSet<Integer> set = new TreeSet<Integer>(); 也可以

// 根據政黨數量, 取得政黨罷工週期

for (int j = 0; j < groups; j++) {

// 取得罷工週期, 罷工一天

int cycle = scanner.nextInt();

// 計算此政黨罷工次數, 使用 for 迴圈產生 1 ~ 題目要求的天數

for (int date = 1; date <= day; date++) {

// 將非罷工日排除, 判斷此日除以 7 的餘數是否不為 6(禮拜五) 與 0(禮拜六), 因題目中 1 代表禮拜日

// 判斷此日是否可以被罷工週期整除, 若可以, 表示此天為此政黨的罷工日

if (date % 7 != 6 && date % 7 != 0 && date % cycle == 0) {

set.add(date);

}

}

}

// 印出總罷工天數

System.out.println(set.size());

}

}

}

# 10922 - 2 the 9s 2個9

A well-known trick to know if an integer N is a multiple of nine is to compute the sum S of its digits.If S is a multiple of nine, then so is N. This is a recursive test, and the depth of the recursion needed to obtain the answer on N is called the 9-degree of N.Your job is, given a positive number N, determine if it is a multiple of nine and, if it is, its 9-degree. 要知道整數 N 是否是 9 的倍數，一個眾所周知的技巧是計算其數字之和 S。如果 S 是 9 的倍數，則 N 也是 9 的倍數。這是一個遞歸測試，並且深度獲得 N 上的答案所所需的遞歸稱為 N 的 9 度。您的工作是，給定一個正數 N，確定它是否是 9 的倍數，如果是，則確定它的 9 度。

Input 輸入

The input is a file such that each line contains a positive number. A line containing the number 0 is the end of the input. The given numbers can contain up to 1000 digits.

輸入是一個文件，每行包含一個正數。 包含數字 0 的行是輸入的結尾。 給定的數字最多可以包含 1000 位數字。

Output 輸出

The output of the program shall indicate, for each input number, if it is a multiple of nine, and in case it is, the value of its nine-degree. See the sample output for an example of the expected formatting of the output.

對於每個輸入數字，程式的輸出應指示它是否是九的倍數，如果是，則指示其九度的值。 有關輸出的預期格式的範例，請參閱範例輸出。

Sample Input 輸入範例

999999999999999999999

9

9999999999999999999999999999998

0

Sample Output 輸出範例

999999999999999999999 is a multiple of 9 and has 9-degree 3.

9 is a multiple of 9 and has 9-degree 1.

9999999999999999999999999999998 is not a multiple of 9.

解法：

1. 輸入的數字有 1000 位數，所以無法用數字處理，需要用字串。
2. 設輸入數字為 999，則 degree 為 2。

/\* UVa 10922 - 2 the 9s

\* by C111193227林佩璇,

\* 2025/10/08

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 偵測是否有資料輸入

while (scanner.hasNextLine()) {

// 取得字串 n, 因為數字太大, 所以用字串儲存

String n = scanner.nextLine();

// 判斷數字字串是否等於字串 0

if (n.equals("0")) {

// 結束計算

break;

}

// 根據題目要求將原始的數字資料印出, 因後面會操作此變數, 所以先印出來

System.out.print(n);

// 宣告變數, 用來儲存次數

int degree = 0;

// 使用 while 迴圈來實作無限迴圈, 模擬不斷計算的過程

// 當數字不為 9 的倍數, 或只剩下個位數時, 可呼叫 break 結束 while 迴圈

// 以下使字串 "189" 為例子

while (true) {

// 宣告變數 sum, 儲存每一位數字的加總

int sum = 0;

// 讀取字串中的每一個字元

for (int i = 0; i < n.length(); i++) {

// 將每一個字元轉數字, 並累加到數字加總中

// 逐一讀取字元: '1', '8', '9', 將字元轉數字: 減去字元 '0'

// sum 的加總過程為 1 + 8 + 9 = 18

// 也可以寫成 sum = sum + Character.getNumericValue(n.charAt(i));

sum = sum + (n.charAt(i) - '0');

}

// 判斷加總是否可以被 9 整除(餘數為 0)

// 判斷 18 是否可以被 9 整除

if (sum % 9 == 0) {

// 將次數累加

degree = degree + 1;

// 判斷加總是否等於 9

if (sum == 9) {

// 若加總等於 9 則結束計算

// 代表只剩個位數, 不需再繼續做每一位數字的累加

break;

} else {

// 將數字加總轉成字串, 並重新執行 while 迴圈

// 將數字 18 轉成字串 "18", 將 n 從 "189" 更新為 "18"

n = Integer.toString(sum);

}

} else {

// 若加總不能被 9 整除, 結束計算

break;

}

}

// 判斷次數是否大於 0

if (degree > 0) {

// 表示 n 是 9 的倍數, 並印出次數

System.out.println(" is a multiple of 9 and has 9-degree " + degree + ".");

} else {

// 表示 n 不是 9 的倍數

System.out.println(" is not a multiple of 9.");

}

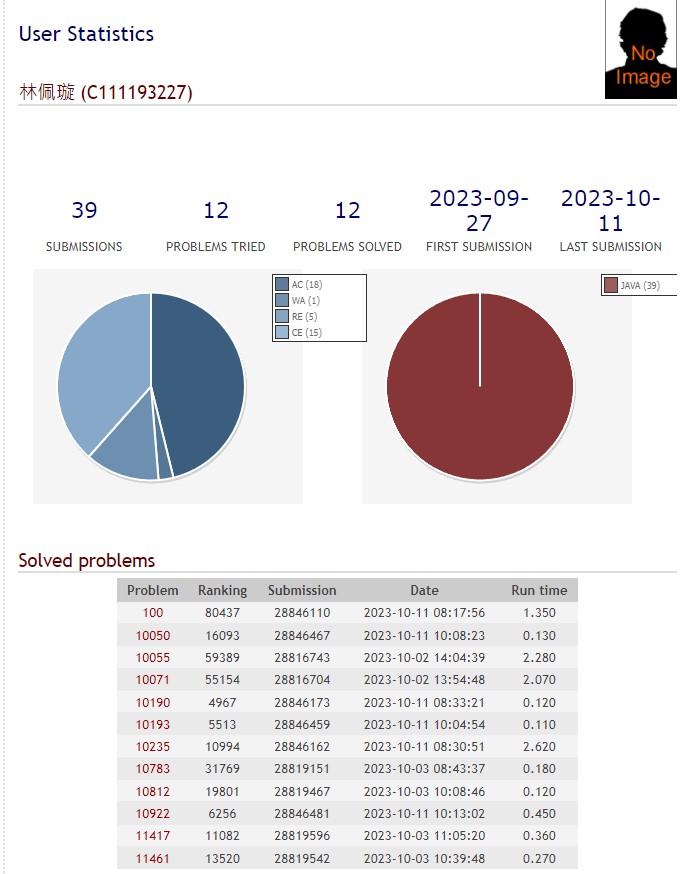
}

}

}

}

}



# 10931 – Parity平價

我們將整數 n 的奇偶校驗定義為以模 2 計算的二進位表示形式的位元之和。例如，數字 21 = 101012 在其二進位表示中包含三個 1，因此它的奇偶校驗為 3(mod2)，即 1。在此問題中，您必須計算整數 1 ≤ I ≤ 2147483647 的奇偶校驗。

我們將整數 n 的奇偶校驗定義為以模 2 計算的二進位表示形式的位元之和。 例如，數字 21 = 101012 在其二進位表示中包含三個 1，因此它的奇偶校驗為 3(mod2)，即 1。在此問題中，您必須計算整數 1 ≤ I ≤ 2147483647 的奇偶校驗。

Input 輸入

Each line of the input has an integer I and the end of the input is indicated by a line where I = 0 that should not be processed. 輸入的每一行都有一個整數 I，輸入的結尾由不應處理的 I = 0 的行指示。

Output 輸出

For each integer I in the inputt you should print a line ‘The parity of B is P (mod 2).’, where B is the binary representation of I.

對於輸入中的每個整數 I，您應該列印一行“The parity of B is P (mod 2).”，其中 B 是 I 的二進位表示。

Sample Input 輸入範例

1

2

10

21

0

Sample Output 輸出範例

The parity of 1 is 1 (mod 2).

The parity of 10 is 1 (mod 2).

The parity of 1010 is 2 (mod 2). The parity of 10101 is 3 (mod 2).

解法：直接使用 Java 提供的 API(Integer.toBinaryString)，因為它回傳 String，所以可以直接解析字串，並計算有幾個 1。再依題目需求印出答案即可。

/\* UVa 10931 - Parity

\* by C111193227林佩璇,

\* 2025/10/15

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 當數字有輸入時

while (scanner.hasNextInt()) {

// 取得數字

int number = scanner.nextInt();

// 判斷數字是否為 0

if (number == 0) {

// 數字為 0, 不需要處理, 結束 while 迴圈

break;

}

// 記錄 2 進制中 1 出現的次數

int count = 0;

// 記錄二進制表達式

String binary = "";

// 使用 while 迴圈做短除法, 當數字大於 0 時, 則表示要進行短除法

while (number > 0) {

// 將數字除以 2(二進制)取餘數, 例如數字 7 除以 2, 會得到餘數 1

int result = number % 2;

// 將餘數加入到二進制的字串中

binary = result + binary;

// 判斷餘數是否為 1

if (result == 1) {

// 將 1 出現的次數加 1

count = count + 1;

}

// 更新被除數, 將數字除以 2 取商數, 例如數字 7 除以 2, 會得到商數 3

number = number / 2;

}

// 輸出

System.out.println("The parity of " + binary + " is " + count + " (mod 2).");

}

}

}

# 10019 - Funny Encryption Method有趣的加密方法有趣的加密方法

ITESM 蒙特雷校區的一名學生正在嘗試一種新的數位加密方法。這些方法由以下步驟組成：步驟：範例 1. 讀取要加密的數字 N ： M = 265 2. 將 N 解釋為十進位數： X1 = 265（十進位） 3. 將 N 的十進位解釋轉換為其二進位表示形式： X1 = 100001001（二進位） 4. 令 b1 等於該二進位表示形式中 1 的數量： b1 = 3 5. 將 N 解釋為十六進位表示： X2 = 265（十六進位） 6. 將N的十六進位解釋轉換為其二進位表示形式： X2 = 1001100101 7. 令 b2 等於最後一個二進位表示形式中 1 的數量： b2 = 5 8. 加密是 M xor (b1 b2) 的結果： 265 xor ( 3\*5) = 262 這位學生沒有通過計算組織，這就是為什麼這位學生向 ITESM Campus Monterrey 內部 ACM 編程競賽的評委詢問這兩種表示形式中 1 的位數，以便他可以繼續比賽。您必須編寫一個程式來讀取數字並給出數字 b1 和 b2 作為輸出

ITESM 蒙特雷校區的一名學生正在嘗試一種新的數位加密方法。 這些方法由以下步驟組成： 步驟：範例 1. 讀取要加密的數字 N：M = 265 2. 將 N 解釋為十進位數：X1 = 265（十進位） 3. 將 N 的十進位解釋轉換為其二進位表示形式： X1 = 100001001（二進位） 4. 令 b1 等於該二進位表示形式中 1 的數量： b1 = 3 5. 將 N 解釋為十六進位： X2 = 265（十六進位） 6. 將 N 的十六進位解釋轉換為其二進位表示形式： X2 = 1001100101 7. 令 b2 等於最後一個二進位表示形式中 1 的數量： b2 = 5 8. 加密是 M xor (b1  b2) 的結果： 265 xor (3\*5) = 262 該學生未通過計算組織，這就是為什麼該學生要求 ITESM Campus Monterrey 內部 ACM 編程競賽的評委詢問這兩個表示中 1 的位數，以便他可以繼續比賽。 您必須編寫一個程式來讀取數字並給出數字 b1 和 b2 作為輸出

Input 輸入

The first line will contain a number N which is the number of cases that you have to process. Each of the following N Lines (0 < N ≤ 1000) will contain the number M (0 < M ≤ 9999, in decimal representation) which is the number the student wants to encrypt.

第一行將包含數字 N，它是您必須處理的案例數。 接下來的 N 行 (0 < N ≤ 1000) 中的每一行都將包含數字 M (0 < M ≤ 9999，十進位表示)，這是學生想要加密的數字。

Output 輸出

You will have to output N lines, each containing the number b1 and b2 in that order, separated by one space corresponding to that lines number to crypt

您必須輸出 N 行，每行依序包含數字 b1 和 b2，並以與要加密的行號相對應的空格分隔

Sample Input 輸入範例

3

265

111

1234

Sample Output 輸出範例

3 5

6 3

5 5

解法：依計概所教的方法去求出 2 進位，並計算有多少個 1。

/\* UVa 10019 - Funny Encryption Method

\* by C111193227林佩璇,

\* 2025/10/15

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 取得資料筆數

int cases = scanner.nextInt();

// 根據資料筆數取得, 產生 0 ~ 小於資料筆數的索引位置

for (int i = 0; i < cases; i++) {

// 取得數字, 根據題目描述, 此數字為 10 進制數字

int number = scanner.nextInt();

// 將 10 進制的數字轉 2 進制, 並取得 1 出現的 2 進制表達式的次數

int b1 = getBinaryOneCount(number);

// 用來儲存 16 進位轉 10 進位的數字

int hexToDecimalNumber = 0;

// 用來儲存 16 的次方數, 轉 10 進位的過程會用到

int power = 0;

// 將 16 進制的數字轉 10 進制

// 使用 while 迴圈處理字串中的每一個位數, 當數字大於 0 時, 則表示要進行轉換

while (number > 0) {

// 將數字除以 10 取餘數, 取得數字的最後一個數字, 例如數字 123 以 10 取餘數, 會得到餘數 3

int result = number % 10;

// 將這一位數字乘上 16 所對應的次方數, 就是 16 轉 10 進制的結果

hexToDecimalNumber = hexToDecimalNumber + result \* (int) Math.pow(16, power);

// 更新被除數, 將數字除以 10 取商數, 例如數字 123 以 10, 會得到商數 12

number = number / 10;

// 讀取每一位數字時, 16 的次方數就要加 1

power = power + 1;

