Backtrack

624. CD

Bạn có một chuyến lái xe đường trường. Bạn có một máy ghi âm, nhưng thật không may bài hát hay nhất của bạn lại lưu trên những chiếc CD. Bạn cần phải có ghi vào các đoạn băng, nên vấn đề bạn cần giải quyết là: Bạn có một đoạn băng dài N phút. Làm sao chọn những bài từ CD để sử dụng được nhiều nhất phần trống của đoạn băng và phần thừa trên băng là ít nhất có thể.

Giả sử :

- Số lượng bài trong CD không quá 20 bài.
- Không bài nào dài hơn N phút.
- Không bài nào lặp lại.
- Độ dài của mỗi bài hát là một số nguyên.
- N nguyên.

Chương trình phải tìm ra cách xếp các bài sao cho làm đầy đoạn băng nhất có thể và in nó ra dưới dạng số giống như những bài đã được lưu trên CD.

INPUT

Có nhiều test, mỗi test nằm trên một dòng. Mỗi dòng bao gồm giá trị của N, số bài hát và thời lượng của các bài. Ví dụ xét test đầu tiên trong bộ dữ liệu mẫu: N=5, số bài hát là=3, bài hát đầu tiên kéo dài 1 phút, bài 2 dài 3 phút, bài 3 dài 4 phút .

OUTPUT

Danh sách thời lượng các bài hát và sâu kí tự ``sum:" và sau đó là tổng các thời lượng.

Sample Input	Sample Output
5 3 1 3 4	1 4 sum:5
10 4 9 8 4 2	8 2 sum:10
20 4 10 5 7 4	10 5 4 sum:19
90 8 10 23 1 2 3 4 5 7	10 23 1 2 3 4 5 7 sum:55
45 8 4 10 44 43 12 9 8 2	4 10 12 9 8 2 sum:45

380. Call Forwarding

Trong 10 năm qua, công nghệ thông tin cải tiến triệt để hệ thống viễn thông – đặc biệt là hệ thống điện thoại. Chúng ta có menu tự động, máy trả lời tự động, hội thoại nhóm, địa chỉ nhóm… Một tính năng chung của một hệ thống điện thoại là khả năng thiết lập việc chuyển tiếp cuộc gọi. Ví dụ nếu Bon ở công ty Nobody's Home Company (NHC) đi nghỉ, chàng sẽ thiết lập để các cuộc gọi tới chàng sẽ được tự động chuyển cho đồng nghiệp Jane. Bài toán này yêu cầu xác định cách thức giúp công ty điện thoại quản lý được việc chuyển tiếp cuộc gọi.

Tất cả các điện thoại ở NHC đều có 4 số mở rộng. Nhân viên công ty có thể đặt chế độ chuyển tiếp cuộc gọi bằng cách nhập vào thông tin phù hợp trên máy điện thoại. Trước khi đi nghỉ, nhân viên nhập vào các thông tin sau: 4 số điện thoại mở rộng của họ, thời điểm họ đi nghỉ, khoảng thời gian đi nghỉ và 4 số điện thoại mở rộng của người đồng nghiệp sẽ nhận điện thoại hộ. Chú ý rằng:

Phần mở rộng có 4 số.

2 số 0000 and 9999 là 2 số đặc biệt không dành cho nhân viên.

Thời gian được tính theo đơn vị giờ. Đồng hồ đặt mốc 0000 vào giữa đêm của ngày 1 tháng giêng mỗi năm. Như vậy, khi xác định thời điểm vắng mặt, nhân viên phải ghi 1 con số từ 0000 đến 8784 (=366*24). Hệ thống chuyển tiếp cuộc gọi sẽ tự động được thiết lập lại sau mỗi năm

Chuyển tiếp cuộc gọi từ thời điểm X trong khoảng thời gian Y có hiệu lực từ thời điểm X đến thời điểm X+Y.

Mọi người nói chung đều nhập thông tin đúng. Họ tuân theo đúng khuôn mẫu trên. Họ không đưa ra yêu cầu như khoảng thời gian nghỉ vượt qua thời điểm cuối năm. Họ cũng không đưa ra hai yêu cầu mà khoảng thời gian chuyển tiếp có phần trùng nhau. Tuy nhiên khi người dùng đưa thông tin hợp lý, hệ thống vẫn có thể gặp trục trặc trong tình huống như Bob chuyển tiếp cuộc gọi cho Sue, Sue chuyển tiếp cuộc gọi cho Joe và Joe chuyển tiếp cuộc gọi cho Bob. Khi ai đó gọi cho 1 trong 3 người trên, cuộc gọi này sẽ được chuyển tiếp vòng quanh mãi. Để ngăn ngừa, số điện thoại đặc biệt 9999 được sử dụng để làm số chết. Cuộc gọi nào cho Joe, Sue hoặc Bob sẽ được chuyển tiếp cho số 9999.

INPUT

Dòng đầu tiên là một số N biểu thị số hệ thống (số test). 1≤N≤10. Sau đó là M dòng, ghi các yêu cầu chuyển tiếp theo khuôn dạng `*source time duration target*'. (0≤M≤100). Yêu cầu này có ý nghĩa là chuyển tiếp cuộc gọi tới số máy source sang số máy target trong khoảng duration, tính từ thời điểm time. Cả 4 giá trinh này đều có dạng `*dddd dddd dddd dddd*'. Dòng nào mà source có giá trị 0000 đánh dấu điểm kết thúc của phần input này. Danh sách các yêu cầu liệt kê theo thứ tự tiếp nhận. Kế sau là 1 hoặc nhiều dòng `*time extension*' (có khuôn dạng `*dddd dddd*'). Đó là một cuộc gọi vào số máy extension vào thời điểm time. Các dòng này được liệt kê theo thứ tự không giảm của time. Dòng có số 9000 đánh dấu sự kết thúc của nhóm dữ liệu này.

OUTPUT

Dòng Output đầu tiên ghi `CALL FORWARDING OUTPUT`. Sau đó là các dòng mô phỏng hoạt động của hệ thống chuyển tiếp cuộc gọi. Each of these sections should be headed by the line Mỗi hệ thống bắt đầu bằng dòng `SYSTEM` *N*, trong đó N là số thứ tự của hệ thống đang xét. Sau đó là các dòng kết quả của từng cuộc gọi, có khuôn dạng``AT *dddd* CALL TO *dddd* RINGS dddd". Dòng cuối cùng ghi `END OF OUTPUT`.

