***ÔN THI SRV***

Contents

[Overview 2](#_Toc185585363)

[SMPDIV - Divisibility 2](#_Toc185585364)

[PROB01 - Số nén 1 2](#_Toc185585365)

[PRIME1E - Prime Generator (Easy) 3](#_Toc185585366)

[SUMPROE - SUM OF PRODUCT (Easy) 4](#_Toc185585367)

[CALEXP - Phép tính lũy thừa 5](#_Toc185585368)

[NUMPER - Số chính phương 6](#_Toc185585369)

[Array 6](#_Toc185585370)

[PROB02 - Hệ thống phân tích rèm cửa 6](#_Toc185585371)

[PROB03 - Vùng bao lớn nhất 9](#_Toc185585372)

[PROB06 - Hệ thống viễn thông 10](#_Toc185585373)

[QKPE - Queens, Knights and Pawns (Easy) 11](#_Toc185585374)

[PAIRS1E - Count the Pairs (Easy) 13](#_Toc185585375)

[QUE1E - Queue Rookie (Easy) 13](#_Toc185585376)

[FSTDUP - First duplicate 14](#_Toc185585377)

[BLDTWR - Xây tháp 15](#_Toc185585378)

[CHKSDK - Kiểm tra Sudoku 16](#_Toc185585379)

[OVCOBS - Vượt chướng ngại vật 17](#_Toc185585380)

[LOGOSYM - Logo đối xứng 20](#_Toc185585381)

[FNDMTX - Tìm kiếm ma trận 21](#_Toc185585382)

[BTFNUM - Số đẹp 22](#_Toc185585383)

[SLTKNGHT - Quân mã được chọn 23](#_Toc185585384)

[String 24](#_Toc185585385)

[TOANDFRO - To and Fro 24](#_Toc185585386)

[ABSYSE - Anti-Blot System (Easy) 25](#_Toc185585387)

[FNDSTR - Tìm kiếm String 26](#_Toc185585388)

[CNTSYMST - Đếm chuỗi con đối xứng 27](#_Toc185585389)

[MAXSYMST - Chuỗi con đối xứng dài nhất 28](#_Toc185585390)

# Overview

|  |
| --- |
| SMPDIV - Divisibility |

Print all integers *ai* such that *ai* is divisible by *x* and not divisible by *y*, where 1 < *ai < n* < 100000.

Input

First, you are given *t* (*t* < 100) - the number of test cases. In each of the following *t* lines, 3 integers: *n x y*.

You might assume also that *x* < *n* and *x* is not divisible by *y*.

Output

In each of the following *t* lines, numbers requested in the problem description in the separated by a single space in ascending order.

A screenshot of a computer program

Description automatically generated

|  |
| --- |
| PROB01 - Số nén 1 |

Problem

Tèo và Tí là đôi bạn thân, họ thường đố nhau về những bài học đã được học trên trường. Cách làm này làm 2 bạn rất dễ áp dụng kiến thức đã học và nhanh hiểu bài.

Một hôm, Tí qua nhà Tèo chơi và đố Tèo về một phép toán nén số nguyên.

Nghe qua thì rất đơn giản, Tí định nghĩa phép nén một số nguyên dương là tính tổng các chữ số của nó.

Sao cho, sau một số phép nén, thì số còn lại chỉ có một chữ số và không nén được nữa.

Ví dụ cho số 86. Sau phép nén thứ nhất ta đk: 8+6=14. Sau phép nén thứ 2: 1+4=5 => Số nén tối giản của 86 là 5.

Cho một số nguyên hãy tìm số nén tối giản của nó.

Tí rất hứng thú với phép toán này, và cậu quyết định lập trình để giải quyết bài toán thay vì tính tay. Các bạn hãy cùng Tí giải quyết bài toán này nhé.

Input

Dòng đầu chứa số bộ test (1 <= T <= 100).

T dòng tiếp theo, mỗi dòng tiếp theo chứa 1 số nguyên dương N. (N <= 10^9)

Output

Kết quả mỗi testcase được in trên 1 dòng. Với:

Bắt đầu bằng ký tự "#", tiếp theo là số thứ tự của testcase đó, tiếp đến là 1 khoảng trắng (dấu cách), và cuối cùng là kết quả của số nén tối giản tương ứng.

A screenshot of a computer

Description automatically generated

|  |
| --- |
| PRIME1E - Prime Generator (Easy) |

Peter wants to generate some prime numbers for his cryptosystem. Help him! Your task is to generate all prime numbers between two given numbers!

Input

The input begins with the number t of test cases in a single line (t<=10). In each of the next t lines there are two numbers m and n (1 <= m <= n <= 10000000, n-m<=10000) separated by a space.

Output

For every test case print all prime numbers p such that m <= p <= n, one number per line, test cases separated by an empty line.

A screenshot of a computer

Description automatically generated

|  |
| --- |
| SUMPROE - SUM OF PRODUCT (Easy) |

Given a number **N**, find the sum of all products **x\*y** such that **N/x = y (Integer Division)**. Since, the sum can be very large, please output this modulo 1000007.