}

// 將 10 進制的數字轉 2 進制, 並取得 1 出現的 2 進制表達式的次數

// 呼叫此方法須傳入一個數字

int b2 = getBinaryOneCount(hexToDecimalNumber);

// 印出 b1 與 b2, 中間以空白隔開

System.out.println(b1 + " " + b2);

}

}

// 將 10 進制的數字轉 2 進制, 並取得 1 出現的 2 進制表達式的次數

public static int getBinaryOneCount(int number) {

// 記錄 2 進制中 1 出現的次數

int count = 0;

// 使用 while 迴圈做短除法, 當數字大於 0 時, 則表示繼續進行短除法

while (number > 0) {

// 將數字除以 2(二進制)取餘數, 例如數字 7 除以 2, 會得到餘數 1

// 判斷餘數是否為 1

if (number % 2 == 1) {

// 將 1 出現的次數加 1

count = count + 1;

}

// 更新被除數, 將數字除以 2 取商數, 例如數字 7 除以 2, 會得到商數 3

number = number / 2;

}

return count;

}

}

# 948 - Fibonaccimal Base斐波那契基數

眾所周知的斐波那契數列是從 0 和 1 開始，然後將最後兩個數字相加得到下一個數字。例如，序列中的第三個數字是1 (1=1+0)，第四個數字是2 (2=1+1)，第五個數字是3 (3=2+1)，依此類推。

眾所周知的斐波那契數列是從 0 和 1 開始，然後將最後兩個數字相加得到下一個數字。 例如，序列中的第三個數字是 1 (1=1+0)，第四個數字是 2 (2=1+1)，第五個數字是 3 (3=2+1)，依此類推。



這個序列出現在我們生活和自然界的許多事物中，並且具有重大意義。除此之外，您是否知道所有正整數都可以表示為斐波那契數列中的數字總和？不僅如此，所有正整數都可以表示為一組斐波那契數的和，即序列中的數字，不重複。例如：13 可以是集合 {13}、{5,8} 或 {2,3,8} 的和，17 以 {1,3,13} 或 {1,3,5,8} 表示。由於所有數字都具有此屬性（您想嘗試自己證明這一點嗎？），該集合可能是用作表示數字的“基礎”的好方法。但是，正如我們所看到的，有些數字有多個集合，其總和就是該數字。我們要怎麼解決這個問題呢？簡單的！如果我們加入集合不能有兩個連續的斐波那契數的約束，那麼每個數都有一個唯一的表示！這個限制是因為任兩個連續的斐波那契數和就是後面的斐波那契數。現在我們知道了這一切，我們可以準備一個很好的方法來表示任何正整數。我們將使用二進位序列（只有零和一）來做到這一點。例如，17 = 1 + 3 + 13（請記住，不能使用兩個連續的斐波那契數）。讓我們從右側開始，為每個未使用的斐波那契數寫一個 0，為每個使用的斐波那契數寫一個 1。那麼，17 = 100101。詳細解釋見圖2。在這個表示法中，我們不應該在左邊有零，也就是說，我們應該只從第一個開始寫。為了讓您更好地理解，請注意，在該方案中，不使用兩個連續的斐波那契數意味著二進位序列不會有兩個連續的斐波那契數。當我們使用這種表示形式表示數字時，我們說我們正在使用斐波那契數底，我們將其寫為 17 = 100101 (fib)。

這個序列出現在我們生活和自然界的許多事物中，並且具有重大意義。 除此之外，您是否知道所有正整數都可以表示為斐波那契數列中的數字總和？ 不僅如此，所有正整數都可以表示為一組斐波那契數的和，即序列中的數字，不重複。 例如：

13 可以是集合 {13}、{5,8} 或 {2,3,8} 的和，17 以 {1,3,13} 或 {1,3,5,8} 表示。 由於所有數字都具有此屬性（您想嘗試自己證明這一點嗎？），該集合可能是用作表示數字的“基礎”的好方法。 但是，正如我們所看到的，有些數字有多個集合，其總和就是該數字。 我們要怎麼解決這個問題呢？ 簡單的！ 如果我們加入集合不能有兩個連續的斐波那契數的約束，那麼每個數都有一個唯一的表示！ 這個限制是因為任兩個連續的斐波那契數和就是後面的斐波那契數。 現在我們知道了這一切，我們可以準備一個很好的方法來表示任何正整數。 我們將使用二進位序列

（只有零和一）來做到這一點。 例如，17 = 1 + 3 + 13（請記住，不能使用兩個連續的斐波那契數）。 讓我們從右側開始，為每個未使用的斐波那契數寫一個 0，為每個使用的斐波那契數寫一個 1。 那麼，17 = 100101。詳細解釋見圖 2。 在這個表示法中，我們不應該在左邊有零，也就是說，我們應該只從第一個開始寫。 為了讓您更好地理解，請注意，在該方案中，不使用兩個連續的斐波那契數意味著二進位序列不會有兩個連續的斐波那契數。 當我們使用這種表示形式表示數字時，我們說我們正在使用斐波那契數底，我們將其寫為 17 = 100101 (fib)。



Given a set of numbers in decimal base, your task is to write them in the Fibonaccimal base. 給定一組十進制數字，您的任務是將它們寫入斐波那契基數。

Input 輸入

The first line of input contains a single number N, representing the quantity of numbers that follow (1 ≤ N ≤ 500). Than follow exactly N lines, each one containing a single positive integer smaller than 100 000 000. These numbers can come in any order.

輸入的第一行包含一個數字 N，表示後面的數字數量 (1 ≤ N ≤ 500)。 緊接著 N 行，每行包含一個小於 100 000 000 的正整數。這些數字可以以任何順序出現。

Output 輸出

You should output a single line for each of the N integers in the input, with the format

‘DEC BASE = F IB BASE (fib)’. DEC BASE is the original number in decimal base and F IB BASE is its representation in Fibonaccimal base. See the sample output for an example. 您應該為輸入中的 N 個整數輸出一行，格式為「DEC BASE = F IB BASE (fib)」。 DEC BASE 是十進制的原始數字，F IB BASE 是其斐波那契數的表示方式。 有關範例，請參閱範例輸出。

Sample Input 輸入範例

10

1

2

3

4

5

6

7

8

9

10

Sample Output 輸出範例

1. = 1 (fib)
2. = 10 (fib)
3. = 100 (fib)
4. = 101 (fib)
5. = 1000 (fib)
6. = 1001 (fib)
7. = 1010 (fib)
8. = 10000 (fib)
9. = 10001 (fib)
10. = 10010 (fib) 解法： 1.先建立費氏數列表。

2. 從表中找出與 N(輸入的數字)相差最小的數字，並做相減，並使用一個字串紀錄，假如有相減到則是 1，沒有減到則是 0，一直迴圈，直到 N=0 為止。

/\* UVa 948 - Fibonaccimal Base

\* by C111193227林佩璇,

\* 2025/10/15

\*/

import java.util.Scanner;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 取得資料筆數

int cases = scanner.nextInt();

// 建立串列, 儲存費式數列

ArrayList<Integer> fiblist = new ArrayList<Integer>();

fiblist.add(1); // 題目目的表述使用的數字, 第一個為 1, 將 1 加入串列中

fiblist.add(2); // 題目目的表述使用的數字, 第二個為 2, 將 2 加入串列中

// 根據資料筆數取得數字, 產生 0 ~ 小於資料筆數的索引位置

for (int i = 0; i < cases; i++) {

// 取得原始數字

int number = scanner.nextInt();

// 根據題目要求印出原始數字! = 等號前要有空白

System.out.print(number + " = ");

// 產生足夠的費式數列

// 判斷原始數字是否大於串列的最後一個數字, 表示目前的費式數列不夠用, 要產生新的費式數

while (number > fiblist.get(fiblist.size() - 1)) {

// 取得串列中最後一個費式數

int previousOne = fiblist.get(fiblist.size() - 1);

// 取得串列中倒數第二個費式數

int previousTwo = fiblist.get(fiblist.size() - 2);

// 新的費式數 = 前一個費式數 + 前兩個費式數

fiblist.add(previousOne + previousTwo);

}

// 標記是否可以開始印出費式表述式的 0

boolean isPrintZero = false;

// 計算原始數字的費式表述式, 產生最後一個~第一個費式數的索引位置, 逐次遞減

for (int j = fiblist.size() - 1; j >= 0; j--) {

// 取得特定索引引的費式數

int fib = fiblist.get(j);

// 判斷原始數字是否大於等於目前的費式數

if (number >= fib) {

// 將原始數字減掉目前的費式數

number = number - fib;

// 找到可表述的費式數印出 1

System.out.print("1");

// 將開始印出設為 true

isPrintZero = true;

} else if (isPrintZero) {

// 找不到可表述的費式數印出 0

System.out.print("0");

}

}

// 根據題目要求 (fib) 結尾的字串, 括號前面要有空白

System.out.println(" (fib)");

}

}

}

# 11005 - Cheapest Base最便宜的基礎

在紙張上列印文字時，我們需要墨水。但並非每個字元都需要相同量的墨水來列印：「W」、「M」和「8」等字母比「i」、「c」和「1」等較細的字母更昂貴。在這個問題中，我們將評估在幾個基地列印數字的成本。如您所知，數字可以用幾種不同的基數表示。眾所周知的基數有二進位（基數 2；數字 0 和 1）、十進制（基數 10；數字 0 到 9）和十六進位（基數 16；數字 0 到 9 和字母 A 到 F）。對於一般基數n，我們將使用字串“0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ”的前n 個字符，這意味著該問題中的最高基數是36。最低基數當然是2。該字串中的每個字符都有一個關聯的成本，表示為1 到 128 之間的整數值。以特定基數列印數字的成本是表示該數字所需的所有字元的成本總和。對於輸入中給出的數字，您必須計算最便宜的基數來表示該數字。任何基數中的數字都會列印，且不含前導零。

在紙張上列印文字時，我們需要墨水。 但並非每個字元都需要相同數量的墨水來列印：「W」、「M」和「8」等字母比「i」、「c」和「1」等較細的字母更昂貴。在這個問題中，我們將評估在幾個基地列印數字的成本。 如您所知，數字可以用幾種不同的基數表示。 眾所周知的基數有二進位（基數 2；數字 0 和 1）、十進制（基數 10；數字 0 到 9）和十六進位（基數 16；數字 0 到 9 和字母

A 到 F）。 對於一般基數 n，我們將使用字串

“0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ”的前 n 個字符，這意味

著該問題中的最高基數是 36。最低基數當然是 2。該字串中的每個字符都有一個關聯的成本，表示為 1 到 128 之間的整數值。以特定基數列印數字的成本是表示該數字所需的所有字元的成本總和。 對於輸入中給出的數字，您必須計算最便宜的基數來表示該數字。任何基數中的數字都會列印，且不含前導零。

Input 輸入

輸入的測試案例少於 25 個。輸入檔案的第一行表示測試案例的數量。每個測試案例的描述如下： 每個用例的前 4 行各包含 9 個整數：按上述順序排列的 36 個字元的成本。然後是一行中的查詢數量。每個查詢單獨出現在一行上，並由 0 到 2000000000 之間的十進制數字組成。

輸入的測試案例少於 25 個。 輸入檔案的第一行表示測試案例的數量。 每個測試案例的描述如下： 每個用例的前 4 行各包含 9 個整數：按上述順序排列的 36 個字元的成本。 然後是一行中的查詢數量。 每個查詢單獨出現在一行上，並由 0 到 2000000000 之間的十進制數字組成。

輸出輸出

輸入的測試案例少於 25 個。輸入檔案的第一行表示測試案例的數量。每個測試案例的描述如下： 每個用例的前 4 行各包含 9 個整數：按上述順序排列的 36 個字元的成本。然後是一行中的查詢數量。每個查詢單獨出現在一行上，並由 0 到 2000000000 之間的十進制數字組成。

輸入的測試案例少於 25 個。 輸入檔案的第一行表示測試案例的數量。 每個測試案例的描述如下： 每個用例的前 4 行各包含 9 個整數：按上述順序排列的 36 個字元的成本。 然後是一行中的查詢數量。 每個查詢單獨出現在一行上，並由 0 到 2000000000 之間的十進制數字組成。

Sample Input 輸入範例

2

1. 8 12 13 15 13 13 16 9
2. 18 24 21 23 23 23 13 15

17 33 21 23 27 26 27 19 4

22 18 30 30 24 16 26 21 21

5

98329921

12345

800348

14

873645

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

4

0

1

10

100

Sample Output 輸出範例

Case 1:

號碼 98329921 最便宜的鹼基：24

號碼 12345 最便宜的鹼基： 13 31

號碼 800348 最便宜的鹼基：31

14號最便宜的鹼：13

號碼 873645 最便宜的鹼基：22

案例2：

數字 0 的最便宜鹼基： 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

23 24 25 26 27 28 29 30 31 32 33 34 35 36

數字 1 最便宜的鹼： 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

23 24 25 26 27 28 29 30 31 32 33 34 35 36

10號最便宜的鹼： 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

28 29 30 31 32 33 34 35 36

Cheapest base(s) for number 100: 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

28 29 30 31 32 33 34 35 36

解法：直接計算每個進位的數字，在計算此數字所需要多少錢

/\* UVa 11005 - Cheapest Base

\* by C111193227林佩璇,

\* 2025/10/22

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int count = 0;

// 取得資料組數

int cases = scanner.nextInt();

// 輸出時須要使用的組數編號

// 取得資料組數

int length = scanner.nextInt();

int ink[] = new int[length];

// 根據資料組數取得數字, 產生 0~小於資料組數的索引位置

for (int caseIndex = 0; caseIndex < cases; caseIndex++) {

count = count + 1;

System.out.println("Case " + count + ":"); // 根據題目要求印出

// 產生 0~35 對應的數字, 讀取墨水成本

for (int i = 0; i < length; i++) {

// 讀取墨水成本, 存入陣列的索引位置中

ink[i] = scanner.nextInt();

}

// 取得查詢數量

int queries = scanner.nextInt();

// 根據查詢數量, 取得查詢的數字, 產生 0~小於查詢的索引位置

for (int q = 0; q < queries; q++) {

// 取得查詢的數字

int number = scanner.nextInt();