Sample Input	Sample Output
2 1111 0100 0200 2222 1111 0301 0500 4444 2222 0200 0200 3333 3333 0250 1000 1111 7777 1000 2000 7777 0000 0050 1111 0150 1111 0200 1111 0225 2222 0270 1111 0320 1111 0320 3333 0900 3000 1250 3333 1250 7777 9000 0000 3000 1111 9000	CALL FORWARDING OUTPUT SYSTEM 1 AT 0050 CALL TO 1111 RINGS 1111 AT 0150 CALL TO 1111 RINGS 2222 AT 0200 CALL TO 1111 RINGS 3333 AT 0225 CALL TO 2222 RINGS 3333 AT 0270 CALL TO 1111 RINGS 9999 AT 0320 CALL TO 1111 RINGS 4444 AT 0320 CALL TO 3333 RINGS 4444 AT 0900 CALL TO 3000 RINGS 3000 AT 1250 CALL TO 3333 RINGS 1111 AT 1250 CALL TO 7777 RINGS 9999 SYSTEM 2 AT 3000 CALL TO 1111 RINGS 1111 END OF OUTPUT

10576. Y2K

Accounting for Computer Machinists (ACM) bị lỗi Y2K và mất nhiều dữ liệu quan trọng để làm báo cáo thường niên cho MS Inc.

Tất cả những gì họ nhớ là MS Inc đăng báo cáo về thặng dư hoặc thâm hụt mỗi tháng của năm 1999 và điều đặc biệt là nếu thặng dư thì số tiền thặng dư luôn là s và nếu thâm hụt thì số tiền thâm hụt luôn là d (cho mọi tháng). Họ không nhớ có bao nhiêu tháng hoặc là những tháng nào mình có thặng dư hoặc thâm hụt. MS Inc, không giống các công ty khác, đăng thu nhập của mình cho mỗi 5 tháng liên tiếp trong năm. ACM biết rằng cả 8 báo

cáo này đều thông báo tình trạng thâm hụt, nhưng họ không biết là bao nhiêu. Kế toán trưởng gần như chắc chắn rằng MS Inc có thặng dư trong cả năm 1999. Chắc chắn nhưng không tuyệt đối.

Viết một chương trình xác định liệu MS Inc bị thâm hụt trong năm 1999, hoặc nếu họ có thặng dư trong năm 1999 thì số tiền thặng dư tối đa là bao nhiêu.

INPUT & OUTPUT

Input là dãy các dòng, mỗi dòng có chứa hai số nguyên dương s và d. Với mỗi dòng input, xuất ra 1 dòng chứa hoặc là giá trị thặng dư của năm đó, hoặc xuất ra dòng chữ `Deficit' nếu công ty thâm hụt.

Sample Input	Sample Output
59 237 375 743 200000 849694 2500000 8000000	116 28 300612 Deficit

11085. BACK TO THE 8-QUEENS

Trên bàn cờ (kích thước 8×8) có 8 quân hậu được đặt một cách ngẫu nhiên (mỗi con nằm trên một cột khác nhau để không con nào có thể ăn nhau theo chiều dọc). Tuy nhiên, một số quân hậu có nguy cơ bị các quân khác ăn theo đường chéo hoặc chiều ngang. Bạn phải di chuyển các quân hậu làm sao không có quân nào có nguy cơ bị ăn. Bạn chỉ được di chuyển các quân hậu theo hàng ngang. Yêu cầu đề ra là bạn phải in ra số bước đi ngắn nhất để không có quân hậu nào có thể bị ăn.

INPUT

Sẽ có nhiều test (ít hơn 1000 test). Mỗi dòng các 8 số nguyên thuộc [1,8]. Số thứ i thể hiện tọa độ hàng của quân hậu trong cột thứ i.

OUTPUT

Đối với mỗi trường hợp in ra 1 số nguyên duy nhất thể hiện cho số bước đi tối thiểu.

Sample Input	Sample Output
1 2 3 4 5 6 7 8 1 1 1 1 1 1 1 1	Case 1: 7 Case 2: 7

729. THE HAMMING DISTANCE PROBLEM

Khoảng cách Hamming của hai chuỗi bit là số lượng các bit các vị trí giống nhau nhưng có giá trị khác nhau. Người ta có thể đo khoảng cách này bằng các áp dụng phép toán XOR trên các bit tương ứng của hai chuỗi bit, rồi đếm số số 1 trên chuỗi kết quả. (a và b là 2 bit, a XOR b=1 khi a≠b). Ví dụ với 2 chuỗi bit A và B sau:

A 0 1 0 0 1 0 1 0 0 0

B 1 1 0 1 0 1 0 1 0 0

A XOR B = 1 0 0 1 1 1 1 1 0 0

Khoảng cách Hamming (*H*) của hai chuỗi này là 6.

INPUT

Input có nhiều bộ test. Dòng đầu tiên chứa số bộ test, sau đó là một dòng trống. Mỗi test nằm trên 1 dòng, chứa số nguyên N là độ dài chuỗi bit và số H là khoảng cách Hamming. Môi test cách nhau một dòng trống.

OUTPUT

Với mỗi test, in ra tất cả các chuỗi độ dài N mà khoảng cách Hamming với chuỗi có n số 0 là H. Có nghĩa là tất cả các chuỗi có độ dài N, trong đó có đúng H số 1. Các chuỗi phải được in ra theo thứ tự từ điển. (1≤H≤N≤16)

Sample Input	Sample Output
1 4 2	0011 0101 0110 1001 1010 1100

598. BUNDLING NEWSPAPERS

Một công ty nghiên cứu thống kê cần có một chương trình để tạo ra hướng dẫn cho một ứng dụng tạo ra câu hỏi thăm dò về báo chí. Ứng dụng đòi hỏi hướng dẫn là danh sách của các tờ báo, cách nhau bằng dấu phẩy, mỗi danh sách bao gồm một tập hợp con của tập tất cả các tờ báo trong một thành phố nào đó. Chương trình của bạn sẽ đọc vào một danh sách của các tờ báo và xuất ra tập hợp con các tờ báo có kích thước nào đó; các tờ báo trong tập con được sắp xếp theo một thứ tự.

INPUT

Dòng đầu tiên của input là một số nguyên M là số test, sau đó một dòng trống, tiếp theo là M bộ dữ liệu cho các bộ test. Có một dòng trống giữa các bộ dữ liệu.

Mỗi bộ dữ liệu bao gồm một dòng đầu tiên mô tả các độ lớn của các tập con, theo sau là các dòng ghi tên của mỗi tờ báo, mỗi tờ trên một dòng. Mỗi dòng được kết thúc bằng ký tự EOL. Tên báo sẽ có ít nhất 30 ký tự, và sẽ có tối đa 12 tên tờ báo.

OUTPUT

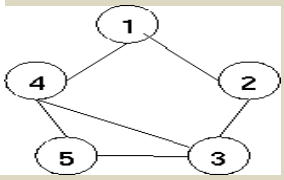
Với mỗi bộ test, in ra các tập con có kích thước chỉ định.