Input

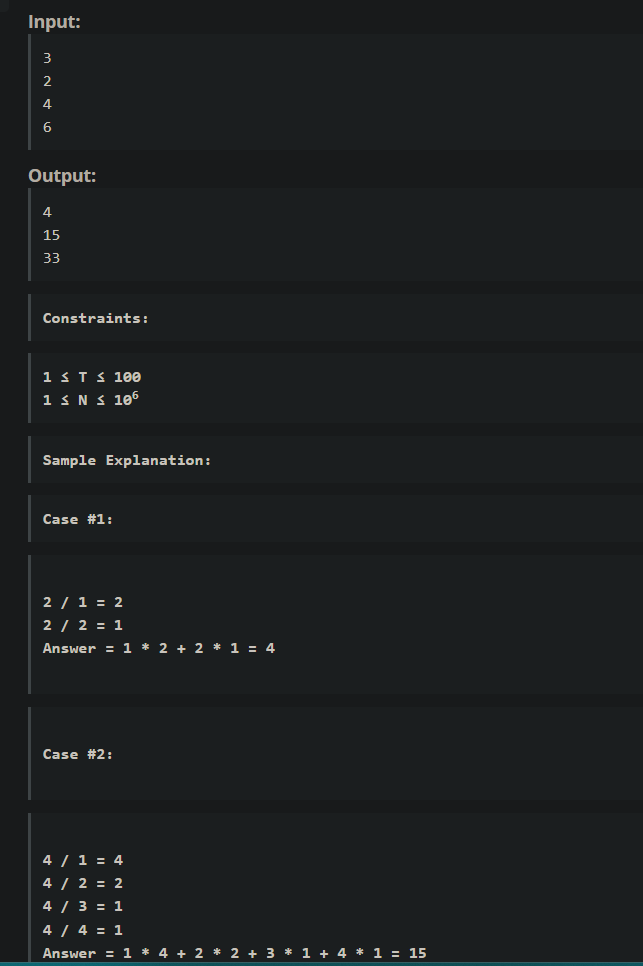
The first line of input file contains an integer **T**, the number of test cases to follow. Each of the next **T** lines contain an integer **N**.

Output

Output **T** lines containing answer to corresponding test case.

A screenshot of a computer

Description automatically generated



|  |
| --- |
| CALEXP - Phép tính lũy thừa |

Cho 2 số nguyên a và b. Hãy tính ab

Input

Dòng đầu tiên của input là số lượng test case T (T <= 20). T dòng tiếp theo chứa thông tin của các test case. Test case gồm duy nhất 1 dòng chứa 2 số nguyên a và b

Output

In kết quả theo định dạng sau: ký tự đầu tiên là "#" tiếp theo là số thứ tự của test case, tiếp theo đó là khoảng trắng (dấu cách) và cuối cùng là kết quả của phép tính.

A screenshot of a computer

Description automatically generated

|  |
| --- |
| NUMPER - Số chính phương |

Cho 2 số a, b (1 <= a <= b <= 107) và b - a <= 104. Hãy tìm các số chính phương trong khoảng [a, b].  
Biết số chính phương là số khi căn bậc hai số đó ta thu được kết quả là 1 số nguyên.

Input

Dòng đầu tiên của input là số lượng test case T (T <= 50). T dòng tiếp theo chứa thông tin của các test case. Test case gồm duy nhất 1 dòng chứa 2 số nguyên a và b

Output

In kết quả theo định dạng sau: ký tự đầu tiên là "#" tiếp theo là số thứ tự của test case, tiếp theo đó là khoảng trắng (dấu cách) và cuối cùng là các số chính phương thỏa mãn được viết cách nhau bởi khoảng trắng (dấu cách).  
Nếu trong khoảng [a, b] không có số chính phương nào, in ra "NO NUMBER".

A screenshot of a computer program

Description automatically generated

# Array

|  |
| --- |
| PROB02 - Hệ thống phân tích rèm cửa |

Công ty của Tí đang có nhu cầu bán rèm của cho các căn hộ tại một khu đô thị. Có điều Tí không dám chắc sẽ thành công với việc mở điểm bản hàng tại khu vực đó hay không.

Tí có nhờ công ty của Tèo (chuyên về phân tích thị trường) đánh giá mức độ sử dụng rèm cửa tại khu vực mà công ty của Tí đang muốn mở điểm bán hàng.

Công ty của Tèo tiến hành việc phân tích thói quen sử dụng rèm cửa của những hộ dân trong khu đô thị đó bằng cách chụp một tấm ảnh có chứa tất cả các tầng của toàn nhà,

mang về xử lý để có thể mã hoá thành dữ liệu hệ thống phân tích có thể làm việc được. Sau khi mã hoá xong, hình ảnh trạng thái của một cửa sổ sẽ như sau:

A grid with people in it

Description automatically generated

Các cửa sổ có thể có rèm hoặc không, và được biểu diễn bởi matrix 4\*4.