// 印出查詢數字

System.out.print("Cheapest base(s) for number " + number + ":");

// 記錄最小的成本, 初始值為一個很大的數, 讓第一次計算的成本取代初始值

int min = Integer.MAX\_VALUE;

// 宣告成本陣列, 儲存轉成 2~36 進制後所須的墨水成本

// 因使用進制數直接對應索引位置, 最大的數字是 36, 則陣列長度須為 37

int[] cost = new int[37];

// 使用 for 迴圈產生 2~36

for (int base = 2; base <= 36; base++) {

// 宣告被除數, 初始值為查詢的數字

int targetNumber = number;

// 當被除數大於 0, 表示要繼續進行短除法

while (targetNumber > 0) {

// 被除數 除以 進制數後取餘數, 此餘數可對應到墨水的索引位置

int position = targetNumber % base;

// 此進制的墨水成本 = 使用餘數讀取成本的累加

cost[base] = cost[base] + ink[position];

// 更新被除數, 將被除數 除以 進制數後取商數

targetNumber = targetNumber / base;

}

// 判斷此進制的成本是否小於目前的最小值

if (cost[base] < min) {

// 若小於目前的最小時, 則將此進制的成本更新至最小值

min = cost[base];

}

}

// 使用 for 迴圈, 檢查 2~36 進制的墨水成本

for (int base = 2; base <= 36; base++) {

// 判斷此進制的墨水成本是否等於目前的最小值

if (cost[base] == min) {

// 若相等, 則印出進制數

System.out.print(" " + base);

}

}

// 每筆查詢數字要換行

System.out.println();

}

// 判斷是不是最後一組

if (caseIndex != cases - 1) {

// 如果不是最後一組, 每組資料與下一組資料要間隔一行

System.out.println();

}

}

}

}

# 10041-Vito's family維託的家人

世界知名的黑幫大佬維托·戴德斯通 (Vito Deadstone) 即將移居紐約。他在那裡有一個很大的家庭，全都住在拉馬菲亞大道。由於他會經常拜訪所有親戚，因此他正在努力尋找靠近他們的房子。維托想要最小化到他們所有人的總距離，並勒索你寫一個程式來解決他的問題。

世界知名的黑幫大佬維托·戴德斯通 (Vito Deadstone) 即將移居紐約。 他在那裡有一個很大的家庭，全都住在拉馬菲亞大道。 由於他會經常拜訪所有親戚，因此他正在努力尋找靠近他們的房子。 維托想要最小化到他們所有人的總距離，並勒索你寫一個程式來解決他的問題。

Input 輸入

輸入由多個測試用例組成。第一行包含測試案例的數量。對於每個測試案例，您將獲得整數個親戚 r (0 < r < 500) 和街道號碼（也是整數） s1, s2, ...。 。 。 ,斯 , . 。 。 , sr 他們居住的地方 (0 < si < 30000 )。

請注意，幾個親戚可能住在同一個街道號碼。

輸入由多個測試用例組成。 第一行包含測試案例的數量。 對於每個測試案例，您將獲得整數個親戚 r (0 < r < 500) 和街道號碼（也是整數） s1, s2,...。 。 。 , 斯 , . 。 。 , sr 他們居住的地方 (0 < si < 30000 )。 請注意，幾個親戚可能住在同一個街道號碼。

Output 輸出

For each test case your program must write the minimal sum of distances from the optimal Vito’s house to each one of his relatives. The distance between two street numbers si and sj is dij = |si − sj |.

對於每個測試案例，您的程式必須寫入從最佳維託的房子到他的每個親戚的最小距離總和。 兩個門牌號碼 si 和 sj 之間的距離為 dij = |si − sj |.

Sample Input 輸入範例

2

1. 2 4
2. 2 4 6

Sample Output 輸出範例

2

1. 解法：1. 先將資料進行排序。

2. 找出中位數。(中位數跟其他數的差總和會最小)

/\* UVa 10041 - Vito's family

\* by C111193227林佩璇,

\* 2025/10/22

\*/

import java.util.Scanner;

import java.util.Arrays;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 取得測試資聊的筆數

int cases = scanner.nextInt();

// 根據資料筆數, 逐一讀取測試資料

for (int i = 1; i <= cases; i++) {

// 取得親戚數量

int number = scanner.nextInt();

// 宣告一個陣列, 用來記錄親戚住在哪個街

int[] streets = new int[number];

// 根據親戚數量, 逐一讀取親戚的街道位置

for (int j = 0; j < number; j++) {

// 將街道位置存入陣列中

streets[j] = scanner.nextInt();

}

// 排序親戚街道位置, 由小至大, 方便計算中位數

Arrays.sort(streets);

// 計算中位數

int middle = streets[number / 2];

// 宣告整數, 用來儲存與每位親戚距離的總和

int sum = 0;

// 使用 foreach, 將親戚家街道號碼逐一與中位數相減, 使用 Math.abs() 取絕對值

for (int street : streets) {

sum = sum + Math.abs(middle - street);

}

// 印出最短距離之總和

System.out.println(sum);

}

}

}

# 10057-A mid-summer night’s dream 仲夏夜之夢。

這是西元2200年。兩百年來科學取得了很大進展。這裡提到200年，是因為這個問題是藉助時光機回到西元2000年。現在可以在人與電腦CPU之間建立直接連線。人們可以在3D顯示器（也就是今天的顯示器）上觀看別人的夢想，就像看電影一樣。本世紀的一個問題是，人們已經變得如此依賴計算機，以至於他們的分析能力接近零。計算機現在可以讀取問題並自動解決它們。但它們只能解決困難的問題。現在不存在簡單的問題了。我們的首席科學家遇到了大麻煩，因為他忘記了密碼鎖的號碼。出於安全原因，現今的電腦無法解決密碼鎖相關問題。在一個仲夏夜，科學家做了一個夢，他看到許多無符號整數飛來飛去。他在電腦的幫助下記錄它們，然後他有一條線索，如果數字是（X1，X2， ...， Xn），他將必須找到一個整數A（這個A是密碼鎖代碼），這樣那

(|X1 - A| + |X2 - A| + . . . + |Xn - A|)

是最小值。

這是西元 2200 年。 兩百年來科學取得了很大進展。 這裡提到 200 年，是因為這個問題是藉助時光機回到西元 2000 年。 現在可以在人與電腦 CPU 之間建立直接連線。 人們可以在 3D 顯示器（也就是今天的顯示器）上觀看別人的夢想，就像看電影一樣。 本世紀的一個問題是，人們已經變得如此依賴計算機，以至於他們的分析能力接近零。 計算機現在可以讀取問題並自動解決它們。 但它們只能解決困難的問題。 現在不存在簡單的問題了。 我們的首席科學家遇到了大麻煩，因為他忘記了密碼鎖的號碼。 出於安全原因，現今的電腦無法解決密碼鎖相關問題。在一個仲夏夜，科學家做了一個夢，他看到許多無符號整數飛來飛去。 他在電腦的幫助下記錄它們，然後他有一條線索，如果數字是（X1，X2，...，Xn），他將必須找到一個整數 A（這個 A 是密碼鎖代碼），這樣那是最小值。

Input 輸入

Input will contain several blocks. Each block will start with a number n (0 < n ≤ 1000000) indicating how many numbers he saw in the dream. Next there will be n numbers. All the numbers will be less that 65536. The input will be terminated by end of file.

輸入將包含多個區塊。 每個區塊都會以數字 n (0 < n ≤ 1000000) 開始，表示他在夢中看到了多少個數字。 接下來會有 n 個數字。 所有數字都將小於 65536。輸入將在文件末尾終止。

輸出輸出

對於每組輸入，都會有一行輸出。該行將包含 A 的最小可能值。接下來，它將包含輸入中有多少個數字滿足 A 的屬性（與 A 的絕對偏差總和最小）。最後，您必須列印 A 有多少個可能的不同整數值（這些值不需要出現在輸入中）。這些數字將以一個空格分隔。

對於每組輸入，都會有一行輸出。 該行將包含 A 的最小可能值。接下來，它將包含輸入中有多少個數字滿足 A 的屬性（與 A 的絕對偏差總和最小）。 最後，您必須列印 A 有多少個可能的不同整數值（這些值不需要出現在輸入中）。 這些數字將以一個空格分隔。

Sample Input 輸入範例

2

10

10

4

1

2

2

4

Sample Output 輸出範例

10 2 1

2 2 1

解法：每個資料的第一個數字為有多少資料需要被處理(cases)，接下來就是資料(X1、

X2 ... Xn)。

1. 找出中位數。
2. 計算有幾個和中位數一樣的數字。資料(cases)是偶數個時中位數有 2 個、資料是奇數個時中位數有 1 個。
3. 找出可能有幾種最小值(包含不在所輸入的資料裡面)，資料為奇數個時答案為 1，資料為偶數個時答案為 2 個中位數相減加 1。

/\* UVa 10057 - A mid-summer night’s dream

\* by C111193227林佩璇,

\* 2025/10/22

\*/

import java.util.Scanner;

import java.util.Arrays;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 偵測是否有輸入

while (scanner.hasNextInt()) {

// 取得陣列長度

int size = scanner.nextInt();

// 宣告陣列用來儲存讀取到的數字

int[] numbers = new int[size];

// 使用 for 迴圈, 產生 0 ~ 小於陣列大小的索引, 逐一讀取陣列中的每一個數字

for (int i = 0; i < size; i++) {

// 將讀取到的數字存入陣列

numbers[i] = scanner.nextInt();

}

// 將陣列由小到大排序

Arrays.sort(numbers);

// 取得中位

// 例如: 讀取數列 2 4 4 6 6 8, 中間的數字為 4 與 6

// 例如: centerNumber1 = numbers[(size - 1) / 2]; // 4 的索引位置 = (陣列長度 / 2) - 1

// int centerNumber2 = numbers[size / 2]; // 6 的索引位置 = 陣列長度 / 2

// 如果是奇數個數列, size / 2 = (size - 1) / 2, 會存取相同的索引, 也就是同一個數值

int centerNumber1 = numbers[(size - 1) / 2];

int centerNumber2 = numbers[size / 2];

// 統計與中位數相同的數字數量

int sameNumbers = 0;

// 使用 for 產生 0 ~ 小於陣列長度的索引, 逐一讀取陣列中的數字

for (int i = 0; i < size; i++) {

// 判斷陣列中的某一個值是否等於中位數, 例如: 是否等於 4 或是是否等於 6

if (numbers[i] == centerNumber1 || numbers[i] == centerNumber2) {

// 若數值等於中位數, 則歸納次數

// 例如: 4 出現 2 次, 6 出現 2 次, 所以與中位數相同的數字共有 4 個

sameNumbers = sameNumbers + 1;

}

}

// 計算中位數可能的數值, 例如 4~6, 總共有 3 種數值: 4, 5, 6, 也可以由 6-4+1 推得

int possibleNumbers = centerNumber2 - centerNumber1 + 1;

// 找出最小的中位數, 中位數在數列中出現的次數, 中位數有幾種可能

// 例如: 數列 2 4 4 6 6 8

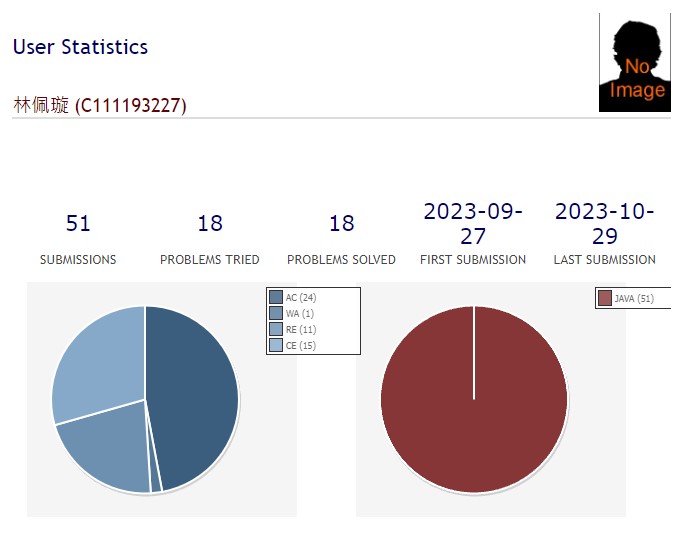
// 最小中位數是 4, 數列中與中位數(4或6)相同的數字有 4 個, 可能的情況有 3 種(數值4, 5, 6)

System.out.println(centerNumber1 + " " + sameNumbers + " " + possibleNumbers);

}

}

}



# 10035 - Primary Arithmetic初級算術

Children are taught to add multi-digit numbers from right-to-left one digit at a time. Many find the “carry” operation - in which a 1 is carried from one digit position to be added to the next - to be a significant challenge. Your job is to count the number of carry operations for each of a set of addition problems so that educators may assess their difficulty.

孩子們被教導從右到左一次一位數地添加多位數字。 許多人發現「進位」操作（其中將 1 從一位數字位置進位並添加到下一位數字位置）是一個重大挑戰。 您的工作是計算一組加法問題中每個問題的進位運算數量，以便教育工作者可以評估其難度。

Input 輸入

Each line of input contains two unsigned integers less than 10 digits. The last line of input contains ‘0 0’. 每行輸入包含兩個小於 10 位元的無符號整數。 輸入的最後一行包含“0 0”。

Output 輸出

For each line of input except the last you should compute and print the number of carry operations that would result from adding the two numbers, in the format shown below.

對於除最後一行之外的每一行輸入，您應該計算並列印兩個數字相加所產生的進位運算的數量，格式如下所示。

Sample Input 輸入範例

123 456

555 555

123 594

0 0

Sample Output 輸出範例

No carry operation.

3 carry operations.

1 carry operation.