Sample Input	Sample Output
1 2 3 Times Herald-Tribune Post New Advocate	Size 2 Times, Herald-Tribune Times, Post Times, New Advocate Herald-Tribune, Post Herald-Tribune, New Advocate Post, New Advocate Size 3 Times, Herald-Tribune, Post Times, Herald-Tribune, New Advocate Times, Post, New Advocate Herald-Tribune, Post, New Advocate

677. ALL WALKS OF LENGTH N

Người ta có thể biểu diễn một mạng máy tính dưới dạng đồ thị. Giả sử G = (V, E) là đồ thị vô hướng và V=(v_1, v_2,…, v_m) là tập hợp m đỉnh của đồ thị (v_1 là đỉnh đầu tiên, v_m là đỉnh cuối cùng), và E là tập hợp các cạnh của đồ thị, đồ thị có k cạnh. Định nghĩa ma trận A = $(a_{ij})_{mxm}$ như sau: nếu v_i và v_j có cạnh nối với nhau thì a_{ij}=1, nếu không a_{ij}=0. Hình vẽ dưới đây minh họa ma trận và đồ thị tương ứng:

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$



Tính toán : A_n = A.A.A…A (có n số A) và sử dụng toán tử Boolean trong đó 0+0=0; 0+1=1+0=1; 1+1=0.

Phần tử hàng i, cột j của A_n là 1 khi và chỉ khi ít nhất có một đường đi có độ dài n giữa nút thứ i và nút thứ j của đồ thị V. Nói cách khác có thể có nhiều đường có độ dài n giữa đỉnh i và đỉnh j của đồ thị (chú ý trên đường đi có thể xuất hiện 1 đỉnh nhiều lần)

Ví dụ sau minh họa đường đi có độ dài 2.

$$A^2 = A \cdot A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Hãy viết chương trình tính toán và in ra số đường đi có độ dài n. (Hạn chế n≤5 và m≤10).

INPUT

Dòng đầu tiên ghi số lượng test. Các test cách biệt nhau bằng số -9999. Mỗi test bắt đầu bằng 1 dòng ghi hai số: số đỉnh trong đồ thị (m) và số bước đi (n). m dòng kế tiếp ghi ma trận cạnh.

OUTPUT

Với mỗi test, in ra số đường đi có độ dài n, của tất cả các nút, bắt đầu từ nút đầu tiên và liệt kê theo thứ tự từ điển. In ra dòng `no walk of length *n*' nếu không có đường đi nào. Các test cách nhau bởi dòng trống

Sample Input	Sample Output
5 2 0 1 0 1 0 1 0 1 0 0 0 1 0 1 1 1 0 1 0 1 0 0 1 1 0 -9999 5 3 0 1 0 1 0 1 0 1 0 0 0 1 0 1 1 1 0 1 0 1 0 0 1 1 0	(1,2,3) (1,4,3) (1,4,5) (1,2,3,4) (1,2,3,5) (1,4,3,2) (1,4,3,5) (1,4,5,3)

574. SUM IT UP

Cho trước một tổng t và danh sách n số nguyên, hãy tìm tất cả các biểu thức cộng có toán hạng lấy từ danh sách trên và có kết quả của phép cộng là t. Ví dụ, nếu $t = 4$, $n = 6$, và danh sách là [4, 3, 2, 2, 1, 1], thì có 4 biểu thức có tổng là 4: 4, 3+1, 2+2, and 2+1+1. (Một số nếu xuất hiện bao nhiêu lần trong danh sách thì có thể xuất hiện bấy nhiêu lần trong biểu thức, và duy nhất 1 số cũng được coi là 1 biểu thức). Công việc của bạn là giải bài toán tổng quát này.

INPUT

Input có nhiều test, mỗi test nằm trên 1 dòng.

Số đầu tiên là tổng t, số thứ hai là số n – số phần tử trong danh sách, tiếp theo là n số trong danh sách x_1 x_2 … x_n. n=0 đánh dấu kết thúc input. Các số phân biệt nhau bằng dấu cách. $0 < t \le 1000$, $0 \le n \le 12$, $0 \le x_i \le 12$.

OUTPUT

Với mỗi test, in ra dòng `Sums of ', tổng t, dấu hai chấm (:). Sau đó xuất ra các biểu thức, mỗi biểu thức trên một dòng. Nếu không có biểu thức nào, in dòng NONE. Các số trong biểu thức xuất hiện theo thứ tự không giảm. Nếu không tồn tại biểu thức, in ra dòng NONE. Các biểu thức được sắp xếp theo thứ tự giảm dần (dựa theo độ lớn của các số có trong biểu thức: các biểu thức được sắp xếp theo số đầu tiên, nếu số đầu tiên giống nhau thì dựa trên số thứ hai,…. Các biểu thức phải khác nhau.

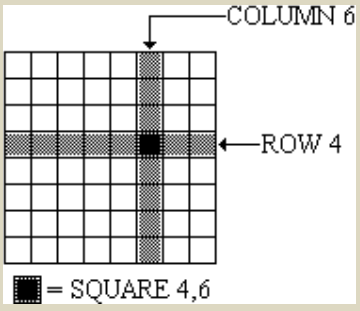
Sample Input	Sample Output
4 6 4 3 2 2 1 1 5 3 2 1 1 400 12 50 50 50 50 50 50 25 25 25 25 25 25 0 0	Sums of 4: 4 3+1 2+2 2+1+1 Sums of 5: NONE Sums of 400: 50+50+50+50+50+50+25+25+25+25 50+50+50+50+50+25+25+25+25+25+25

750. TÁM HẬU

Trên bàn cờ, người ta có thể đặt 8 quân hậu sao cho không con nào có thể ăn con nào. Hãy viết một chương trình, xác định tất cả các cách sắp xếp khả dĩ của 8 con hậu khi biết trước vị trí của một quân hậu.

INPUT

Dòng đầu tiên là số test. Sau đó là một dòng trống. Mỗi test gồm 2 số nguyên biểu diễn vị trí ô một trong 8 quân hậu đứng. Chú ý: góc trên bên trái có tọa độ (1,1). Các dòng chạy từ trên xuống dưới, dòng đầu là 1. Các cột chạy trừ trái qua phải và cột bên trái nhất là cột 1. Ô (4, 6) là hàng 4 và cột 6. Mỗi test cách nhau một dòng trống.