Do hệ thống đã cũ và lạc hậu, dẫn đến độ chính xác không cao. Công ty của Tèo đã quyết định lập trình lại hệ thống phân tích của mình với mong muốn như sau:

Cho vào matrix đã mã hoá có kích thước M\*N, tính ra số trạng thái của mỗi loại rèm cửa trong bức ảnh đó.

Các bạn hãy cùng thử sức với nhiệm vụ lập trình xây dựng hệ thống này nhé.

Input

Dòng đầu tiên mô tả số testcase T của đề bài (1<= T <= 100)

Với mỗi testcase:

- dòng đầu tiên là 2 số M, N mô tả kích thước của bức ảnh đầu vào (1<=M, N <= 100)

- Các dòng sau được mô tả như sau:

    + Khung cửa được kí hiệu bằng dấu #

    + Cửa số kích thước 4x4 gồm 1 trong các trạng thái như trên

- Mỗi cửa sổ có kích thước 4\*4 mô tả trạng thái của rèm cửa, như hình trên

(Dễ thấy, mỗi toà nhà được biểu diễn bởi 5\*M+1 dòng và 5\*N+1 cột)

Output

Kết quả mỗi testcase được in trên 1 dòng. Với:

Bắt đầu bằng ký tự "#", tiếp theo là số thứ tự của testcase đó, tiếp đến là 1 khoảng trắng (dấu cách), và cuối cùng là số trạng thái tương ứng của mỗi loại (cách nhau bởi 1 dấu cách)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## PROB03 - Vùng bao lớn nhất

Cho một ma trận kích thước M\*N (1<= M, N <= 100). Mỗi điểm trong ma trận là một số nguyên dương.

Cho 2 số W và H (1<=H<=M, 1<=W<=N). Với một hình chữ nhật tại 1 điểm bất kỳ trong ma trận có kích thước H\*W (H là chiều dọc, W là chiều ngang).

Hãy tìm ra hình chữ nhật có tổng các số chẵn là lớn nhất và in ra tổng đó.

A screenshot of a number grid

Description automatically generated

Input

Dòng đầu tiên là số lượng test case của bài toán (1 <= T <= 100)

Với mối testcase

- Dòng đầu tiên là 4 số H, W, M, N

- Tiếp đến là ma trận kích thước M\*N

Output

Kết quả mỗi testcase được in trên 1 dòng. Với:

Bắt đầu bằng ký tự "#", tiếp theo là số thứ tự của testcase đó, tiếp đến là 1 khoảng trắng (dấu cách), và cuối cùng là Kết quả của testcase đó.

A screenshot of a computer

Description automatically generated

## PROB06 - Hệ thống viễn thông

Công ty Viễn thông ABC có kế hoạch rà soát lại Hệ thống trạm phát sóng của mình.

Hệ thống trạm phát sóng của công ty  được xây dựng không đồng nhất trên cả nước (trạm xây dựng trước, trạm xây dựng sau).

Do đó chất lượng phủ sóng của các trạm này cũng sẽ khác nhau. Mỗi trạm loại A sẽ chỉ phát sóng cho phạm vi trên dưới, trái phải mỗi chiều 1km,

tương tự trạm loại B là mỗi chiều 2km, và cao cấp nhất là trạm loại C với mỗi chiều 3km. (Quan sát hình vẽ)

A screenshot of a cross

Description automatically generated

Để tiện cho việc nâng cấp, công ty viễn thông ABC cần phải tính toán số lượng hộ dân mà các trạm phát sóng này không phủ sóng tới được.

Hãy giúp công ty Viễn thông ABC thực hiện việc tính toán này.

Input

Dòng đầu tiên là số lượng testcase T của bài toán (1 <= T <= 100)

Dòng đầu tiên của mỗi testcase là 2 số M, N mô tả kích thước của bản đồ (4 <= M, N <= 100)

Tiếp theo là dữ liệu mô tả bản đồ được cho trên M dòng, mỗi dòng N ký tự với:

 - X: Vị trí không có dân cư

 - H: Vị trí có hộ dân

 - A, B, C: Vị trí tương ứng đã có trạm phủ sóng loại A, B, C

Output

Kết quả mỗi testcase được in trên 1 dòng. Với:

Bắt đầu bằng ký tự "#", tiếp theo là số thứ tự của testcase đó, tiếp đến là 1 khoảng trắng (dấu cách), và cuối cùng là Kết quả của testcase đó.

A screenshot of a computer

Description automatically generated

## QKPE - Queens, Knights and Pawns (Easy)

You all are familiar with the famous 8-queens problem which asks you to place 8 queens on a chess board so no two attack each other. In this problem, you will be given locations of queens and knights and pawns and asked to find how many of the unoccupied squares on the board are not under attack from either a queen or a knight (or both). We’ll call such squares “safe” squares. Here, pawns will only serve as blockers and have no capturing ability. The board below has 6 safe squares. (The shaded squares are safe.)

A screenshot of a grid

Description automatically generated

Recall that a knight moves to any unoccupied square that is on the opposite corner of a 2x3 rectangle from its current position; a queen moves to any square that is visible in any of the eight horizontal, vertical, and diagonal directions from the current position. Note that the movement of a queen can be blocked by another piece, while a knight’s movement can not.