解法：使用 StringBuilder 建立 String，主要是因為 StringBuilder 提供字串反轉的

功能，把字串反轉後計算進位幾次會比較方便。開始計算是否進位，並印出結果。

/\* UVa 10035 - Primary Arithmetic

\* by C111193227林佩璇,

\* 2025/10/29

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 偵測是否有資料輸入

while (scanner.hasNextInt()) {

// 取得數字 a

int a = scanner.nextInt();

// 取得數字 b

int b = scanner.nextInt();

// 檢查 a 與 b 是否都等於 0

if (a == 0 && b == 0) {

// 結束計算

break;

}

// 記錄每位數加總結果

int sum = 0;

// 記錄進位

int carry = 0;

// 記錄進位次數

int count = 0;

// 開始進位進行相加

// 當 a 或 b 不為 0 時, 表示還有數字還沒被加總

while (a != 0 || b != 0) {

// 取得 a 與 b 兩數的最後一個數字進行相加, 並加上進位數

// 最後一個數字可透過除以 10 取餘數取得, 例如 123 除以 10, 餘數為 3

sum = a % 10 + b % 10 + carry;

// 判斷相加結果是否大於 10

if (sum >= 10) {

// 累加進位次數

count = count + 1;

// 進位數字設為 1

carry = 1;

} else {

// 若相加小於 10, 進位數字設為 1

carry = 0;

}

// 更新數字, 除以 10 取商數, 即可得到未處理的數字

a = a / 10;

b = b / 10;

}

// 根據進位次數印出字串

if (count == 0) {

System.out.println("No carry operation.");

} else if (count == 1) {

System.out.println("1 carry operation.");

} else {

System.out.println(count + " carry operations.");

}

}

}

}

# 10929 - You can say 19你可以說11你可以說11

Your job is, given a positive number N, determine if it is a multiple of eleven. 你的工作是，給定一個正數 N，確定它是否是 11 的倍數。

Input 輸入

The input is a file such that each line contains a positive number. A line containing the number ‘0’ is the end of the input. The given numbers can contain up to 1000 digits. 輸入是一個文件，每行包含一個正數。 包含數字“0”的行是輸入的結尾。 給定的數字最多可以包含 1000 位數字。

Output 輸出

The output of the program shall indicate, for each input number, if it is a multiple of eleven or not. 對於每個輸入數字，程式的輸出應指示它是否是 11 的倍數。

Sample Input 輸入範例

112233

30800

2937

323455693

5038297

112234

0

Sample Output 輸出範例

112233 is a multiple of 11.

30800 is a multiple of 11.

2937 is a multiple of 11.

323455693 is a multiple of 11.

5038297 is a multiple of 11.

112234 is not a multiple of 11.

解法：

1.數字最大會有 1000 位數，故不是合用數字處理。

2.使用 String 儲存數字，並且使用 11 倍數的判別法( 奇數-偶數=11 倍數(包括 0)，即是 11 倍數)。

/\* UVa 10929 - You can say 19

\* by C111193227林佩璇,

\* 2025/10/29

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 偵測是否有資料輸入

while (scanner.hasNext()) {

// 取得文字

String numberText = scanner.next();

// 判斷文字是否等於字串"0"

if (numberText.equals("0")) {

// 結束計算

break;

}

// 記錄奇數位置總和

int odd = 0;

// 記錄偶數位置總和

int even = 0;

// 使用 for 迴圈產生 0 ~ 小於字串長度的索引, 準備讀取字串中的每一個字元

for (int i = 0; i < numberText.length(); i++) {

// 將讀取到的字元, 減去字元 '0', 可將此字元轉數字, 或使用 Character.getNumericValue() 也可以

int number = Character.getNumericValue(numberText.charAt(i));

// 判斷是否為偶數個字元

if ((i % 2) == 0) {

// 加到偶數位置的數字

even = even + number;

} else {

// 加到奇數位置的數字

odd = odd + number;

}

}

// 判斷奇數位總和和偶數位總和, 是否可被 11 整除

if (((odd - even) % 11) == 0) {

// 若餘數為 0, 則為 11 的倍數

System.out.println(numberText + " is a multiple of 11.");

} else {

// 若餘數不為 0, 則為 11 的倍數

System.out.println(numberText + " is not a multiple of 11.");

}

}

}

}

# 10062 - Tell me the frequencies! 告訴我頻率！

Given a line of text you will have to find out the frequencies of the ASCII characters present in it. The given lines will contain none of the first 32 or last 128 ASCII characters. Of course lines may end with \n and \r but always keep those out of consideration.

給定一行文本，您必須找出其中出現的 ASCII 字元的頻率。 給定的行將不包含

前 32 個或後 128 個 ASCII 字元。 當然，行可能以 \n 和 \r 結尾，但始終不要考慮這些。

Input 輸入

Several lines of text are given as input. Each line of text is considered as a single input.

Maximum length of each line is 1000.

給出幾行文字作為輸入。 每行文字都被視為一個輸入。 每行的最大長度為

1000。

Output輸出

根據下面給定的格式列印出現的 ASCII 字元的 ASCII 值及其頻率。應使用空白行分隔每組輸出。依出現頻率的升序列印 ASCII 字元。如果同時存在兩個字符，則先列印 ASCII 值較高的 ASCII 字符的資訊。輸入由文件末尾終止。

根據下面給定的格式列印出現的 ASCII 字元的 ASCII 值及其頻率。 應使用空白行分隔每組輸出。 依出現頻率的升序列印 ASCII 字元。 如果同時存在兩個字符，則先列印 ASCII 值較高的 ASCII 字符的資訊。 輸入由文件末尾終止。

Sample Input 輸入範例

AAABBC

122333

Sample Output 輸出範例

67 1

66 2

65 3

1. 1
2. 2
3. 3

解法：

/\* UVa 10062 - Tell me the frequencies!

\* by C111193227林佩璇,

\* 2025/10/29

\*/

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 判斷是否印出換行, 第一筆資料不用印, 但第二筆開始都要多印換行

boolean isPrintLine = false;

// 偵測是否有輸入

while (scanner.hasNextLine()) {

// 判斷 isPrintLine 是否為 true

if (isPrintLine) {

// 為 true, 印出換行

System.out.println();

} else {

// 讀取第一筆資料將 isPrintLine 設為 false, 將 isPrintLine 更新為 true

// 下一次進來 while 迴圈時, 則會進入 if 印出換行

isPrintLine = true;

}

// 讀取測試資料

String text = scanner.nextLine();

// 建議陣列, 記錄 ascii code 出現的次數, 將索引位置當作 ascii code 使用

// 題目會使用到 ascii code 128, 故將 128 當作索引使用, 陣列長度要是 129

int[] table = new int[129];

// 使用 for 迴圈產生 0 ~ 小於字串長度的索引, 逐一讀取字串中的每一個字元

for (int i = 0; i < text.length(); i++) {

// 讀取特定索引的字元

char symbol = text.charAt(i);

// 將 char 可以轉為 int, 可直接透過字元操作符號的索引位置, 將該 ascii code 的出現次數 + 1

// 例如字元 A 是 65, 即可操作 table[65] 儲存的出現次數

table[symbol] = table[symbol] + 1;

}

// 根據題目要求印出出現次數由大到小排列, 像樣的寫法為 1(0次不須處理), 逐次遞增, 出現次數只會小於等於字串長度

// 例如字串: AAAAA, 字串長度是 5, A 的出現次數就是 5 次

// 用 for 迴圈由小至大尋找次數, 逐一檢查是否有 ascii code 的出現次數符合目前尋找的次數

for (int count = 1; count <= text.length(); count++) {

// 根據題目要求, 印出次數相同, 由大寫小寫 ascii code

// 使用 for 迴圈由大到小, 初始以 ascii code 設陣列長度-1 開始, 逐次遞減

for (int asciiCode = table.length - 1; asciiCode >= 0; asciiCode--) {

// 判斷特定 asciiCode 的出現次數是否等於目前尋找的次數

if (table[asciiCode] == count) {

// 若相等, 則印出 ascii code 與出現次數

System.out.println(asciiCode + " " + count);

}

}

}

}

}

}

# 10008 - What's Cryptanalysis? 什麼是密碼分析？

Cryptanalysis involves breaking down cryptographic writing. This process typically involves performing statistical analysis on the ciphertext. Your task is to write a program that performs a simple analysis of the ciphertext.

密碼翻譯（cryptanalysis）是指把某個人寫的密文（cryptographic writing）加以分解。這個程序通常會對密文訊息做統計分析。你的任務就是寫一個程式來對密文作簡單的分析。

Input Description 輸入說明

The first column of the input contains a positive integer n, which represents the number of columns of ciphertext that need to be analyzed. The next n columns each contain 0 or more characters (possibly including whitespace characters).

輸入的第1列有一個正整數n，代表以下有多少列需要作分析的密文。 接下來的n列，每列含有0或多個字元（可能包含空白字元）

Output Description 輸出說明

Each column contains an uppercase character (A-Z) and a positive integer. This positive integer represents the number of times the character appears in the input. Uppercase and lowercase characters (e.g., A and a) are considered identical characters in the input. When outputting, arrange the characters in descending order of frequency. If two or more characters have the same frequency, arrange them in ascending order of case (e.g., A before H). Please note: If a character does not appear in the input, it should not appear in the output.

每列包含一個大寫字元（A~Z）和一個正整數。這個正整數代表該字元在輸入中出現的次數。輸入中大小寫（例如：A及a）視為相同的字元。輸出時請按照字元出現的次數由大到小排列，如果有2個以上的字元出現次數相同的話，則按照字元的大小（例如：A在H之前）由小到大排列。 請注意：如果某一字元未出現在輸入中，那他也不應出現在輸出中。

Example Input 輸入範例

3

This is a test.

Count me 1 2 3 4 5.

Wow!!!! Is this question easy?

Example Output輸出範例

S 7

T 6

I 5

E 4

O 3

A 2

H 2

N 2

U 2

W 2

C 1

M 1

Q 1

Y 1

解體概要：先讀取一行字串，在做字母頻率的紀錄。字母不分大小寫，輸出一律以大寫為主。

/\* UVa 10008 - What's Cryptanalysis?

\* by C111193227林佩璇,

\* 2025/11/12

\*/

**import** java**.**util**.**Scanner**;**

class Main**{**

public static void main**(**String args**[]){**

Scanner sc**=new** Scanner**(**System**.**in**);**

//處理有多少case

String temp**=**sc**.**nextLine**();**

int N**=**0**;**

**for(**int i**=**0**;**i**<**temp**.**length**();**i**++){**

**if(**temp**.**charAt**(**i**)>=**48 **&&** temp**.**charAt**(**i**)<=**56**)** N**=**N**\***10**+**temp**.**charAt**(**i**)-**48**;**

**}**

//初始化陣列

int f**[]=new** int**[**26**];**

**for(**int i**=**0**;**i**<**26**;**i**++){**

f**[**i**]=**0**;**

**}**

//開始記錄字母出現的頻率

**for(**int i**=**0**;**i**<**N**;**i**++){**

String st**=**sc**.**nextLine**();**

**for(**int j**=**0**;**j**<**st**.**length**();**j**++){**

**if(**st**.**charAt**(**j**)>=**65 **&&** st**.**charAt**(**j**)<=**90**)** f**[**st**.**charAt**(**j**)-**65**]++;**

**else** **if(**st**.**charAt**(**j**)>=**97 **&&** st**.**charAt**(**j**)<=**122**)** f**[**st**.**charAt**(**j**)-**97**]++;**

**}**

**}**

//找出最大值

int max**=**0**;**

**for(**int i**=**0**;**i**<**26**;**i**++){**

**if(**f**[**i**]>**max**)** max**=**f**[**i**];**

**}**

//印出資料

**for(**int i**=**max**;**i**>**0**;**i**--){**

**for(**int j**=**0**;**j**<**26**;**j**++){**

**if(**f**[**j**]==**i**)** System**.**out**.**println**((**char**)(**j**+**65**)+**" "**+**i**);**

**}**

**}**

**}**

**}**

# 10268- 498-bits 498 位元

Looking throw the “Online Judge’s Problem Set Archive” I found a very interesting problem number 498, titled “Polly the Polynomial”. Frankly speaking, I did not solve it, but I derived from it this problem.

Everything in this problem is a derivative of something from 498. In particular, 498 was “... designed to help you remember ... basic algebra skills, make the world a better place, etc., etc.”. This problem is designed to help you remember basic derivation algebra skills, increase the speed in which world becomes a better place, etc., etc.

In 498 you had to evaluate the values of polynomial

a0xn +a1xn−1 +...+an−1x+an.

In this problem you should evaluate its derivative. Remember that derivative is defined as a0nxn−1 +a1(n−1)xn−2 +...+an−1.

All the input and output data will fit into integer, i.e. its absolute value will be less than 231.

Input Description 輸入說明

翻閱“線上法官問題集檔案”，我發現了一個非常有趣的問題，編號為 498，標題為“Polly the Polynomial”。坦白說，我沒有解決它，但是我從中衍生出了這個問題。

這個問題中的所有內容都是 498 中某些內容的衍生。特別是，498 是「…旨在幫助您記住…基本代數技能，讓世界變得更美好等等」。這個問題旨在幫助你記住基本的代數推導技能，提高世界變得更美好的速度等等。

在 498 中，你必須計算多項式的值

a0xn +a1xn−1 +...+an−1x+an。

在這個問題中，你應該評估它的導數。請記住，導數定義為

a0nxn−1 +a1(n−1)xn−2 +...+an−1。

所有輸入和輸出資料都將適合整數，即其絕對值將小於231。

Output Description 輸出說明

Your program should accept an even number of lines of text. Each pair of lines will represent one problem. The first line will contain one integer - a value for x. The second line will contain a list of integers a0, a1, ..., an−1, an, which represent a set of polynomial coefficients.

Input is terminated by <EOF>.

Example Input 輸入範例

您的程式應該接受偶數行文字。每一對線代表一個問題。第一行將包含一個整數 - x 的值。第二行將包含整數 a0、a1、...、an−1、an 的列表，它們表示一組多項式係數。

輸入以<EOF>終止。

Example Output輸出範例

For each pair of lines, your program should evaluate the derivative of polynomial for the given value x and output it in a single line.