OUTPUT

Với mỗi test, in ra đáp án cần tìm. Các giải pháp được đánh số từ 1 đến N. Mỗi giải pháp gồm 8 số, mỗi số là tọa độ theo dòng của một cột. Ví dụ số thứ i là tọa độ dòng của con hậu đứng trên cột i… Trong ví dụ mẫu có 4 giải pháp. Biểu diễn cả 4 giải pháp được minh họa như sau:

```
    SOLUTION 1        SOLUTION 2        SOLUTION 3        SOLUTION 4

1 0 0 0 0 0 0 0   1 0 0 0 0 0 0 0   1 0 0 0 0 0 0 0   1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0   0 0 0 0 0 0 1 0   0 0 0 0 0 1 0 0   0 0 0 0 1 0 0 0
0 0 0 0 1 0 0 0   0 0 0 1 0 0 0 0   0 0 0 0 0 0 0 1   0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1   0 0 0 0 0 1 0 0   0 0 1 0 0 0 0 0   0 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0   0 0 0 0 0 0 0 1   0 0 0 0 0 0 1 0   0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0   0 1 0 0 0 0 0 0   0 0 0 1 0 0 0 0   0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0   0 0 0 0 1 0 0 0   0 1 0 0 0 0 0 0   0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0   0 0 1 0 0 0 0 0   0 0 0 0 1 0 0 0   0 0 0 1 0 0 0 0
```

Các giải pháp được sắp xếp theo thứ tự từ điển. Giữa các test có một dòng trống. Chú ý in cả số thứ tự của các giải pháp.

Sample Input	Sample Output
1 11	SOLN COLUMN # 12345678 1 15863724 2 16837425 3 17468253 4 17582463

10503. QUEEN CHESS

Một người đàn ông thường chơi domino với một số bạn bè trong một thị trấn nhỏ. Một số gia đình đã rời khỏi thị trấn và hiện nay thị trấn chỉ còn hai vợ chồng người đàn ông. Người đàn ông muốn tiếp tục chơi domino, nhưng bà vợ không thích chơi. Ông phát minh ra một trò chơi để chơi một mình. Hai quân được chọn ra và đặt ở hai đầu của một dòng có độ dài n (có thể chứa n quân domino). Sau đó, m quân khác được chọn để chèn vào chỗ trống. Số quân được chọn lớn hơn hoặc bằng độ dài của dòng ($m \geq n$), và số lượng quân đômino không vượt quá 14 ($m \leq 14$). Các quân đomino được xếp vào không gian theo quy tắc chơi domino: số điểm liền kề trên hai quân domino cạnh nhau phải giống nhau. Domino với 2 ô giống nhau cũng phải đặt ngang giống như các ô khác, không được đặt vuông góc.

Bài toán này yêu cầu viết một chương trình, nhập vào kích thước dòng n, số quân domino (m), hai quân domino ở hai đầu mút (i_1, i_2) và (d_1, d_2), và m quân đomino được đặt vào giữa (p_1, q_1), (p_2, q_2), ..., (p_m, q_m), và xác định xem có thể lấp đầy không gian dài n giữa hai quân đầu mút bằng cách sử dụng các quân đôminô trong nhóm m quân đomino.

Ví dụ, với n = 3, m = 4, 2 quân đầu mút là (0,1) và (3,4), và 4 quân (2,1), (5,6), (2,2) và (3,2), câu trả lời là CÓ, bởi vì tồn tại một giải pháp: (0,1), (1,2), (2,2), (2,3), (3,4).

Với n = 2, m = 4, 2 quân đầu mút là (0,1) và (3,4), và 4 quân (1,4), (4,4), (3,2) và (5,6), câu trả lời là KHÔNG.

INPUT

Input có nhiều test, mỗi test nằm trên nhiều dòng. Dòng đầu tiên là độ lớn không gian (n), dòng thứ hai là số lượng quân domino (m). Hai dòng tiếp theo lần lượt là quân domino được đặt bên trái và quân domino được đặt bên phải. m dòng tiếp theo là nhóm m quân đomino. Mỗi quân domino gồm 2 số, cách nhau bởi dấu cách.

OUTPUT

Với mỗi test, in ra 1 dòng YES hoặc No tùy theo đáp án.

Sample Input	Sample Output
3 4 0 1 3 4 2 1 5 6 2 2 3 2 2 4 0 1 3 4 1 4 4 4 3 2 5 6 0	YES NO

Backtrack – Chalenge

165. STAMP

Chính phủ Nova Mareterrania đòi hỏi tất cả văn bản pháp luật đều phải dán tem (để chính phủ có thêm doanh thu). Theo quy định mới, mỗi loại tài liệu chỉ có thể được dán không quá một số tem nào đó. Chính phủ muốn biết họ cần in bao nhiêu con tem khác nhau, và mệnh giá của từng con tem để đảm bảo thực hiện điều này. Mệnh giá của tem luôn là bội số của 1$

Các nhà toán học của chính phủ sau khi phân tích đã đi đến công thức n(h,k), ở đây h là số lượng tối đa tem có thể được dán vào vào một tài liệu, k là số lượng mệnh giá tem, và n(h,k) là giá trị lớn nhất có thể thu được (tổng mệnh giá các con tem trên tài liệu) trong một chuỗi liên tiếp bắt đầu từ 1$. Ví dụ, nếu h=3, k=2 và mệnh giá là 1$ và 4$, chúng ta có thể tạo ra tất cả các giá trị từ 1$ đến 6$ (8, 9 và 12$). Tuy nhiên với cùng một giá trị của h và k, nhưng bằng cách sử dụng con tem mệnh giá 1$ và 3$, chúng ta có thể tạo ra tất cả các giá trị từ 1$ đến 7$ (cũng như 9$). 7 là giá trị cực đại, do đó n(3,2)=7.

Thật không may là công thức tính n(h,k) từ h,k cũng như mệnh giá của các con tem đã bị thất lạc. Giờ đây cho các giá trị của h và k, xác định một tập hợp tối ưu mệnh giá các con tem và giá trị n(h,k).

INPUT

INPUT bao gồm nhiều dòng, mỗi dòng chứa một giá trị h và k. Input kết thúc bằng hai số không (0 0). Vì lý do kỹ thuật, tổng h và k không vượt quá 9. (Tổng thống bị mất ngón tay út của mình trong một tai nạn bắn súng và không thể đếm quá 9).

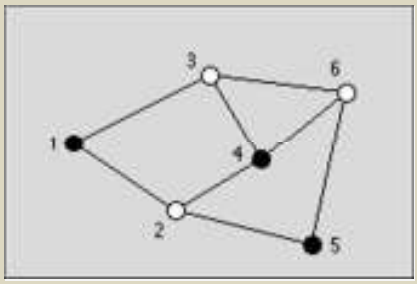
OUTPUT

Với mỗi bộ giá trị của h và k, xuất ra giá trị mệnh giá các con tem theo thứ tự tăng dần, sau đó là một khoảng trống và một mũi tên (->) và giá trị của n (h,k).

Sample Input	Sample Output
3 2 0 0	1 3 -> 7

193. ĐỒ THỊ MÀU

Bạn phải viết một chương trình tô màu một đồ thị cụ thể. Mỗi đỉnh của đồ thị phải được tô bằng màu đen hoặc màu trắng. Tuy nhiên 2 đỉnh liền kề (có chung một cạnh) không được có cùng màu đen. Màu của đồ thị được gọi là tối ưu nếu số đỉnh có màu đen là cực đại.