Input

There will be multiple test cases. Each test case will consist of 4 lines. The first line will contain two integers n and m, indicating the dimensions of the board, giving rows and columns, respectively. Neither integer will exceed 1000. The next three lines will each be of the form

*k r1 c1 r2 c2 · · · rk ck*

indicating the location of the queens, knights and pawns, respectively. The numbering of the rows and columns will start at one. There will be no more than 100 of any one piece. Values of n = m = 0 indicate end of input.

Output

Each test case should generate one line of the form

**Board b has s safe squares.**

where *b* is the number of the board (starting at one) and you supply the correct value for *s*.

A screenshot of a computer program

Description automatically generated

## PAIRS1E - Count the Pairs (Easy)

Given N integers 0 < N < 10^4, count the total pairs of integers that have a difference of K. (Everything can be done with 32 bit integers).

Input Format

1st line contains T is number of test case (T<=50)  
  
Each test case has 2 lines.  
  
1st line of test case contains N & K (integers).  
  
2nd line of test case contains N numbers of the set. All the N numbers are distinct.

Output Format

One integer - the number of pairs of numbers that have a diff K.

A black screen with white text

Description automatically generated

## QUE1E - Queue Rookie (Easy)

There are N people standing in a Queue. You are given the height of each person and the number of people who are taller and standing ahead of him. You have to find the position of each person.

Input

First line conatins a single integer T, the number of test cases. It is followed by T test cases each of which contains 3 lines. First line of each test case contains a single integer N. Second line contains N integers representing the heights of these N people. Third line also contains N integers denoting the number of taller people standing ahead of him.

Output

Output one line for each test case which contains the heights of the N people in the order in which they are standing.

Constraints

0 < T <= 100  
  
0 < N <= 1000  
  
Expected Time Complexity = O(N2)

A screenshot of a computer program

Description automatically generated

* Thuật toán

 Đọc dữ liệu và lưu chiều cao, số người cao hơn vào danh sách cặp.

 Sắp xếp danh sách giảm dần theo chiều cao.

 Dùng danh sách trống để chèn từng người vào đúng vị trí dựa trên số người cao hơn đứng trước.

 In kết quả.

2

6

55 33 44 11 22 66

0 2 1 5 3 0

5

33 11 22 44 55

0 2 1 1 0

|  |
| --- |
| FSTDUP - First duplicate |

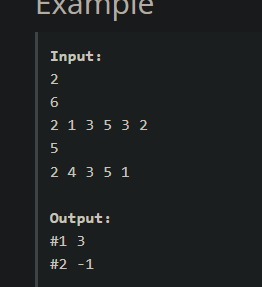
Cho một mảng kích thước N (1 <= N <= 100000) chứa các số nguyên có giá trị trong đoạn [1,N], tìm số lặp lại đầu tiên mà lần xuất hiện thứ 2 có chỉ số nhỏ nhất. Nói cách khác, nếu có nhiều hơn 1 số trùng lặp, hãy trả về số mà lần xuất hiện thứ 2 có chỉ số nhỏ hơn. Nếu không có phần tử trùng lặp thì in ra -1.  
Ví dụ: Cho mảng a = [2,1,3,5,3,2], đáp án là 3  
Có 2 số trùng lặp là 2 và 3. Lần xuất hiện thứ hai của số 3 có chỉ số nhỏ hơn lần xuất hiện thứ hai của số 2. Vì vậy đáp án là 3.  
Cho mảng a = [2,4,3,5,1], đáp án là -1

Input

Dòng thứ nhất chứa số test case, mỗi test case gồm 2 dòng, dòng đầu tiên của test case là số phần tử của mảng, dòng thứ 2 là các số trong mảng được viết cách nhau bởi dấu cách.

Output

In ra theo định dạng sau: đầu tiên là kí tự "#", tiếp theo là số thứ tự của test case, tiếp theo là khoảng trắng (dấu cách), tiếp theo là kết quả



|  |
| --- |
| BLDTWR - Xây tháp |

Cho N viên gạch có cân nặng là A1, A2, A3, ... An và độ cao của 2 tòa tháp M1, M 2. Biết số gạch đã cho vừa đủ để xây dựng 2 tòa tháp (M1 + M2 = N). Hãy viết ch ương trình tính chi phí nhỏ nhất để xây dựng 2 tòa tháp, biết rằng chi phí xây dựn g tháp được tính bằng công thức: cân nặng của viên gạch nhân với tầng để đặt vi ên gạch đó.  
Ví dụ: Cho N = 5, M1 = 3, M2 = 2  
Cho các viên gạch có cân nặng lần lượt là: 7kg, 3kg, 4kg, 1kg, 5kg  
Tòa tháp M1 được xây dựng với chi phí là:  
1 x 3 = 3 (viên gạch 1kg đặt ở tầng 3 của tháp M1)  
3 x 2 = 6 (viên gạch 3kg đặt ở tầng 2 của tháp M1)  
7 x 1 = 7 (viên gạch 7kg đặt ở tầng 1 của tháp M1)  
Tổng chi phí xây dựng tháp M1 là 16.Xây tháp  
Tòa tháp M2:  
4 x 2 = 8 (viên gạch 4kg đặt ở tầng 2 của tháp M2)  
5 x 1 = 5 (viên gạch 5kg đặt ở tầng 1 của tháp M2)  
Tổng chi phí xây dựng tháp M2 là 13  
Vậy tổng chi phí nhỏ nhất để xây dựng 2 tòa tháp là 16 + 13 = 29  
Điều kiện:  
1 <= N, M1, M2 <= 50  
1 <= cân nặng viên gạch <= 1000