對於每一對行，您的程式應計算給定值 x 的多項式的導數並將其輸出在一行中。

解題方向：直接依微分的方式打成程式碼。

/\* UVa 10268 - 498-bits

\* by C111193227林佩璇,

\* 2025/11/12

\*/

**import** java**.**util**.**Scanner**;**

class Main**{**

public static void main**(**String args**[]){**

Scanner sc**=new** Scanner**(**System**.**in**);**

**while(**sc**.**hasNextLine**()){**

long x**=**Long**.**parseLong**(**sc**.**nextLine**());**

String an**[]=**sc**.**nextLine**().**split**(**"\\s+"**);** //跳過多個空白鍵

long ans**=**0**;**

long xp**=**1**;**

**for(**int i**=**an**.**length**-**2**;**i**>=**0**;**i**--,**xp**\*=**x**){**

ans**+=**Long**.**parseLong**(**an**[**i**])\*(**an**.**length**-**1**-**i**)\***xp**;** //微分計算

**}**

//Output

System**.**out**.**println**(**ans**);**

**}**

**}**

**}**

# 12019-Doom’s Day Algorithm 世界末日演算法

No. Doom’s day algorithm is not a method to compute which day the world will end. It is an algorithm created by the mathematician John Horton Conway, to calculate which day of the week (Monday, Tuesday, etc.) corresponds to a certain date.

This algorithm is based in the idea of the doomsday, a certain day of the week which always occurs in the same dates. For example, 4/4 (the 4th of April), 6/6 (the 6th of June), 8/8 (the 8th of August), 10/10 (the 10th of October) and 12/12 (the 12th of December) are dates which always occur in doomsday. All years have their own doomsday.

In year 2011, doomsday is Monday. So all of 4/4, 6/6, 8/8, 10/10 and 12/12 are Mondays. Using that information, you can easily compute any other date. For example, the 13th of December 2011 will be Tuesday, the 14th of December 2011 will be Wednesday, etc.

Other days which occur on doomsday are 5/9, 9/5, 7/11 and 11/7. Also, in leap years, we have the following doomsdays: 1/11 (the 11th of January) and 2/22 (the 22nd of Febrary), and in non-leap years 1/10 and 2/21.

Given a date of year 2011, you have to compute which day of the week it occurs.

不。末日演算法不是計算世界末日的方法。它是由數學家 John Horton Conway 創建的演算法，用於計算一周中的哪一天（星期一、星期二等）對應於某個日期。

該演算法基於世界末日的思想，即一周中的某一天總是發生在相同的日期。例如，4/4（4 月4 日）、6/6（6 月6 日）、8/8（8 月8 日）、10/10（10 月10 日）和12/12（12 月12 日）十二月）是總是發生在世界末日的日期。所有的歲月都有自己的末日。

Input Description 輸入說明

2011年，世界末日是星期一。所以 4/4、6/6、8/8、10/10 和 12/12 都是星期一。使用該信息，您可以輕鬆計算任何其他日期。例如，2011年12月13日為星期二，2011年12月14日為星期三等。

其他發生在世界末日的日子有 5/9、9/5、7/11 和 11/7。此外，在閏年，我們有以下世界末日：1/11（1 月 11 日）和 2/22（2 月 22 日），非閏年有 1/10 和 2/21。

給定 2011 年的日期，您必須計算它發生在一周中的哪一天。

Output Description 輸出說明

The input can contain different test cases. The first line of the input indicates the number of test cases. For each test case, there is a line with two numbers: M D. M represents the month (from 1 to 12)

and D represents the day (from 1 to 31). The date will always be valid

輸入可以包含不同的測試案例。輸入的第一行表示測試案例的數量。對於每個測試案例，有一行包含兩個數字：M D。M 代表月份（從 1 到 12）

D 代表日期（從 1 到 31）。日期永遠有效

Example Input 輸入範例

For each test case, you have to output the day of the week where that date occurs in 2011. The days of the week will be: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.

對於每個測試案例，您必須輸出該日期在 2011 年發生的星期幾。星期幾將為：星期一、星期二、星期三、星期四、星期五、星期六、星期日。

Example Output輸出範例

8

16

2 28 45

5 26 81 11 1 12 25 12 31

Thursday

Monday

Tuesday

Thursday

Monday

Tuesday

Sunday

Saturday

解題方向：找出1/1~所要查詢的日期的總天數，以7取餘數後查表即可知道答案。

/\* UVa 12019 - Doom's Day Algorithm

\* by C111193227林佩璇,

\* 2025/11/12

\*/

**import** java**.**util**.**Scanner**;**

public class Main **{**

public static void main**(**String**[]** args**)** **{**

Scanner scanner **=** **new** Scanner**(**System**.**in**);**

// 宣告 1 ~ 12 月, 每一個月的天數

int months**[]** **=** **{**31**,** 28**,** 31**,** 30**,** 31**,** 30**,** 31**,** 31**,** 30**,** 31**,** 30**,** 31**};**

// 宣告星期一 ~ 星期天的英文單字

String weeks**[]** **=** **{**"Sunday"**,** "Monday"**,** "Tuesday"**,** "Wednesday"**,** "Thursday"**,** "Friday"**,** "Saturday"**};**

// 索引位置

// 0 1 2 3 4 5 6

// 取得資料筆數

int cases **=** scanner**.**nextInt**();**

// 使用 for 迴圈, 產生 0 ~ 小於資料筆數的索引, 根據資料筆數取得對應的月與日期

**for** **(**int i **=** 0**;** i **<** cases**;** i**++)** **{**

int month **=** scanner**.**nextInt**();** // 月份

int day **=** scanner**.**nextInt**();** // 日期

// 記錄 1/1 到測試日期總共經過幾天, 後續可使用總天數來推測測試日期為星期幾

int totalDay **=** 0**;**

// 使用 for 迴圈產生 1月~測試月份上一個月的索引位置

// 計算從 1/1(包含)到測試日期前一天

// 假設測試資料為 4 月 16 號

// for 會產生 1月~3月的上一個月的索引 0 ~ 2

// 總天數會等於 31(1月的天數)+28(2月的天數)+31(3月的天數) = 90 天

**for** **(**int m **=** 0**;** m **<** month **-** 1**;** m**++)** **{**

// 總天數 = 目前的總天數 + 讀取天數陣列中的特定索引位置

totalDay **=** totalDay **+** months**[**m**];**

**}**

// 總天數 = 目前的總天數 + 測試資料的日期

// 例如 4/16 在 for 迴圈沒有累加 4 月的天數, 所以要另外將 16 累加到總天數中

totalDay **=** totalDay **+** day**;**

// 總天數 = 目前的總天數 + 調整日期, 才能讀取到正確的星期

// 由題目可知 1/6 為星期四, 可推得 1/1 是星期六

// 而 1/1 ~ 1/6, 總天數為 6 天, 除以 7 取餘數=6

// 所以當讀取星期陣列的索引位置 6 時, 取得的值為星期六

// 但題目說 1/6 是星期四, 所以要統一將天數加 5 才能得到禮拜四

totalDay **=** totalDay **+** 5**;**

// 將總天數除以 7(一個禮拜有 7 天)取餘數, 可得數字 0 ~ 6, 可對應到星期陣列的索引位置

int weekIndex **=** totalDay **%** 7**;**

// 輸出計算結果, 讀取星期陣列的特定索引位置

System**.**out**.**println**(**weeks**[**weekIndex**]);**

**}**

**}**

**}**

# 10226- 硬木樹種

在第一幕中，萊波雷洛向唐娜·埃爾維拉講述了他主人的一長串征服：

「這是我師父所喜愛的美女名單，是我自己列的，你看一下，跟我一起讀一下。義大利有六百四十個，德國有兩百三十一個，法國有一百個，土耳其有九十一個；但在西班牙已經一千零三了！其中有鄉村姑娘、侍女、都市麗人；有伯爵夫人、男爵夫人、侯爵夫人、公主：各個等級、各個體型、各個年齡段的女性。”

（Madamina，il Catalogo questo）由於萊波雷洛按照時間順序記錄了唐璜“愛過”的所有“美女”，因此他向別人展示他主人的征服是非常麻煩的，因為他需要根據她們的數量來數「美女」的數量。每次都國籍。你要幫助萊波雷洛數數。

在第一幕中，Leporello 正在告訴 Donna Elvira 他主人的長長的征服清單：

“這是我主人喜歡的美女清單，我自己做的清單：看看，和我一起讀一讀。 在義大利六十四十，在德國二百三十一，在法國一百，在土耳其九十一；但在西班牙已經一千三！ 其中有鄉下女孩、侍女、城市美女；有伯爵夫人、男爵夫人、女侯爵夫人、公主：各種級別、各種規模、各種年齡的女性。” （Madamina，il catalogo questo）當 Leporello 按時間順序記錄所有 Don Giovanni“喜愛的”“美女”時，他向他人展示主人的征服是非常麻煩的，因為他每次都需要按國籍計算“美女”的數量。 你要幫助 Leporello 數數。 Input 輸入

輸入最多包含 2000 行。第一行包含數字n，表示還有n行。接下來的每一行最多 75 個字符，包含一個國家（第一個單字）和一個女人的名字（該行的其餘單字）

喬瓦尼很喜歡。您可能會假設所有國家/地區的名稱僅由一個單字組成。

輸出輸出

輸入最多由 2000 行組成。 第一行包含一個數字 n，表示還會有 n 行。 以下每一行最多 75 個字元，包含一個國家（第一個單詞）和一個喬瓦尼所愛的女人的名字（該行其餘單詞）。 您可以假設所有國家的名稱僅由一個單詞組成。

Sample Input 輸入範例

C111193227,ID: 1623323

The output consists of lines in alphabetical order. Each line starts with the name of a country, followed by the total number of women Giovanni loved in that country, separated by a space.

輸出按字母順序排列。 每一行都以一個國家的名字開頭，然後是喬瓦尼在該國愛過的女性總數，由一個空格隔開。

3

Spain Donna Elvira

England Jane Doe

Spain Donna Anna

Sample Output 輸出範例

England 1 Spain 2

解法：

1. 存放 2 份資料，一份使用 Set 存放(因為 Set 不會存放相同的 Object)，一份用一般 array 存放。
2. 存放 2 份資料，一份使用 Set 存放(因為 Set 不會存放相同的 Object)，一份用一般 array 存放。

/\* UVa

* by C111193227 林佩璇,
* 2023//

\*/

導入java.util.Scanner ；導入java.util.Arrays；導入java.util.Set；

導入java.util.HashSet ;

類別主要{

public static void main( String args[]){ 掃描器 sc=new Scanner(System.in); int 案例=sc.nextInt();

字串國家[ ] =新字串[案例];

Set<String> set=new HashSet<String> ( );

for(int i=0;i<cases;i++){ country[i]=sc.next(); //存放國家名稱

set.add(country[i]); //使用 set 在存放一份國家名稱。Set 特性：不會存放相同的 Object。

String No=sc.nextLine(); //因為姓名沒有用到，所以不重要。

}

//由於如果國家數目相同會需要比較國家名稱，故先把國家名稱進行排列。

Arrays.sort(country);

String st[]=set.toArray(new String[set.size()]);

Arrays.sort(st);

//拿剛剛 set 存放的國家名稱去跟剛剛另一份 array 的資料做比較，計算有多少個相同的國家名稱。

for(int i=0;i<st.length;i++){

int count=0;

for(int j=0;j<cases;j++){

if(st[i].equals(country[j])) count++;

}

//輸出

System.out.println(st[i]+" "+count);

}

}

}

C 111193227，ID ：1623323

# 10226-硬木樹種

硬木是具有寬闊葉子、結出果實或堅果、通常在冬季休眠的植物群。

America’s temperate climates produce forests with hundreds of hardwood species — trees that share certain biological characteristics. Although oak, maple and cherry all are types of hardwood trees, for example, they are different species. Together, all the hardwood species represent 40 percent of the trees in the United States. On the other hand, softwoods, or conifers, from the Latin word meaning “conebearing”, have needles. Widely available US softwoods include cedar, fir, hemlock, pine, redwood, spruce and cypress. In a home, the softwoods are used primarily as structural lumber such as 2×4s and 2×6s, with some limited decorative applications. Using satellite imaging technology, the Department of Natural Resources has compiled an inventory of every tree standing on a particular day. You are to compute the total fraction of the tree population represented by each species. 硬木是具有寬闊葉子、結出果實或堅果、通常在冬季休眠的植物群。美國的溫帶氣候孕育出擁有數百種闊葉樹種的森林，這些闊葉樹種具有某些共同的生物學特徵。 例如，雖然橡樹、楓樹和櫻桃樹都是硬木樹種，但它們是不同的物種。 所有硬木樹種加起來占美國樹木的 40%。

另一方面，軟木或針葉樹，源自拉丁語，意思是“圓錐軸承”，有針。 廣泛使用的美國軟木包括雪松、冷杉、鐵杉、松樹、紅木、雲杉和柏樹。 在家庭中，軟木主要用作結構木材，例如 2×4 和 2×6，以及一些有限的裝飾應用。利用衛星成像技術，自然資源部編制了特定日期每棵樹的清單。 您要計算每個物種所代表的樹木族群的總比例。

Input 輸入

The first line is the number of test cases, followed by a blank line.

Each test case of your program consists of a list of the species of every tree observed by the satellite; one tree per line. No species name exceeds 30 characters. There are no more than 10,000 species and no more than 1,000,000 trees.

There is a blank line between each consecutive test cases. 第一行是測試案例的數量，後面是一個空白行。

程式的每個測試案例都包含衛星觀測到的每棵樹的物種清單；每行一棵樹。 物種名稱不超過 30 個字元。 物種不超過 10,000 個，樹木不超過

1,000,000 棵。

每個連續的測試案例之間都有一個空白行

Output 輸出

For each test case print the name of each species represented in the population, in alphabetical order, followed by the percentage of the population it represents, to 4 decimal places.

Print a blank line between 2 consecutive test cases.

對於每個測試案例，按字母順序列印種群中代表的每個物種的名稱，然後是它代表的種群的百分比，精確到小數點後 4 位。在兩個連續的測試案例之間列印一個空白行。

輸入範例輸入範例

1

紅榿木

灰

阿斯彭

椴木

灰

山毛櫸

黃樺木

灰

櫻桃

三葉楊

灰

柏

紅榆樹

膠

樸樹

白橡木

山胡桃木

胡桃

硬楓木

白橡木

軟楓木

紅橡木

紅橡木

白橡木

波普蘭

檫樹

梧桐黑胡桃木

柳

範例輸出輸出範例

灰 13.7931

阿斯彭 3.4483

椴木 3.4483

山毛櫸 3.4483

黑胡桃木 3.4483

櫻桃3.4483

三葉楊 3.4483

柏樹 3.4483

口香糖 3.4483

樸樹3.4483

硬楓木 3.4483

山核桃木 3.4483

山核桃 3.4483

波普蘭 3.4483

紅榿木 3.4483

紅榆3.4483

Red Oak 6.8966

Sassafras 3.4483

Soft Maple 3.4483

Sycamore 3.4483

White Oak 10.3448

Willow 3.4483

Yellow Birch 3.4483 解法：

1. 使用 TreeMap 去做資料存放。
2. 依序算出每個物種所占多少百分比。

# 299-Train Swapping

At an old railway station, you may still encounter one of the last remaining “train swappers”. A train swapper is an employee of the railroad, whose sole job it is to rearrange the carriages of trains.