Một đồ thị tối ưu với ba nút đen

INPUT

Dữ liệu về đồ thị là tập hợp của các nút biểu thị bằng các con số từ 1 đến n, và tập hợp các cạnh vô hướng biểu hiện bằng các cặp đỉnh nối. Input có m đồ thị. Số m nằm ở dòng đầu tiên. Với mỗi đồ thị, dòng đầu tiên ghi 2 số n và k biểu diễn số lượng nút và số lượng cạnh. k dòng là hai số nguyên cách nhau bởi một dấu cách, mô tả 2 đỉnh có 1 cạnh nối.

OUTPUT

In ra 2m dòng, mỗi đồ thị trong input ứng với hai dòng. Dòng đầu tiên chứa số lượng tối đa của các nút có thể được tô màu đen. Dòng thứ hai là danh sách các đỉnh được tô bằng màu đen. Các số cách nhau một dấu cách

Sample Input	Sample Output
1 6 8 1 2 1 3 2 4 2 5 3 4 3 6 4 6 5 6	3 1 4 5

208. XE CHỮA CHÁY

Ban PCCC của Thành phố phối hợp với Bộ GTVT để bảo trì các bản đồ của thành phố phản ánh hiện trạng đường phố của thành phố. Ở bất cứ ngày nào, một vài con đường được đóng lại để sửa chữa hoặc xây dựng. Nhân viên cứu hỏa cần có thể lựa chọn các tuyến đường từ các trạm chữa cháy đến đám cháy mà không đi qua tuyến đường bị đóng.

Thành phố được chia thành các khu vực không giao nhau, mỗi khu vực chứa một trạm chữa cháy. Khi một đám cháy được báo, trung tâm thông tin báo động cho trạm chữa cháy ở khu vực có đám cháy, và cho một danh sách các tuyến đường từ trạm chữa cháy đến đám cháy. Bạn phải viết một chương trình mà trung tâm thông tin có thể sử dụng để biết được các tuyến đường từ trạm chữa cháy đến đám cháy.

INPUT

Thành phố có một bản đồ riêng biệt cho từng khu vực. Hai đầu mút của con đường là hai điểm được xác định bởi một số nguyên dương nhỏ hơn 21, trạm chữa cháy luôn ở điểm số 1. Dữ liệu vào chứa một vài test đại diện cho các đám cháy khác nhau của các khu vực khác nhau.

Dòng đầu tiên của mỗi test gồm có một số nguyên là số điểm gần đám cháy.

Các dòng tiếp theo gồm một cặp số nguyên dương được ngăn các bởi khoảng trống, cặp số này là hai điểm liền nhau của một con đường mở (hai mút của con đường). (Ví dụ, cặp 4 7 nằm trên cùng một dòng của dữ liệu vào, thì con đường giữa điểm 4 và điểm 7 được thông. Không có điểm nào nằm giữa 4 và 7 trên đoạn đường này.)

Dòng cuối cùng của mỗi test là một cặp 0 0.

OUTPUT

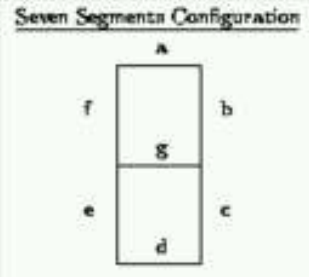
Với mỗi test, đầu ra phải xác định trường hợp bởi một số (`CASE#1, CASE#2, …`). Phải liệt kê mỗi tuyến đường trên một dòng phân biệt. Và phải ghi tổng số tuyến đường từ trạm chữa cháy đến đám cháy. Chỉ bao gồm tuyến đường mà một điểm không bao giờ bị đi qua hai lần (rõ ràng cục PCCC không muốn xe của họ phải đi vòng tròn!)

Đầu ra trong mỗi trường hợp khác nhau phải xuất hiện trên những dòng khác nhau.

Sample Input	Sample Output
6 1 2 1 3 3 4 3 5 4 6 5 6 2 3 2 4 0 0 4 2 3 3 4 5 1 1 6 7 8 8 9 2 5 5 7 3 1 1 8 4 6 6 9 0 0	CASE 1: 1 2 3 4 6 1 2 3 5 6 1 2 4 3 5 6 1 2 4 6 1 3 2 4 6 1 3 4 6 1 3 5 6 There are 7 routes from the firestation to streetcorner 6. CASE 2: 1 3 2 5 7 8 9 6 4 1 3 4 1 5 2 3 4 1 5 7 8 9 6 4 1 6 4 1 6 9 8 7 5 2 3 4 1 8 7 5 2 3 4 1 8 9 6 4 There are 8 routes from the firestation to streetcorner 4.

416. LED TEST

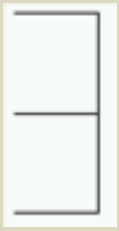
Có nhiều thiết bị hiển thị chữ số bằng mạng lưới các đèn LED. Một chữ số thông thường được hiện lên bằng 7 thanh LED, với cách sắp xếp và đặt tên như sau:



Các chữ số được hiển thị bằng cách làm sáng một số thanh LED theo như bảng sau:

Digit Displayed	Segments Illuminated (Y-Yes, N-No)						
	a	b	c	d	e	f	g
0	Y	Y	Y	Y	Y	Y	N
1	N	Y	Y	N	N	N	N
2	Y	Y	N	Y	Y	N	Y
3	Y	Y	Y	Y	N	N	Y
4	N	Y	Y	N	N	Y	Y
5	Y	N	Y	Y	N	Y	Y
6	Y	N	Y	Y	Y	Y	Y
7	Y	Y	Y	N	N	N	N
8	Y	Y	Y	Y	Y	Y	Y
9	Y	Y	Y	Y	N	Y	Y

Ví dụ, số 3 sẽ được hiển thị bằng cách làm sáng đèn `{a,b,c,d,g}` như sau:



Vấn đề dưới đây xuất phát từ hiện tượng xảy ra trong cuộc kiểm tra một số bộ phận mới, trong thiết bị sản xuất lò vi sóng. Các thanh LED được kiểm tra để đảm bảm nó hoạt động đúng chức năng và được kiểm soát bởi các thiết bị tự động.

Chú ý: Giải pháp của bài toán này sẽ không hoàn chỉnh cũng như không cần thiết các test có ích cho sự kiểm tra thực tế sẽ diễn ra.

Bạn cần tạo một chương trình theo dõi lần lượt các điều kiện phát sáng của 7 thanh LED của một chữ số, và quyết định xem sự hiển thị có phải là một thứ tự đếm ngược hay không.

Thật không may, các đèn LED mà chương trình của bạn theo dõi, một số các thanh LED có thể bị cháy khi bắt đầu thử nghiệm, và các thanh LED khác có thể bị cháy trong thời gian thử nghiệm. Sẽ không có thanh LED nào cháy khi đang sáng, thanh LED đã cháy không thể phục hồi. Tuy nhiên, bạn có thể "đọc" những thứ không rõ ràng trên màn hình bị trục trặc, bằng cách xem xem màn hình đếm ngược một vài lần.