Input

Dòng thứ nhất chứa số test case, mỗi test case gồm 2 dòng, dòng đầu tiên của test case là 3 số lần lượt là N, M1 và M2, dòng thứ 2 là N số thể hiện cân nặng của N viên gạch cách nhau bởi dấu cách.

Output

In kết quả theo định dạng sau: kí tự đầu tiên là "#", tiếp theo là số thứ tự của test case, tiếp theo là khoảng trắng (dấu cách), và cuối cùng là kết quả

A screenshot of a computer

Description automatically generated

|  |
| --- |
| CHKSDK - Kiểm tra Sudoku |

Trò chơi sudoku là một trò chơi điền các số từ 1 đến 9 vào ma trận 9x9 sao cho c ác số không bị trùng lặp trên hàng ngang, hàng dọc và ma trận con 3x3. Cho 1 ma trận 9x9 tượng trưng cho 1 đề Sudoku, trong đó vị trí của các dấu “.” là vị trí chưa được giải.  
Viết một chương trình kiểm tra tính đúng đắn của một đề sudoku. Đề sudoku hợp lệ là không có 2 số trùng nhau trên cùng 1 hàng, 1 cột và ma trận con 3x3. Nếu h ợp lệ in ra số 1, không hợp lệ in ra số 0.

Input

Dòng đầu tiên của input là số test case, mỗi test case sẽ được viết trên 9 dòng.

Output

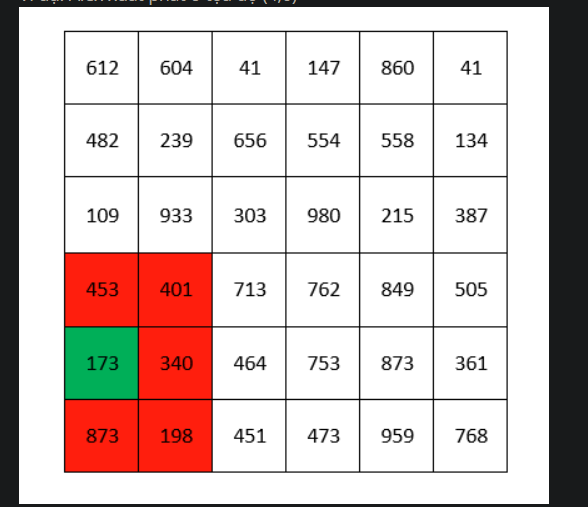
In ra theo định dạng sau: đầu tiên là kí tự "#", tiếp theo là số thứ tự của test case, tiếp theo là khoảng trắng (dấu cách), tiếp theo là kết quả.

A screen shot of a black background

Description automatically generated

|  |
| --- |
| OVCOBS - Vượt chướng ngại vật |

Cho một ma trận vuông kích thước NxN (N <= 100) chứa các số nguyên nằm trong khoảng từ 1 đến 1000 là độ cao của các chướng ngại vật.  
Alex được đặt vào vị trí x, y trong ma trận, anh ta phải nhảy lên 1 vị trí có độ cao cao hơn vị trí mà anh ta đang đứng, tuy nhiên Alex chỉ có đủ sức để nhảy lên vị trí có độ cao thấp nhất trong số những chướng ngại vật xung quanh anh ta.  
Alex sẽ dừng lại khi không còn chướng ngại vật để vượt qua.  
Cho biết kích thước ma trận, vị trí xuất phát của Alex và ma trận chướng ngại vật  
Hãy viết 1 chương trình đếm số chướng ngại vật mà Alex đã vượt qua.  
Ví dụ: Alex xuất phát ở tọa độ (4,0)



Vị trí màu xanh là vị trí xuất phát của Alex, các chướng ngại vật xung quanh Alex được đánh dấu màu đỏ.  
Vì Alex chỉ có thể nhảy sang được vị trí chướng ngại vật có độ cao thấp nhất nên chướng ngại vật có độ cao 198 được chọn.

A screenshot of a math game

Description automatically generated

Sau khi di chuyển, vị trí hiện tại của Alex là chướng ngại vật có độ cao 198 (vị trí màu xanh lá cây), các chướng ngại vật thỏa mãn điều kiện xung quanh Alex được đánh dấu màu đỏ. Vì 173 là chướng ngại vật có độ cao thấp hơn vị trí Alex đan g đứng nên không thỏa mãn điều kiện.  
Vì vậy, vị trí tiếp theo Alex di chuyển đến đó là chướng ngại vật có độ cao 340.  
……  
Tiếp tục di chuyển như vậy ta thu được quá trình di chuyển của Alex theo thứ tự: 173 – 198 – 340 – 401 – 453 – 933

A grid of numbers with red squares

Description automatically generated

Vì vậy, không tính điểm xuất phát, Alex đã vượt qua 5 chướng ngại vật, kết quả cần in ra là 5.