在舊火車站，您可能仍然會遇到僅存的“火車交換者”之一。 火車調換員是鐵路的僱員，其唯一的工作就是重新安排火車車廂。

一旦車廂按照最佳順序排列，火車司機所要做的就是將車廂一節一節地放到要裝載貨物的車站。

Input 輸入

Once the carriages are arranged in the optimal order, all the train driver has to do, is drop the carriages off, one by one, at the stations for which the load is meant. The title “train swapper” stems from the first person who performed this task, at a station close to a railway bridge. Instead of opening up vertically, the bridge rotated around a pillar in the center of the river. After rotating the bridge 90 degrees, boats could pass left or right.

第一個火車改裝者發現這座橋最多只能在兩節車廂上行駛。透過將橋旋轉 180 度，車廂交換了位置，允許他重新排列車廂（作為副作用，車廂然後面向相反的方向，但火車車廂可以向任何方向移動，所以誰在乎呢）。

現在幾乎所有的火車交換器都已經消失，鐵路公司希望實現其營運自動化。待開發的程序的一部分是一個例程，它決定對於給定的列車，訂購列車所需的兩個相鄰車廂的最少交換次數。你的任務就是創建這個例程。

「火車交換者」這個稱號源自於第一個在靠近鐵路橋的車站執行這項任務的人。這座橋不是垂直開放的，而是圍繞著河中心的一根柱子旋轉。 橋旋轉 90 度後，船隻可以向左或向右通過。第一個火車改裝者發現這座橋最多只能在兩節車廂上行駛。 透過將橋旋轉 180 度，車廂交換了位置，允許他重新排列車廂（作為副作用，車廂然後面向相反的方向，但火車車廂可以向任何方向移動，所以誰在乎呢）。現在幾乎所有的火車交換器都已經消失，鐵路公司希望實現其營運自動化。 待開發的程序的一部分是一個例程，它決定對於給定的列車，訂購列車所需的兩個相鄰車廂的最少交換次數。 你的任務就是創建這個例程。

輸出輸出

對於每個測試案例輸出句子：“最佳列車交換需要 S 次交換。”其中 S 是整數。

輸入的第一行包含測試案例的數量 (N)。 每個測試案例由兩條輸入線組成。 測試案例的第一行包含一個整數 L，決定列車的長度 (0 ≤ L ≤ 50)。 測試案例的第二行包含數字 1 到 L 的排列，指示車廂的當前順序。 托架的排序應使托架 1 首先出現，然後是托架 2，依此類推，托架 L 最後出現。

Sample Input 輸入範例

3

3

132 4 4321 2

21

Sample Output 輸出範例

Optimal train swapping takes 1 swaps.

Optimal train swapping takes 6 swaps.

Optimal train swapping takes 1 swaps.

解法：

# 10222- 破解狂人

到了BUET，一位老教授徹底瘋了。他開始用一些奇怪的字眼說話。沒有人能聽懂他的演講和講座。最後BUET當局陷入了大麻煩。沒有辦法讓那個人繼續在大學工作。突然，一名學生（他肯定是 UVA ACM 分會的註冊作者，並且在 24小時線上評判中擁有良好的排名）創建了一個能夠解碼該教授演講的程式。他的發明之後，大家又得到了安慰，老老師又像以前一樣開始了日常工作。

因此，如果您訪問 BUET 並看到老師用麥克風說話，麥克風連接到配備語音識別軟體的 IBM 計算機，並且學生正在計算機屏幕上聽課，請不要感到震驚！因為現在你的工作就是寫同樣的程式來解碼那個瘋狂老師的演講！

到了 BUET，一位老教授徹底瘋了。 他開始用一些奇怪的字眼說話。 沒有人能聽懂他的演講和講座。 最後 BUET 當局陷入了大麻煩。 沒有辦法讓那個人繼續在大學工作。 突然，一名學生（他肯定是 UVA ACM 分會的註冊作者，並且在 24 小時線上評審中擁有良好的排名）創建了一個能夠解碼該教授演講的程式。 他的發明之後，大家又得到了安慰，老老師又像以前一樣開始了日常工作。 因此，如果您訪問 BUET 並看到老師用麥克風講話，該麥克風連接到配備語音識別軟體的 IBM 計算機，並且學生正在計算機屏幕上聽課，請不要感到震驚！ 因為

現在你的工作就是寫同樣的程式來解碼那個瘋狂老師的演講！

Input 輸入

The input file will contain only one test case i.e. the encoded message. The test case consists of one or more words.

輸入檔案將僅包含一個測試案例，即編碼訊息。 測試案例由一個或多個單字組成

Output 輸出

For the given test case, print a line containing the decoded words. However, it is not so hard task to replace each letter or punctuation symbol by the two immediately to its left alphabet on your standard keyboard.

對於給定的測試案例，列印包含解碼後的單字的行。 然而，在標準鍵盤上用緊鄰其左側的兩個字母替換每個字母或標點符號並不是一件困難的任務。

Sample Input 輸入範例

k[r dyt I[o

Sample Output 輸出範例

how are you 解法：

1. 建立一個鍵盤的字串方便查詢。
2. 將輸入的字依序比較 step1 所建立的表進行比較，當找到相同字元時，輸出前第 2 個位子的字元。 Ex 表:qwer 當輸入 r 時，對照表並輸出前第 2 個位子的字元 w。

# 11332-總和數字求和

For a positive integer n, let f(n) denote the sum of the digits of n when represented in base 10. It is easy to see that the sequence of numbers n, f (n), f (f (n)), f (f (f (n))), . . . eventually becomes a single digit number that repeats forever. Let this sin- gle digit be denoted g(n).

For example, consider n = 1234567892. Then: f (n) = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 2 = 47 f(f(n)) = 4 + 7 = 11 f(f(f(n))) = 1 + 1 = 2

Therefore, g(1234567892) = 2.

對於正整數 n，設 f(n) 表示以 10 為基數表示的 n 的各位數字之和。很容易看

出數字序列 n, f (n), f (f (n))， f (f (f (n))), . 。 。 最終變成一個永遠重複的個位數。 讓這個單一數字表示為 g(n)。

例如，考慮 n = 1234567892。然後：

f(n) = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 2 = 47 f(f(n)) = 4 + 7 = 11 f(f(f(n))) = 1 + 1 = 2 因此，g(1234567892) = 2。

Input 輸入

For a positive integer n, let f(n) denote the sum of the digits of n when represented in base 10. It is easy to see that the sequence of numbers n, f (n), f (f (n)), f (f (f (n))), . . . eventually becomes a single digit number that repeats forever. Let this sin- gle digit be denoted g(n).

For example, consider n = 1234567892. Then:

f (n) = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 2 = 47 f(f(n)) = 4 + 7 = 11 f(f(f(n))) = 1 + 1 = 2

Therefore, g(1234567892) = 2.

對於正整數 n，設 f(n) 表示以 10 為基數表示的 n 的各位數字之和。很容易看

出數字序列 n, f (n), f (f (n))， f (f (f (n))), . 。 。 最終變成一個永遠重複的個位數。 讓這個單一數字表示為 g(n)。

例如，考慮 n = 1234567892。然後：

f(n) = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 2 = 47 f(f(n)) = 4 + 7 = 11 f(f(f(n))) = 1 + 1 = 2 因此，g(1234567892) = 2。

Output 輸出

For each such integer, you are to output a single line containing g(n). 對於每個這樣的整數，您將輸出包含 g(n) 的一行。

Sample Input 輸入範例

2 11 47 1234567892 0

Sample Output 輸出範例

2 2 2 2

解法：用字串讀取輸入，在一個一個字元相加。

# 272-TeX 報價 TeX報價

TEX 是 Donald Knuth 開發的排版語言。它需要源文本和一些

排版說明並產生（人們希望）一份漂亮的文件。漂亮的文檔使用“

和 」來分隔引號，而不是大多數鍵盤提供的普通的 " 。

鍵盤通常沒有定向雙引號，但有左單引號和右單引號 '。現在檢查鍵盤找到左單引號鍵（有時稱為“反引號鍵”）和右單引號鍵 '（有時稱為“反引號鍵”）

“撇號”或只是“引用”）。小心不要將左單引號 ` 與「反斜線」鍵混淆

\. TEX 允許使用者鍵入兩個左單引號 `` 以建立左雙引號「 和兩個右單引號 '' 以建立右雙引號 」。然而，大多數打字員習慣用無方向的雙引號 " 來界定他們的引文。

TEX 是 Donald Knuth 開發的排版語言。 它需要源文本和一些

排版說明並產生（人們希望）一份漂亮的文件。 漂亮的文檔使用“ 和 ” 來分隔引號，而不是大多數鍵盤提供的普通的 " 。

鍵盤通常沒有定向雙引號，但有左單引號“和右單引號”。 現在檢查您的鍵盤以找到左單引號鍵（有時 稱為“反引號鍵”）和右單引號鍵 ' （有時稱為“撇號”或 只是“引用”）。 小心不要將左單引號 與「反斜線」鍵 \ 混淆。 TEX 讓

使用者輸入兩個左單引號``以建立左雙引號「和兩個右單引號'' 建立右雙引號」。 然而，大多數打字員習慣對引文進行界定帶有無方向的雙引號“。

If the source contained

"To be or not to be," quoth the bard, "that is the question." then the typeset document produced by TEX would not contain the desired form:

“To be or not to be,” quoth the bard, “that is the question.”

In order to produce the desired form, the source file must contain the sequence:

`To be or not to be,'' quoth the bard,` that is the question.''

您將編寫一個程序，將包含雙引號 (") 字元的文字轉換為文字

除了雙引號已被所需的兩個字元序列替換之外，其他都是相同的

由 TEX 使用定向雙引號來分隔引號。雙引號 (") 字元應該

適當地替換為 `如果 " 開啟報價，如果 " 關閉報價則替換為 ''。請注意，不會出現嵌套引號的問題：第一個 " 必須替換為`，下一個替換為 ''，下一個替換為 `，下一個替換為 ''，下一個替換為` ，下一個替換為''，依此類推在。

如果來源包含

“生存還是毀滅，”吟遊詩人說道，“這就是問題所在。” 那麼 TEX 產生的排版文件將不包含所需的形式： “生存還是毀滅，”吟遊詩人說道，“這就是問題所在。” 為了產生所需的形式，原始檔案必須包含以下序列：

“生存還是毀滅，”吟遊詩人說道，“這就是問題所在。”

您將編寫一個程序，將包含雙引號 (") 字元的文字轉換為文字除了雙引號已被所需的兩個字元序列替換之外，其他都是相同的由 TEX 使用定向雙引號來分隔引號。 雙引號 (") 字元應該適當地替換為 `如果 " 開啟報價，如果 " 關閉報價則替換為 ''。 請注意，不會出現嵌套引用的問題：第一個“必須替換為`，下一個按 ''，下一個按 `，下一個按 ''，下一個按`，下一個按 ''，依此類推。

Input 輸入

Input will consist of several lines of text containing an even number of double-quote (") characters.

Input is ended with an end-of-file character.

輸入將由多行文字組成，其中包含偶數個雙引號 (") 字元。

輸入以檔案結束符結束。

Output 輸出

The text must be output exactly as it was input except that:

* the first " in each pair is replaced by two ` characters: `` and
* the second " in each pair is replaced by two ' characters: ''.

文字的輸出必須與輸入時完全相同，但以下情況除外：

* 每對中的第一個“ 替換為兩個` 字元：`` 和
* 每對中的第二個“ 替換為兩個' 字元：''。

Sample Input 輸入範例

The text must be output exactly as it was input except that: • the first " in each pair is replaced by two ` characters: `` and

* the second " in each pair is replaced by two ' characters: ''.

文字的輸出必須與輸入時完全相同，但以下情況除外：

* 每對中的第一個“ 替換為兩個` 字元：`` 和
* 每對中的第二個“ 替換為兩個' 字元：''。

Sample Output 輸出範例

`To be or not to be,'' quoth the Bard,` that is the question''.

The programming contestant replied: ``I must disagree.

To `C' or not to` C', that is The Question!''

「生存還是毀滅，」吟遊詩人說道，「是問題''。

程式設計選手回答說：“我一定不同意。” 去‘C’還是不去‘C’，這就是問題！”

解法：用一個 count 計數，當餘數為 1 時(代表第一個雙引號)用 `` 取代，當餘數為 0 時(代表第二個雙引號)用 '' 取代，其餘字就正常輸出。

# 490-旋轉句子

在「旋轉句子」中，您需要將一系列輸入的句子順時針旋轉 90 度。所以

您的程式不會從左到右、從上到下顯示輸入句子，而是顯示

它們從上到下，從右到左。

輸入輸入

作為程式的輸入，您最多會收到 100 個句子，每個句子不超過 100

長字符。合法字符包括：換行符、空格、任何標點符號、數字和小寫字符

大小寫或大寫英文字母。 （注意：製表符不是合法字元。）

輸出輸出

程式的輸出應將最後一句垂直印在最左邊的列中；

the first sentence of the input would subsequently end up at the rightmost columnSample Input 輸入範例

Rene Decartes once said, "I think, therefore I am."