INPUT

Input chứa một dãy tập hợp dữ liệu có dạng như sau:

Dòng đầu tiên - Một số nguyên dương N < 11. Số này được đặt tại lề trái của dòng.

N dòng tiếp theo - Mỗi dòng chữa một chuỗi 7 chữ cái Y và N liền nhau chỉ ra màn hình LED đang được hiển thị. Các dòng liên tiếp này được cho là đại diện cho một tuần tự đếm ngược. Kí tự đầu trong 7 kí tự này, đại diện cho thanh LED "a", sẽ là kí tự đầu tiên trong dòng.

Kết thúc dữ liệu là một nhóm dữ liệu rỗng có số 0 trên dòng đầu và không có bất cứ dữ liệu nào khác.

Không có thông tin nào về việc trình tự đếm ngược bắt đầu từ đâu; mỗi nhóm đại diện cho một màn hình LED khác nhau đang được kiểm tra.

OUTPUT

Đối với mỗi trình tự được cho là đếm ngược, bạn nhận được một nhóm dữ liệu, bạn phải trả lời: MATCH hoặc MISMATCH

Phụ thuộc vào việc đây có phải là một trình tự đếm ngược hợp lệ - một trình tự nào đó nằm trong trình tự {9,8,7,6,5,4,3,2,1,0} hay không; xem có số nào không rõ do bị cháy đèn, trước và trong khi kiểm tra. Output phải ở đầu dòng.

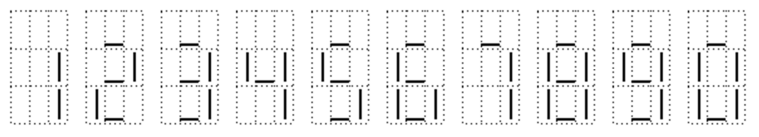
Sample Input	Sample Output
1 YYYYNYY 2 NNNNNNN NNNNNNN 2 YYYYYYY YYYYYYY 3 YNYYYYY YNYYNYY NYYNNYY 3 YNYYYYN YNYYNYN NYYNNYN 3 YNYYYYN YNYYNYN NYYNYYN 4 YYYYYYY NYYNNNN NNYYYYN NNNYNNN 3 NNNNNNN YNNNNNN NNNNYNN 0	MATCH MATCH MISMATCH MATCH MATCH MISMATCH MATCH MATCH

433. NGÂN HÀNG

Các ngân hàng luôn cố gắng cải thiện lợi ích mà mình đem tới cho khách hàng, đòi hỏi máy tính phải chuyên nghiệp hơn để tiến đến một hệ thống mà có thể đọc ngân phiếu. Một ý tưởng trong số đó là sử dụng một chương trình nhận dạng kí tự quang học (viết tắt là OCR) để quản lí tài khoản ngân hàng, điều này sẽ làm giảm chi phí xử lý các tấm séc. Thiết bị này in ra 7 phân đoạn để biểu thị một chữ số.

Một khi một tấm séc được quét, một số hình ảnh được phần mềm xử lý sẽ chuyển các đường ngang và dọc thành dạng mã ASCII với các dấu '|' và gạch dưới '_'

Phiên bản 7 phân đoạn một ASCII của 10 chữ số được mô tả như sau:



Một tài khoản ngân hàng có 9 chữ số cùng với một quy định. Đối với một dãy số tài khoản hợp lệ, ta phải có: $(d_1 + 2*d_2 + 3*d_3 + ... + 9*d_9) \bmod 11 = 0$. Các chữ số được đếm từ phải qua trái như sau: $d_9d_8d_7d_6d_5d_4d_3d_2d_1$.

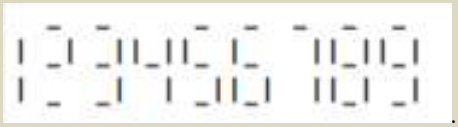
Không may thay, thỉnh thoảng máy quét nhầm lẫn: một vài phân đoạn không được phát hiện. Nhiệm vụ của bạn là viết một chương trình tìm lại số tài khoản gốc, với giả định sau:

Khi input là một số tài khoản hợp lệ, đó chính là số tài khoản gốc;

Nhiều nhất một chữ số bị sai lệch;

Ảnh được quét không có phân đoạn mở rộng hay phân đoạn khác.

Ví dụ, input sau thể hiện số 123456789



INPUT

Tệp tin Input bắt với một dòng chưa một số nguyên thể hiện số lượng số tài khoản cần được xử lý. Mỗi số tài khoản chiếm 3 dòng, mỗi dòng 27 kí tự.

OUTPUT

Với mỗi bộ test, output chứa 9 chữ số thể hiện số tài khoản chính xác mà có thể nhận ra, in ra "failure" nếu không có giải pháp và "ambiguous" nếu có nhiều hơn một giải pháp được tìm thấy.

Sample Input	Sample Output

4
 	
  _ _ _ _ _ _ _
| _| _||_||_ |_ ||_||_|
| _ _| | _||_| ||_| _|
 	
  _ _ _ _ _ _ _
|_||_|| || ||_ | | ||_
  | _||_||_||_| | | | _|
 	
_ _ _ _ _ _ _ _ _
|_||_||_||_||_||_||_||_||_|
|_||_||_||_||_||_||_||_|
 	
_ _ _ _ _ _ _ _
|_| ||_||_||_||_||_||_|
|_| ||_||_||_||_||_||_|
	123456789
	ambiguous
	failure
	878888888

565. PIZZA ANYONE

Bạn chịu trách nhiệm đặt bánh pizza cho lớp. Mỗi bạn trong lớp thông báo cho bạn về loại phủ mình thích và loại phủ mình không thích trên pizza. Dĩ nhiên mọi người đều biết rằng vì có đông người, nên không thể nào có được chiếc pizza thỏa mãn yêu cầu của tất cả mọi người. Vấn đề là bạn nên đặt 1 chiếc pizza nào để thỏa mãn ít nhất một trong các yêu cầu của bất kỳ người bạn nào.

Có các loại phủ sau đây, bạn có thể yêu cầu thêm vào hay bỏ ra bất kỳ loại phủ nào:

Mã	Loại phủ
A	Anchovies
B	Black Olives
C	Canadian Bacon
D	Diced Garlic
E	Extra Cheese
F	Fresh Broccoli
G	Green Peppers
H	Ham
I	Italian Sausage
J	Jalapeno Peppers
K	Kielbasa
L	Lean Ground Beef
M	Mushrooms
N	Nonfat Feta Cheese
O	Onions
P	Pepperoni

Bạn của bạn đưa cho bạn yêu cầu của mình, yêu cầu là 1 dòng văn bản mô tả họ thích và không thích loại phủ nào. Ví dụ dòng +O-H+P nói rằng người bạn đó thích ăn pizza có Onion, không thích loại có ham hay loại có pepperoni. Và dòng -E-I-D+A+J chỉ ra rằng ai đó sẽ chấp nhận loại pizza không có extra cheese, hoặc Italian sausage, hoặc diced garlic, hoặc người đó chấp nhận loại pizza có anchovies or jalapenos.