Input

Dòng đầu tiên của input là số test case.  
Mỗi test case gồm nhiều dòng, dòng đầu tiên chứa 3 số N, x, y, trong đó N là kích thước của ma trận, x và y là tọa độ xuất phát của Alex.  
N dòng tiếp theo chứa thông tin của ma trận.

Output

In ra theo định dạng sau: đầu tiên là kí tự "#", tiếp theo là số thứ tự của test case, tiếp theo là khoảng trắng (dấu cách) và cuối cùng là kết quả.

A screenshot of a computer

Description automatically generated

|  |
| --- |
| LOGOSYM - Logo đối xứng |

Cho một ma trận NxN (N <= 32) trong đó có các số 0 và 1 biểu thị ma trận màu của logo. Vị trí có giá trị bằng 1 là vị trí được tô màu, vị trí có giá trị bằng 0 là vị trí không được tô màu.  
Ví dụ ma trận 5x5 như sau:  
0 1 1 1 0  
0 1 0 1 0  
1 0 0 0 1  
0 1 0 1 0  
0 1 1 1 0  
ma trận được biểu diễn dưới dạng hình học như sau:A grid with blue squares

Description automatically generated

A grid with blue squares

Description automatically generatedLấy điểm trung tâm của logo làm gốc tọa độ, một logo được gọi là đối xứng khi các vị trí tô màu đối xứng với nhau qua trục hoành và trục tung của hệ tọa độ. Từ hình ta có thể thấy logo đối xứng qua trục X và trục Y của hệ tọa độ, vì vậy logo tương ứng với ma trận 5x5 bên trên được gọi là 1 logo đối xứng.  
Ví dụ ta có ma trận 5x5 như sau:  
0 0 1 0 0  
0 1 0 1 0  
1 0 0 0 1  
0 1 0 1 0  
0 1 1 1 0  
ma trận được biểu diễn dưới dạng hình học như sau:

A screenshot of a computer

Description automatically generated

Logo trên chỉ đối xứng qua trục Y, không đối xứng qua trục X vì vậy logo này không đối xứng.  
Cho ma trận biểu thị màu của logo, viết chương trình xác định logo đã cho có đối xứng hay không? Nếu đối xứng in ra YES, không đối xứng in ra NO

Input

Dòng đầu tiên của input là số test case, mỗi test case được viết trên nhiều dòng.  
Dòng đầu tiên của test case là kích thước của ma trận N.  
N dòng tiếp theo là dữ liệu của ma trận.

Output

In kết quả theo định dạng sau: đầu tiên là kí tự "#", tiếp theo là số thứ tự của test case, tiếp theo là khoảng trắng (dấu cách), tiếp theo là YES nếu logo là đối xứng, NO nếu logo không đối xứng.

## FNDMTX - Tìm kiếm ma trận

Cho một ma trận vuông A kích thước NxN (5 <= N <= 100) chứa các số nguyên nằm trong khoảng từ 1 đến 1000, số nguyên M (2 <= M <= N-1) là kích thước ma trận con của ma trận A và số nguyên K.  
Hãy tìm vị trí bắt đầu của ma trận con có kích thước là MxM trong ma trận A sao cho tổng tất cả các số trong ma trận con có giá trị gần nhất với số nguyên K. Nếu có 2 vị trí thỏa mãn thì lấy vị trí ma trận con có tổng nhỏ hơn. Nếu có 2 ma trận con có cùng tổng thì lấy ma trận con có chỉ số hàng nhỏ hơn, nếu có 2 ma trận cùng hàng thỏa mãn thì lấy ma trận có chỉ số cột nhỏ hơn.  
Ví dụ: số nguyên K = 50, ta có 2 tổng của ma trận con là 25 và 75 thì ta sẽ in vị trí của ma trận có tổng là 25.  
Nếu tổng của cả 2 ma trận đều là 25, nhưng một ma trận có chỉ số tọa độ điểm bắt đầu là (3,4), một ma trận có chỉ số tọa độ bắt đầu là (3,5) thì in vị trí (3,4).

Input

Dòng đầu tiên của input chứa số test case.  
Các dòng tiếp theo sẽ chứa thông tin của test case, trong đó:  
- Dòng đầu tiên của mỗi test case là 3 số lần lượt là N, M, K.  
- N dòng tiếp theo là dữ liệu của ma trận A.

Output

In ra kết quả là vị trí bắt đầu của ma trận thỏa mãn (vị trí bắt đầu là vị trí thuộc hàng đầu tiên và cột đầu tiên) theo dạng “#tc x y”.  
Trong đó:  
tc: số thứ tự của test case.  
x: là số thứ tự của hang trong ma trận A.  
y: là số thứ tự của cột trong ma trận A.