Sample Output 輸出範例

"R Ie n te h iD ne kc ,a r tt he es r eo fn oc re e s Ia i ad m, . "

解法： 存入字串，輸出時先輸出最後1行字的第1個字再來輸出倒數第2行的字....依此類推。

# 11349 - Symmetric Matrix

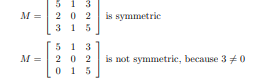
給定一個方陣 M。此矩陣的元素為 Mij ：{0 < i < n, 0 < j < n}。在這個

問題你必須找出給定的矩陣是否對稱。

定義：對稱矩陣是所有元素都非負且對稱的矩陣

與該矩陣的中心有關。任何其他矩陣都被認為是不對稱的。為了

例子：



您所要做的就是確定矩陣是否對稱。輸入中給出的矩陣元素為 −2 32 ≤ Mij ≤ 2 32 且 0 < n ≤ 100。

輸入

輸入的第一行包含測試案例的數量 T ≤ 300。然後 T 個測試案例按照以下方式進行描述。每個測試案例的第一行包含 n – 方陣的維度。然後n行後面的每一行都包含第i行。 Row 恰好包含 n 個由空格字元分隔的元素。第 i 行中的第 j 個數字是您必須處理的矩陣的元素 Mij 。

輸出

對於每個測試案例輸出一行「Test #t: S」。其中 t 是從 1 開始的測試數。如果矩陣是對稱的，則行 S 等於“對稱”，而在任何其他情況下，行 S 等於“非對稱”。

輸入範例

2 N = 3 5 1 3 2 0 2 3 1 5 N = 3 5 1 3 2 0 2 0 1 5

輸出範例

Test #1: Symmetric. Test #2: Non-symmetric.

解法： 把2維矩陣變成1維矩陣，判斷首尾數值是否相同

# 10252 - 常見排列

給定兩個小寫字母字串a和 b，列印最長的小寫字母字串 x，使得 x 的排列是 a 的子序列，而 x 的排列是 b 的子序列。

輸入輸入

輸入檔包含多行輸入。連續的兩行構成一組輸入。這意味著在輸入檔中，第 1 行和第 2 行是一組輸入，第 3 行和第 4 行是一組輸入，依此類推。一對的第一行包含a ，第二行包含 b。每個字串獨佔一行，最多包含 1000 個小寫字母

輸出輸出

對於每組輸入，輸出包含 x 的行。如果有幾個x滿足上述條件，則按字母順序選擇第一個

輸入範例輸入範例

走在街上的漂亮女人

範例輸出輸出範例

恩威特

解法：

1. 需讀一整行( sc.nextLine() or getline())，因為測資可能會有空白字串，如果不讀一行，則會以為結束符號反而去讀別行。

2. 如果字母為大寫，則不比較。

3. 使用2個陣列，存放2個字串所出現的字母頻率。

4. 找出2陣列頻率最小值，並印出字母。

# 10038 快樂跳線

如果連續元素之間的差的絕對值取所有值 1 到 n − 1，則 n > 0 整數的序列稱為歡樂跳。例如，1 4 2 3 是歡樂跳，因為絕對差分別為3、2和1。這個定義意味著任何單一整數的序列都是一個快樂的跳線。您將編寫一個程式來確定多個序列中的每一個是否都是快樂跳躍。

Input 輸入

Each line of input contains an integer n ≤ 3000 followed by n integers representing the sequence.

Output 輸出

For each line of input, generate a line of output saying ‘Jolly’ or ‘Not jolly’

Sample Input 輸入範例

4 1 4 2 3 5 1 4 2 -1 6

Sample Output 輸出範例

Jolly Not jolly

解法： 每行第一個數字為此行有多少數字(case)需要處理，其餘的為需要處理的數字。

1. 先判斷case是否為1，當case為1時則是Jolly。

2. 如case不是1，則取相鄰數字相減的絕對值存放到一個新的陣列裡，並且排序讓此陣列由小到大。

3. 用for迴圈比較相減的值是否有重複、是否有在1~(N-1)裡面，如果有在範圍裡並且無重複則為Jolly，反之為Not jolly。

# 10056 - 機率是多少？

機率一直是電腦演算法的一個組成部分。確定性的地方

演算法無法在短時間內解決問題，機率演算法已經出現了

救援。在這個問題中，我們不處理任何機率演算法。我們會盡力

決定某個玩家的獲勝機率。

遊戲是透過丟骰子之類的東西來玩的（不應該假設它有六個面，例如

一個普通的骰子）。如果玩家擲骰子時發生某個事件（例如得到 3、得到

綠色的一面在上面或其他什麼）他被宣佈為獲勝者。這樣的玩家可以有N個。所以第一個

玩家將擲骰子，然後是第二個玩家，最後是第 N 個玩家，最後是第一個玩家，

很快。當玩家獲得所需的事件時，他或她被宣佈為獲勝者並且比賽停止。你會

必須確定這些玩家中的一個（第 I 個）的獲勝機率

輸入輸入

輸入首先包含一個整數S（S≤1000），它表示有多少組輸入。

接下來的 S 行將包含 S 組輸入。每行包含一個整數 N (N ≤ 1000)，表示

玩家數量，浮點數 p，表示發生某事的機率

單次投擲成功事件（如果成功意味著得到 3 個，那麼 p 是在一次投擲中得到 3 個機率）

單次投擲。對於普通骰子，得到 3 的機率是 1/6)，I (I ≤ N) 是骰子的序列

待確定獲勝機率的玩家（序號從1到N不等）。你可以假設

不會給出無效的機率 (p) 值作為輸入。

輸出輸出

對於每組輸入，在單行中輸出第 I 個玩家獲勝的機率。輸出

floating point number will always have four digits after the decimal point as shown in the sample

output.

Sample Input 輸入範例

2 2 0.166666 1 2 0.166666 2

Sample Output 輸出範例

0.5455 0.4545

解法： 第一個輸入：有多少組測試資料。

第二個輸入：參加遊戲人數。

第三個輸入：獲勝的機率。

第四個輸入：第幾個人獲勝。

※假設獲勝機率：p、輸機率：q=1-p，每次遊戲共有R個回合、有n個人參與遊戲、第k個人獲勝。

# 10170 - 無限房間酒店

HaluaRuti 市有一家奇怪的飯店，有無限的房間。來這家飯店的團體

遵循以下規則：

a) 同時只有一個團體的成員可以租用飯店。

b) 每組入住當天上午入住，當晚離開飯店

退房日。

c) 前一組離開飯店後的第二天早上，另一組就會到來。

d) 傳入群組的一個非常重要的屬性是它比前一個群組多了一個成員

組，除非它是起始組。您將獲得起始組的成員人數。

e) 一個由 n 位成員組成的團體在飯店住宿 n 天。例如，如果一個由四名成員組成的小組

8月1日早上抵達，8月4日晚上離開飯店，

下一批五名成員將於 8 月 5 日上午抵達並停留五天

很快。

給定初始團體人數，您必須找到指定時間入住飯店的團體人數

天。

輸入輸入

輸入每行包含整數 S (1 ≤ S ≤ 10000) 和 D (1 ≤ D < 1015)。 S表示

團體的初始人數，D 表示您必須找到入住飯店的團體人數

第 D 天（從 1 開始）。所有輸入和輸出整數都將小於 1015 年。群組大小 S

意思是第一天有一群S成員來到飯店住了S天然後來

根據前面描述的規則，一組 S + 1 名成員，依此類推

輸出輸出

對於輸入的每一行，在單行上列印第 D 天入住飯店的團體人數

輸入範例輸入範例

1 6 3 10 3 14

範例輸出輸出範例

1 6 3 10 3 14

解法： 每行輸入的第一個數字為旅行團人數(S)、第二個數字為指定日期(D)。S+(S+1)+(S+2)+......+(S+N)<D，求出N即是答案。

# 11063 - B2-序列

B2 序列是正整數 1 ≤ b1 < b2 < b3 的序列。 。 。使得所有成對的和 bi + bj都不同，其中 i ≤ j。您的任務是確定給定序列是否為 B2 序列。

輸入輸入

每個測試案例都以 2 ≤ N ≤ 100（序列中的元素數量）開始。下一行將有 N 個整數，代表序列中每個元素的值。每個元素bi都是一個整數，使得bi≤10000。每個測試案例後面都有一個空行。輸入以檔案結尾 (EOF) 結束。

輸出輸出

對於每個測試案例，您必須列印測試案例的編號（從 1 開始），以及一條指示對應序列是否為 B2 序列的訊息。請參閱下面的範例輸出。在每個測試案例之後，您必須列印一個空白行。

輸入範例輸入範例

4 1 2 4 8 4 3 7 10 14

範例輸出輸出範例

情況 #1：這是一個 B2 序列。

Case #2: It is not a B2-Sequence.

解法： 直接for迴圈計算2數相加後的數字，並用個Vector記錄起來，在判斷是否有重複，當有重複就不是B2序列。

# 10101 - 孟加拉數字

孟加拉數字在擴展和轉換為文本時通常使用“kuti”(10000000)、“lakh”(100000)、“hajar”(1000)、“shata”(100)。您將編寫一個程序，用它們將給定的數字轉換為文字。

輸入輸入

輸入檔案可能包含多個測試案例。每個案例將包含一個非負數 ≤ 999999999999999。

輸出輸出

For each case of input, you have to output a line starting with the case number with four digits adjustment followed by the converted text.

Sample Input 輸入範例

23764 45897458973958

Sample Output 輸出範例

1. 23 hajar 7 shata 64 2. 45 lakh 89 hajar 7 shata 45 kuti 89 lakh 73 hajar 9 shata 58

解法： 使用遞迴取出kuti、lakh、hajar、shata的值。如果不使用遞迴，程式碼會很冗長。

# 10093 - 一個簡單的問題！

您聽過「每個普通數字系統的基數都是 10」這個事實嗎？當然，我不是說話

關於數位系統，例如 Stern Brockot 數位系統。這個問題與此無關

但可能有些相似之處。

您將獲得一個基於 N 的整數 R，並保證 R 是可整除的

通過 (N − 1)。您必須列印 N 的最小可能值。N 的範圍為 2 ≤ N ≤ 62

62 基數的數字符號是（ 0.. 9 和 A..Z 和 a..z）。同樣，數字符號

對於基於 61 的數字系統是0.. 9 和 A..Z 和 a..y) 等等。

輸入輸入

2.. 62)的整數（如數學中定義） 。您必須確定在給定條件下該數字的最小可能基數是多少。不會輸入無效數字。

Output 輸出

Each line in the input file will contain an integer (as defined in mathematics) number of any integer base (2..62). You will have to determine what is the smallest possible base of that number for the given conditions. No invalid number will be given as input.

Sample Input 輸入範例

3 5 A

Sample Output 輸出範例

4 6 11

解法： 以10進位為例，每個位數相加如果是9的倍數，那麼就可以被9整除。

# 10908 - 最大的正方形

給定一個矩形字元網格，您必須找出最大正方形的邊長，使得正方形的所有字元都相同，並且正方形的中心[兩條對角線的交點]位於位置（r， C）。網格的高度和寬度分別為M和N。網格的左上角和右下角分別以(0, 0)和(M − 1, N − 1)表示。考慮下面給出的字元網格。給定位置 (1, 2)，最大正方形的邊長為 3. abbbaaaaaa abbbaaaaaa abbbaaaaaa aaaaaaaaaa aaaaaaaaaa aaccaaaaaa aaccaaaaaa

輸入輸入

輸入以包含單一整數 T (< 21) 的行開始。接下來是 T 測試用例。它們的第一行將包含三個整數 M、N 和 Q (< 21)，並以空格分隔，其中 M、N 表示網格的維度。接下來是 M 行，每行包含 N 個字元。最後，將有 Q 行，每行包含兩個整數 r 和 c。 M 和 N 的值最多為 100。

輸出輸出

對於輸入中的每個測試案例，都會產生 Q + 1 行輸出。在第一行中依序列印 M、N 和 Q 的值，並以單一空格分隔。在接下來的 Q 行中，輸出輸入中每個 (r, c) 對的對應網格中最大正方形的邊長。

輸入範例輸入範例

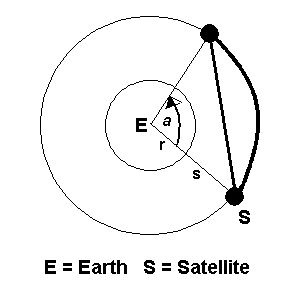
1 7 10 4 abbbaaaaaa abbbaaaaaa abbbaaaaaa aaaaaaaaaa aaaaaaaaaa aaccaaaaaa aaccaaaaaa 1 2 2 4 4 6 5 2

Sample Output 輸出範例

7 10 4 3 1 5 1

解法： 目標字元向外擴散一一檢查是否可成為一個正方形。

# 10221 – 衛星



地球的半徑是6440公里。有許多衛星和小行星繞著地球運行。如果兩顆衛星與地心形成一個角度，你能算出它們之間的距離嗎？我們所說的距離是指弧距和弦距。兩顆衛星都在同一軌道上（但是，請考慮它們是在圓形路徑而不是橢圓路徑上旋轉）。

輸入輸入

輸入檔案將包含一個或多個測試案例。每個測試案例由一行包含兩個整數 s 和 a 以及字串「min」或「deg」組成。這裡s是衛星與地球表面的距離，a是衛星與地心的夾角。它可以以分鐘 ( ′)或度 (°) 為單位。請記住，同一行永遠不會同時包含分和度。

輸出輸出

對於每個測試案例，列印一行包含所需距離，即兩顆衛星之間的弧距和弦距，以千米為單位。此距離將為小數點後六位的浮點數值。

輸入範例輸入範例

500 30 度 700 60 分鐘 200 45 度

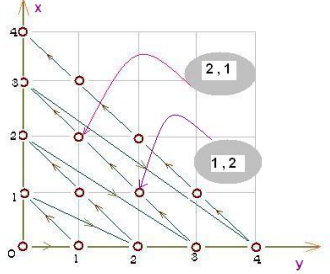
Sample Output 輸出範例

3633.775503 3592.408346 124.616509 124.614927 5215.043805 5082.035982

解法： 1.兩顆衛星的直線距離：直用正弦定理或餘弦定理去解即可。

2.兩顆衛星的弧長：直接用弧長公式去解即可。

# 10642 你能解決嗎？



首先來看一下下面的圖。在這張圖中，每個圓都有一個根據笛卡爾座標系的座標。您可以沿著向前箭頭符號指示的路徑從一個圓圈移動到另一個圓圈。從源圓到目的圓，所需的總步數=經過的中間圓數+1 例如，從(0, 3)到(3, 0)需要經過兩步中間圓(1, 2) 和(2, 1 )。因此，在本例中，所需的總步數為 2 + 1 = 3。在此問題中，您將計算給定源圓和目標圓所需的步數。您可以假設，不可能使用箭頭的相反方向返回。

輸入輸入

輸入中的第一行是要處理的測試案例數 n (0 < n ≤ 500)。接下來有n行，每行包含四個整數（0≤每個整數≤100000），第一對代表源圓的座標，另一對代表目標圓的座標。座標以 (x, y) 形式列出。

輸出輸出

對於每對整數，您的程式應先輸出案例編號，然後輸出從來源到達目的地的步驟數。您可能會假設從來源圓始終可以到達目標圓。

輸入範例輸入範例

3 0 0 0 1 0 0 1 0 0 0 0 2

範例輸出輸出範例

情況 1：1 情況 2：2 情況 3：3

解法：先計算出(0,0)到第一個點的距離(s1)，在計算(0,0)到第二點的距離(s2)。s2-s1即是答案。

# 10242 - Fourth Point !!

Given are the (x, y) coordinates of the endpoints of two adjacent sides of a parallelogram. Find the

(x, y) coordinates of the fourth point.