INPUT

The input có nhiều test. Mỗi test nằm trên nhiều dòng. Các test cách nhau bằng một dòng có 1 dấu chấm. Mỗi test chứa nhiều yêu cầu từ các bạn. Số lượng yêu cầu nhỏ hơn 12. Mỗi yêu cầu nằm trên 1 dòng, kết thúc yêu cầu là dấu ; Mỗi yêu cầu là các cụm (dấu)(chữ cái) liên tiếp nhau. Dấu là + hoặc -. Chữ cái từ A đến P.

OUTPUT

Với mỗi test, in ra loại phủ thỏa mãn yêu cầu đề bài. Mở đầu mỗi dòng ghi dòng chữ ``Toppings: " (10 ký tự). Sau đó liệt kê các loại phủ trên pizza đó (theo thứ tự từ điển). Ví dụ pizza có onion, anchovies, fresh broccoli and Canadian bacon được ghi như sau:

Toppings: ACFO

Nếu không có, in ra dòng

No pizza can satisfy these requests.

Sample Input	Sample Output
+A+B+C+D-E-F-G-H; -A-B+C+D-E-F+G+H; -A+B-C+D-E+F-G+H; . +A+B+C+D; +E+F+F+H; +A+B-G;	Toppings: Toppings: CELP No pizza can satisfy these requests.

+O+J-F; +H+I+C; +P; +O+M+L; +M-L+P; . +A+B+C+D; +E+F+F+H; +A+B-G; +P-O; +O+J-F; +H+I+C; +P; +O; +O+M+L; -O-P; +M-L+P; .	

703. TRIPLE

N người tham gia một cuộc đấu cờ, bất kỳ hai người nào cũng đấu với nhau. Tất cả các kết quả được ghi vào ma trận theo cách thức sau đây:

Nếu người i thắng người j thì ô (i,j) có giá trị 1 - ở đây i là số hàng và j là số cột; và ô (j,i) có giá trị 0..

Nếu người i hòa người j thì ô (i,j) và ô (j,i) đều có giá trị 0.

Các ô ở đường chéo chính có giá trị 0.

Để ra kết quả cuối cùng, ban tổ chức gặp vấn đề rắc rối liên quan đến có một nhóm người có điểm ngang bằng nhau (ngang cơ). Quy tắc chung để xử lý phân loại một nhóm như thế là chỉ xét tất cả các trận đấu giữa các thành viên trong nhóm này. Tuy nhiên, đôi khi quy tắc này không giải quyết được vấn đề. Chẳng hạn, khi nhóm có 2 người và 2 người hòa nhau. Với nhóm 3 người thì có 2 trường hợp không phân loại được:

Nếu cả 3 trận đều hòa.

Nếu có chu trình: người 1 thắng người 2, người 2 thắng người 3, người 3 thắng người 1.

Vì ban tổ chức có thể sẽ thiết lập thêm tiêu chí phân loại khác, như theo tuổi, quốc tịch của người tham gia. Ban tổ chức cần tạo ra 1 bảng chứa tất cả các nhóm 3 người "ngang cơ". Nếu cả 3 trận đấu đều hòa, in ra số thứ tự của 3 người theo trật tự tăng dần, nếu có chu trình, thì in theo thứ tự: người thứ 1 thắng người thứ 2, người thứ 2 thắng người thứ 3. Trong 3 khả năng có thể xảy ra, chọn dòng kết quả được viết theo thứ tự tăng hoặc giảm. Nếu có nhiều chuỗi 3 số thì chúng được liệt kê theo thứ tự từ điển.

INPUT

Có nhiều test. Dòng đầu tiên trong mỗi test ghi một số nguyên dương *N* là số lượng người chơi và *N* là ma trận kết quả ($3 \le N \le 100$).

OUTPUT

Với mỗi test, in ra số lượng nhóm 3 người "ngang cơ" nhau. Nếu M>0, M dòng tiếp theo liệt kê M nhóm 3 người ngang cơ, số thứ tự của mỗi người cách nhau bởi dấu cách. Chú ý thứ tự yêu cầu trong mỗi dòng và giữa các dòng.

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

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

868. MÊ CUNG SỐ

Không ai biết tác giả của các mê cung đầu tiên (mê đạo), nhưng người ta phát hiện ra rất nhiều loại mê cung ở khắp mọi nơi và một số mê cung đã được xây dựng từ thời cổ xưa. Các mê cung nổi tiếng nhất có lẽ là ở đảo Crete, do Daedalus vẽ và xây dựng cho vua Minos, để giam giữ Minotaur (một con quái vật mình người đầu bò). Theo truyền thuyết Theseus sẽ tiêu diệt được con quái vật này.

Hình dưới đây minh họa một loại mê cung

Bạn có thể tìm thấy một con đường bắt đầu từ điểm xuất phát giúp đưa bạn thoát khỏi mê cung? Bạn chỉ được phép di chuyển theo chiều ngang hoặc theo chiều dọc, không được phép đi chéo. Con đường này bao gồm các chuỗi tuân theo các quy tắc sau: 1; 1,2; 1,2,3; 1,2,3,4, …. Có thể thay đổi hướng khi di chuyển.

Vấn đề bạn phải xác định điểm xuất phát và đường dẫn sẽ đưa bạn thoát khỏi mê cung. Điểm bắt đầu luôn luôn là một ô trong các hàng phía trên của mê cung (với giá trị 1) và điểm thoát luôn là một ô nào đó trong hàng cuối cùng của mê cung.

INPUT

INPUT bắt đầu với một dòng ghi số lượng test, sau đó là một dòng trống. Sau đó là dữ liệu về mỗi test. Mỗi test cách nhau một dòng trống.

Dòng đầu mỗi test chứa hai số nguyên dương N và M là số lượng hàng và cột của mê cung. N dòng tiếp theo, mỗi dòng chứa M số cách nhau bằng dấu cách. Giá trị mỗi ô lớn hơn hoặc bằng 1.

OUTPUT

Đối với mỗi trường hợp thủ nghiệm, OUTPUT phải thực hiện theo các mô tả dưới đây. Giữa kết quả của hai test liên tiếp là một dòng trống.