A screenshot of a computer

Description automatically generated

|  |
| --- |
| BTFNUM - Số đẹp |

Tại một số quốc gia, có một số các số được coi là số đẹp hơn các số khác. Ví dụ tại đất nước A, số 5 và số 7 là số đẹp.  
Công ty Samsung muốn bán được nhiều sản phẩm nhất tại nước A, nhưng người nước A chỉ mua sản phẩm khi sản phẩm đó được đánh số trong đó có n số đẹp.  
Ví dụ:  
Các sản phẩm được đánh số từ x đến y (x,y <100 000) ví dụ trong trường hợp này là từ 0 <= mã sản phẩm <= 1000, muốn bán được sản phẩm cần phải chứa ít nhất 2 số đẹp trong đó (là 5 và 7).  
Vậy nên các số như 55, 77, 57, 75, 55x, 77x, 57x, 75x ... được coi là những sản phẩm có số đẹp và bán được.  
Yêu cầu in ra số lượng các số thỏa mãn yêu cầu trên.  
Ví dụ như trên có đáp án là: 55, 77, 57, 75, 77x, 55x, 57x, 75x, 7x7, 5x5, 7x5, 5x7, x55, x57, x75, x77 => 4 + 8 \* 10 + 4 \* 9 = 120 số.

Input

Dòng đầu tiên là số lượng test case. Mỗi test case được viết trên 3 dòng.  
Dòng thứ nhất là số lượng các số đẹp (n số), số lượng các số đẹp yêu cầu có trong mã sản phẩm (m số).  
Dòng thứ 2 gồm n số là các số đẹp cần có trong mã sản phẩm.  
Dòng thứ 3 là 2 số x và y (x <= mã sản phẩm <= y)

Output

In ra theo định dạng sau: đầu tiên là ký tự "#", tiếp theo là số thứ tự của test case, tiếp theo là khoảng trắng (dấu cách), tiếp theo là số lượng các mã sản phẩm thỏa mãn

A screenshot of a computer

Description automatically generated

## SLTKNGHT - Quân mã được chọn

Đôi bạn thân Tí và Tèo đang cùng nhau học chơi Cờ vua, Tí nhận thấy quân Mã trong cờ vua rất thú vị. Vì nó không bị chặn bởi những quân cờ khác mà những bước di chuyển của nó còn đa dạng và khó đoán.  
Cùng lúc này, Tèo nghĩ ra câu hỏi nếu trên bàn cờ có nhiều hơn 4 quân mã thì với thế cờ hiện tại trên bàn cờ, quân mã nào sẽ ăn được nhiều quân cờ nhất có thể.  
Bài toán đặt ra là có rất nhiều quân mã trên bàn cờ, trên đó còn những quân cờ mà quân mã được phép ăn cũng như không được phép ăn. Tèo muốn biết quân mã nào ăn được nhiều quân cờ nhất trên bàn cờ nên đã đưa bài toán này cho Tí.  
Để không mất thời gian kiểm tra lại kết quả của Tí đưa ra, cũng như Tèo sợ Tí chơi ăn gian (đoán mò) nên muốn viết một chương trình để tính toán kết quả của bài toán trên.  
Các bạn hãy cùng Tèo viết chương trình này các bạn nhé.

Input

Dòng đầu tiên là số testcase T của bài toán.  
Mỗi testcase được cho trên 4 dòng, với các thông tin sau:  
- Dòng đầu tiên chứa: kích thước bàn cờ N (1 <= N <= 1000), số quân mã hiện có trên bàn cờ K ( 1 <= K <= N\*N-1 ), số quân tốt hiện có trên bàn cờ M ( 1 <= M <= N\*N-K), số quân cờ không được phép ăn D ( 1 <= D <= M)  
- Dòng tiếp theo chứa 2\*K số nguyên (x, y) cho biết vị trí của quân mã trên bàn cờ (0 <= x, y < N)  
- Dòng tiếp theo chứa 2\*M số nguyên (x, y) cho biết vị trí của những quân cờ còn lại trên bàn cờ (0 <= x, y < N)  
- Dòng cuối cùng chứa 2\* D số nguyên (x, y) cho biết vị trí của những quân cờ không được phéo ăn trên bàn cờ (0 <= x, y < N)  
Dữ liệu đảm bảo không có 2 quân cờ nào trùng nhau, những vị tri quân cờ không được phép ăn đã tồn tại trong bàn cờ.