Input 輸入

每行輸入包含八個浮點數：端點之一的 (x, y) 座標

第一條邊的 (x, y) 座標，然後是第一條邊的另一個端點的 (x, y) 座標，然後是

第二邊的端點之一的 (x, y) 座標，後面跟著第二條邊的 (x, y) 座標

第二邊的另一個端點。所有座標均以米為單位，精確到毫米。所有座標

介於 −10000 和 +10000 之間。輸入由文件末尾終止。

輸出輸出

對於輸入的每一行，列印平行四邊形第四個點的 (x, y) 座標（以公尺為單位），

精確到毫米，用一個空格分隔。

輸入範例輸入範例

0.000 0.000 0.000 1.000 0.000 1.000 1.000 1.000

1.000 0.000 3.500 3.500 3.500 3.500 0.000 1.000

1.866 0.000 3.127 3.543 3.127 3.543 1.412 3.145

Sample Output 輸出範例

1.000 0.000

-2.500 -2.500

0.151 -0.398

解法： 利用對角線相加的原理下去解即可

# 10189 – Minesweeper

你玩過掃雷嗎？這是一款可愛的小遊戲，具有一定的操作性

系統的名字我們實在記不清了。好吧，遊戲的目標是找到所有的

M × N 區域內的地雷。為了幫助您，遊戲在方塊中顯示了一個數字，告訴您如何

那個廣場附近有很多地雷。例如，假設以下 4 × 4 欄位具有 2

地雷（以“\*”字元表示）：

\*...

....

.\*..

....

如果我們代表放置上述提示數字的相同字段，我們最終會

和：

\*100

2210

1\*10

1110

您可能已經注意到，每個方格最多可以有 8 個相鄰的方格。

輸入輸入

輸入將包含任意數量的欄位。每個字段的第一行包含兩個整數

n 和 m (0 < n, m ≤ 100) 分別代表欄位的行數和列數。

接下來的 n 行恰好包含m 個字元並代表該欄位。

每個安全方塊均由「.」表示。字元（不含引號）和每個地雷方塊

由“\*”字元表示（也不帶引號）。第一條場線，其中 n = m = 0

代表輸入結束，不應被處理。

輸出輸出

對於每個字段，您必須單獨在一行中列印以下訊息：

字段#x：

其中x代表欄位編號（從1開始）。接下來的 n 行應包含

帶有“.”的字段字元替換為該方格相鄰地雷的數量。必須有

欄位輸出之間的空白行。

輸入範例輸入範例

4 4

\*...

....

.\*..

....

3 5

\*\*...

……

.\*...

0 0

範例輸出輸出範例

字段#1：

\*100

2210

1\*10

1110

Field #2:

\*\*100

33200

1\*100

解法： 直接判斷此位子的周圍是否有地雷(包含斜線)，並輸出答案，如果此位子本身就是地雷，則直接輸出地雷。

# 10409 死亡遊戲

生活不容易。有時它超出了你的控制範圍。現在，作為參賽者

ACM ICPC 的你，可能剛剛嚐到了人生的苦澀。但別擔心！

不要只看到生命的陰暗面，也要看到生命的光明面。

人生可能是一場有趣的機會遊戲，就像擲骰子一樣。要嘛做，要嘛死！

然後，最終，你也許能夠找到勝利之路。

這個問題來自於使用骰子的遊戲。順便問一下，你知道嗎

一死？它與「死亡」無關。骰子是一個立方體物體，有六個

每個面代表從 1 到 6 的不同數字，並且是

並標示相應數量的點。因為平常會用到

在配對中，「骰子」是一個很少使用的字。不過，您可能聽過一句著名的短語「木已成舟」。

當遊戲開始時，骰子靜止在平坦的桌子上。遊戲過程中，骰子全部翻滾

經銷商的指示。如果您能預測頂面上看到的數字，您將贏得遊戲

骰子停止翻滾的時間。

現在要求您編寫一個程式來模擬骰子的滾動。為了簡單起見，我們

假設骰子既不滑動也不跳躍，只是在桌子上向四個方向（即北）滾動，

東、南、西。每局遊戲開始時，莊家將骰子放在骰子的中央

表格並調整其方向，使數字一、二、三分別位於頂部、北面和北面。

分別是西面。對於其他三個面，我們沒有明確指定任何內容，只是告訴您

黃金法則：任何一對相對面上的數字總和始終為七。

你的程式應該接受一系列命令，每個命令要么是“北”，要么是“東”，

“南”或“西”。 「向北」命令將骰子向北翻滾，也就是說，頂部變成

新的北方，北方成為新的底部，等等。更準確地說，模具圍繞

其北底邊朝北方向，旋轉角度為90度。其他命令還有

按照自己的方向相應地滾動骰子。你的程式最終應該會計算出這個數字

按順序執行命令後顯示在頂部。請注意，該表已足夠

大，骰子在遊戲過程中永遠不會掉落。

輸入輸入

輸入由一個或多個命令序列組成，每個命令序列對應一個遊戲。命令序列的第一行包含一個正整數，表示序列中後續命令列的數量。您可以假設該數字小於或等於 1024。包含零的行表示輸入結束。每個命令列包括「北」、「東」、「南」和「西」之一的命令。您可以假設任何行中都沒有出現空格。

輸出輸出

對於每個命令序列，輸出一行僅包含遊戲結束時頂面上的數字。

輸入範例輸入範例

1 北 3 東北 南 0

範例輸出輸出範例

5 1

解法： 1. 骰子正面位子1(由上往下看)、2(north)、3(west)。每兩相對應的面相加為 7。Ex 1+6=7，1 的正下方為 6 。

# 10415 - Eb 中音薩克斯風演奏家

你喜歡薩克斯風嗎？我有一把中音薩克斯風，如右圖。

播放音樂時我的手指會移動很多，而且我很

對每個手指按按鈕多少次感興趣。假設音樂僅由8種音符組成。他們是：

CDEFG A B 為八度，CDEFGAB 為更高一個八度

八度。我們用c,d ,e,f,g,a,b,C,D,E,F,G,A,B來表示它們。這

我用於每個音符的手指是：

• c: 指 2 ∼ 4, 7 ∼ 10

• d: 指2 ∼ 4, 7 ∼ 9

• e: 指 2 ∼ 4, 7, 8

• f: 指2 ∼ 4, 7

• g：指2 ∼ 4

• a：手指 2、3

• b：手指 2

• C：手指 3

• D：手指 1 ∼ 4, 7 ∼ 9

• E：手指 1 ∼ 4, 7, 8

• F：手指 1 ∼ 4, 7

• G：手指 1 ∼ 4

• A：手指 1 ∼ 3

• B：手指 1 ∼ 2

（注意每個手指都控制一個指定的按鈕，不同的

手指控制不同的按鈕。）

寫一個程式來幫助計算每個手指按下按鈕的次數。一根手指按下

如果註釋中需要但在最後一個註釋中未使用，則為按鈕。另外，如果是第一個音符，每個手指

需要按下按鈕

輸入輸入

第一行輸入的是單一整數t(1≤t≤1000)，表示測試的次數

案例。對於每種情況，只有一行包含歌曲。唯一允許的字元是

{'c ',' d','e','f','g','a','b','C','D','E','F','G',' A'，'B'}。一首歌最多有200個音符，而且歌曲

也許是空的。

輸出輸出

對於每個測試案例，列印 10 個數字，表示每個手指的按壓次數。編號是

由一個空格分隔。

輸入範例輸入範例

3 cdefgab BAGFEDC CbCaDCbCbCCbCbabCCbCbabae

範例輸出輸出範例

0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 8 10 2 0 0 2 2 1 0

解法： 1. 使用一個字元陣列紀錄每個英文字母指法。

# 118 - 變種平面世界探險家

機器人技術、機器人運動規劃和機器學習是跨越許多學科界限的領域

計算機科學的子學科：人工智慧、演算法和複雜性，

電氣和機械工程僅舉幾例。此外，機器人就像「海龜」（靈感來自

Papert、Abelson 和 diSessa 的工作）和「蜂鳴器拾取器」（受 Pattis 的工作啟發）

多年來，學生們一直在研究和使用它作為程式設計入門。

這個問題涉及確定探索前哥倫布時期平坦世界的機器人的位置。

給定矩形網格的尺寸以及機器人位置和指令的序列，您

編寫一個程式來確定每個機器人位置序列和指令的最終結果

機器人的位置。

機器人位置由網格座標（一對整數：x 座標後面跟著 y 座標）和方向（ N、S 、E、W 表示北、南、東和西）組成。機器人指令是一個字串

字母「L」、「R」和「F」分別代表指令：

•左轉：機器人左轉90 度並保持在目前網格點。

•右轉：機器人右轉90 度並保持在目前網格點上。

•前進：機器人沿著目前方位方向前進一個格點，

保持相同的方向。

北方向對應於從網格點 (x, y) 到網格點 (x, y + 1) 的方向。

由於網格是矩形且有界的，因此「離開」網格邊緣的機器人就會遺失

永遠。然而，丟失的機器人會留下機器人“氣味”，阻止未來的機器人掉落。

世界在同一網格點。氣味留在機器人之前佔據的最後一個網格位置

消失在邊緣。一條指令，將世界從一個網格點移開，從該網格點開始

先前遺失的機器人將被目前機器人忽略。

輸入

第一行輸入的是矩形世界的右上座標，左下座標

假設為 0,0。

其餘輸入由一系列機器人位置和指令組成（每個機器人兩行）。

位置由兩個整數組成，指定機器人的初始座標和方向

( N,S ,E,W)，全部在一行上以空格分隔。機器人指令是一串字母「L」、「R」、

和“F”在一行上。

每個機器人都是按順序處理的，即在下一個機器人指令之前完成執行機器人指令。

機器人開始執行。

輸入以檔案結尾結束。

您可以假設所有初始機器人位置都在指定網格的範圍內。這

的最大值為 50。所有指令字串將少於 100 個字元

長度。

輸出輸出

對於輸入中的每個機器人位置/指令，輸出應指示機器人的最終網格位置和方向。如果機器人從網格邊緣跌落，則應在位置和方向後面列印“LOST”一詞。

輸入範例輸入範例

5 3

1 1 E

射頻射頻射頻射頻

3 2 N

FRRRFLFFFRRRFL

0 3 W

LLFFFLFLFL

Sample Output 輸出範例

1 1 E 3 3 N LOST 2 3 S

解法：

1. 自定義一個method判斷進行右轉或左轉後的方位。(也可以把此功能直接打main裡面，只是感覺有點亂)

2. 使用一個陣列紀錄機器人超出界線前的位子。

3. 判斷是不是"之前"有機器人超出界線or"目前"的機器人超出界外，再依情況輸出結果。

# 11150 – 可樂

您看到便利商店的以下特價優惠：

“每退回 3 個空瓶即可獲得一瓶巧克力”

現在您決定從商店購買一些（假設 N 瓶）可樂。你想知道如何

你可以從他們那裡得到最多的可樂。

下圖顯示了 N = 8 的情況。方法 1 是標準方法：完成 8 後

瓶可樂，你有 8 個空瓶。拿走 6 個即可獲得 2 瓶新可樂。現在之後

喝完它們，你有 4 個空瓶子，所以你拿 3 個空瓶子來獲得另一杯新可樂。最後，

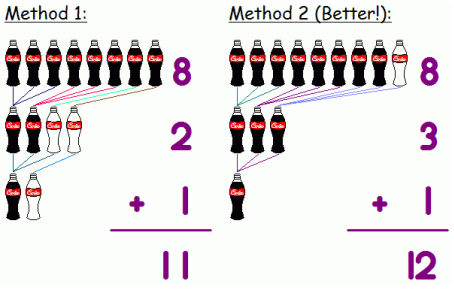
你手上只有2瓶，所以你不能再買新的可樂了。因此，您享受了8+2

+ 1 = 11 瓶可樂。

其實你可以做得更好！在方法2中，你先向你的朋友借一個空瓶子（？！或者

店主？？），那麼你就可以享受8+3+1=12瓶可樂！當然，你必須

將剩餘的空瓶歸還給您的朋友。



輸入

輸入由多行組成，每行包含一個整數 N (1 ≤ N ≤ 200)。

輸出

對於每種情況，您的程式應該輸出您可以享用的最大可樂瓶數。您可以向他人借用空瓶子，但如果您這樣做，請確保事後有足夠的瓶子可以歸還給他們。注意：喝太多可樂對健康有害，所以…不要在家嘗試這個！！ :-)

輸入範例

8

輸出範例

12

解法：當N(可樂瓶子數量)<=1時，可以直接印出目前的可樂瓶數，因為你再怎麼借，也無法歸還空瓶給朋友。當N>1時，用個變數(ans)紀錄已經喝過的可樂，再重新計算可樂瓶子數量(N)，直到可樂瓶子數量(N)=2時，可以跟朋友借一個瓶子，即可換得最後1瓶可樂，喝完的可樂瓶剛好可以還給朋友。當可樂瓶子(N)=1時，結束迴圈，印出答案。