Output

Mỗi test case được in ra trên một dòng, với:  
Bắt đầu bằng ký tự "#", tiếp theo là số thứ tự của testcase đó, tiếp đến là 1 khoảng trắng (dấu cách), tiếp theo là 2 số x, y cho biết vị trí của quân mã ăn được nhiều quân cờ còn lại nhất, tiếp đến là 1 khoảng trắng, và cuối cùng là số quân cờ mà quân mã này đã ăn. Nếu có nhiều hơn 1 quân mã ăn được nhiều quân cờ còn lại nhất, thì in ra quân mã đầu tiên trong dãy các quân mã nhập vào.  
(Chi tiết xem thêm tại example output)

A number on a black background

Description automatically generated

# String

## TOANDFRO - To and Fro

Mo and Larry have devised a way of encrypting messages. They first decide secretly on the number of columns and write the message (letters only) down the columns, padding with extra random letters so as to make a rectangular array of letters. For example, if the message is “There’s no place like home on a snowy night” and there are five columns, Mo would write down

A screenshot of a computer screen

Description automatically generated

Note that Mo includes only letters and writes them all in lower case. In this example, Mo used the character ‘x’ to pad the message out to make a rectangle, although he could have used any letter. Mo then sends the message to Larry by writing the letters in each row, alternating left-to-right and right-to-left. So, the above would be encrypted as



Your job is to recover for Larry the original message (along with any extra padding letters) from the encrypted one.

Input

There will be multiple input sets. Input for each set will consist of two lines. The first line will contain an integer in the range 2...20 indicating the number of columns used. The next line is a string of up to 200 lower case letters. The last input set is followed by a line containing a single 0, indicating end of input.

Output

Each input set should generate one line of output, giving the original plaintext message, with no spaces.

A screenshot of a computer program

Description automatically generated

## ABSYSE - Anti-Blot System (Easy)

Jimmy is a hard-working pupil in his second year at primary school. Recently he decided to convert all his notes into an electronic version. Sadly, he found that his math notes were full of ink blots. He scanned the notes and sent them through his own OCR package (yes, he coded it all by himself at the age of 8). The OCR package replaced all ink blots by the string "machula".

Problem specification

You are given Jimmy's notes, processed by the OCR. They contain simple math exercises, which were used to practice addition on positive integers. Your task is to recover the damaged part of the notes.

Input specification

The first line of the input file contains an integer T specifying the number of test cases. Each test case is preceded by a blank line. Each test case consists of exactly one line. The line represents an equation of the form "number + number = number", where each number is a positive integer. One part of the equation will be replaced by the string "machula". The string always covers a contiguous non-empty sequence of digits, possibly even an entire number. You may assume that for each equation in the input there will be exactly one way to fill in the missing digits.

Output specification

For each test case, the output shall contain one line of the form "number + number = number". The line must represent the equation from that test case with all missing digits filled in.

A screenshot of a computer program

Description automatically generated

|  |
| --- |
| FNDSTR - Tìm kiếm String |

Cho 2 String S1 (0 <= S1.length() <= 10) và S2 (0 <= S2.length() <= 1000). Biết rằng String S1 có độ dài luôn nhỏ hơn String S2. Hãy tìm số lần xuất hiện của String S1 trong String S2.

Input

Có tổng cộng 10 test case.  
Mỗi test case gồm 3 dòng.  
Dòng đầu tiên của mỗi test case là số thứ tự của test case đó.  
Dòng thứ 2 của mỗi test case là String S1.  
Và dòng thứ 3 của mỗi test case là String S2.  
Lưu ý, String S2 có thể chứa khoảng trắng (dấu cách).

Output

In đáp số theo định dạng sau:  
Ký tự đầu tiên là ký tự "#", tiếp theo đó là số thứ tự của test case, tiếp theo là số lần xuất hiện của String S1 trong String S2.

|  |
| --- |
| CNTSYMST - Đếm chuỗi con đối xứng |

Cho 1 chuỗi có độ dài tối đa là 1000 ký tự viết thường.  
Viết chương trình đếm xem trong chuỗi đó có bao nhiêu chuỗi con là chuỗi đối xứng.  
Chuỗi đối xứng là chuỗi khi đọc từ trái sang phải hay từ phải sang trái đều như nhau.

Input

Dòng đầu tiên trong input là số test case. Mỗi test case được viết trên một dòng duy nhất, là chuỗi đầu bài cho.

Output

In ra theo định dạng sau: đầu tiên là kí tự "#", tiếp theo là số thứ tự của test case, tiếp theo là khoảng trắng (dấu cách) và cuối cùng là kết quả

A screenshot of a computer program

Description automatically generated

|  |
| --- |
| MAXSYMST - Chuỗi con đối xứng dài nhất |

Cho 1 chuỗi có độ dài tối đa là 1000 ký tự viết thường  
Viết chương trình in ra độ dài của chuỗi con đối xứng dài nhất  
Chuỗi đối xứng là chuỗi khi đọc từ trái sang phải hay từ phải sang trái đều như nhau

Input

Dòng đầu tiên của input là số test case. Mỗi test case được viết trên một dòng duy nhất, chứa chuỗi đầu bài cho.

Output

In kết quả theo định dạng sau: đầu tiên là kí tự "#", tiếp theo là số thứ tự của test case, tiếp theo là khoảng trắng (dấu cách) và cuối cùng là kết quả.

Example

A screenshot of a computer program

Description automatically generated

# TEST

PTRNLOCK - Pattern lock

A screen shot of a cell phone

Description automatically generated

A screenshot of a math test

Description automatically generated

A white paper with black text

Description automatically generated

A screenshot of a computer

Description automatically generated