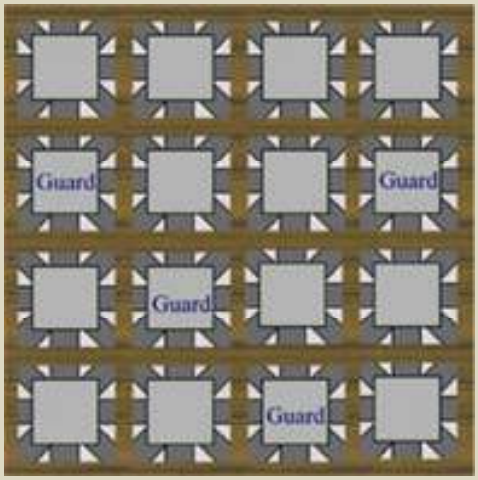
Dòng đầu tiên chứa tọa độ (hàng và cột) của ô xuất phát và dòng thứ hai là tọa độ của ô thoát. Nếu có nhiều giải pháp, in ra điểm nhỏ nhất theo thứ tự từ điển. Nếu có nhiều đường đi, in ra điểm kết thúc có tọa độ nhỏ nhất theo thứ tự từ điển.

Sample Input	Sample Output
1	1 6

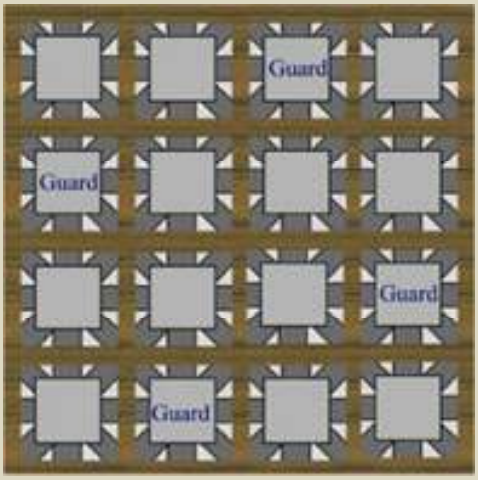
10 11 1 6 5 2 1 1 2 3 2 1 4 1 2 6 3 2 1 1 3 4 5 6 1 2 3 2 1 3 2 5 6 4 2 2 3 1 2 2 3 3 4 5 2 1 3 4 2 3 4 5 3 2 1 4 2 4 3 4 4 5 6 4 3 2 5 3 5 4 2 1 2 3 4 4 3 6 4 6 5 3 2 3 4 5 5 4 1 1 1 6 4 3 5 5 6 6 1 2 3 2 1 5 1 6 6 1 2 2 3 4	10 3

10094. VỊ TRÍ BẢO VỆ

Vua của vùng đất Amazing có một mật thất mà ngài đặt kho báu. Không ai biết kích thước của mật thất. Một số người cho rằng, đó là **4 x 4**, một số người lại cho rằng đó là **1000 x 1000**. Mỗi căn phòng trong mật thất có hình vuông, với kích thước cạnh là môt đơn vị độ dài. Mật thất **4 x 4** có **16** phòng trong hình bên dưới. Và mật thất **n x n** có **n²** phòng. **n** lính gác quan sát mật thất kích thước **n²** . Trong bức tranh bên dưới, mỗi hình vuông lớn thể hiện một phòng. Hình vuông xám bên trong mỗi phòng cho biết khoảng trống nào mà lính bảo vệ đứng. Đường hầm (màu xám đậm) từ các khoảng trống biểu thị đưuòng hầm đi vào và đi ra các căn phòng. Như ta thấy, các đường hầm được thiết kế theo cách : lính bảo vệ có thể nhìn thấy các ô trên những ô cùng hàng, cùng cột, hoặc cùng đường chéo. Nếu ta kí hiệu **(hàng, cột)** để biểu diễn thì nhìn vào **hình 1**, chúng ta có thể nói rằng lính ở **(2,1)** có thể nhìn thấy lính ở **(3,2)** và lính **(3,2)** có thể nhìn thấy lính **(4,3).** Mặc dù lính ở **(2,1)** và lính ở **(4,3)** cùng nằm trên đưuòng chéo nhưng họ không nhìn thấy nhau vì bị lính **(3,2)** che khuất. Vì một lí do rõ rằng, lính ở **(2,1)** có thể nhìn thấy lính ở **(2,4)** nhưng lính ở **(3,2)** không thể nhìn thấy lính ở **(2,4).** Nhà vua luôn sắp xếp vị trí các lính gác theo nguyên tắc : *không lính nào có thể thấy lính khác*. Trong một căn phòng trống (không có lính gác), nhà vua đặt kho báu. Nhà vua làm vậy vì ngài nghĩ rằng : khi một lính nhì thấy lính khác, họ sẽ nói chuyện, buôn dưa lê và mất tập trung. Bạn hãy giúp đỡ nhà vua sắp xếp vị trí các lính.



Hình 1: Cách xếp không hợp lệ



Hình 2 : Cách xếp hợp lệ

INPUT

File đầu vào sẽ chữa mỗi số nguyên một dòng, biểu thị cho số n (độ dài một cạnh của mật thất). Nhớ rằng 1< n < 1001. Dữ liệu vào kết thúc bởi EOF.

OUTPUT

Rõ ràng rằng mỗi cột chỉ có một lính. Đối với mỗi giá trị n, bạn sẽ phải in một dòng có n số nguyên. Các số nguyên được ngăn cách bởi một dấu cách chữ. Các số nguyên này miêu tả hàng mà lính đang đứng của mỗi cột. Ví dụ như Hình 2, bạn sẽ phải in ra: 2 4 1 3. Có thể sẽ có nhiều phương án. Mọi phương án tốt đều được chấp nhận. Nếu lính không thể xếp trong mật thất thì in ra: “Impossible” trên một dòng.

–

Sample Input	Sample Output
4 8 10	2 4 1 3 4 6 8 2 7 1 3 5 2 4 6 8 10 1 3 5 7 9

10309. TẮT ĐÈN

Vì chúng ta đều sinh sống trên mẹ Trái Đất, chúng ta phải có trách nhiệm bảo vệ. Vì vậy, bạn được yêu cầu để tiết kiệm năng lượng bằng cách tắt đèn.

Bạn có người bạn đang gặp vấn đề sau. Có một lưới **10x10**, mỗi hình vuông có một bóng đèn và một công tắc để tác động đến nó. Không may thay, những chiếc đèn không hoạt động như người ta thường nghĩ. Mỗi khi một công tắc được ấn, không chỉ chiếc đèn ở đó *chuyển*, mà những cái bên trái, phải, trên, dưới cũng *chuyển*! Tất nhiên, nếu đèn ở góc lưới, có ít đèn bị *chuyển* hơn.

Khi một đèn được *chuyển* có nghĩa là nó sẽ bật nếu lúc trước nó tắt, và nó sẽ tắt nếu lúc trước nó bật. Xem ví dụ sau, đây chỉ là một vùng nhỏ của cả lưới. Họ đang xem chuyện gì sẽ xảy ra nếu đèn giữa được *chuyển*. “O” có nghĩa là đèn đã bật, “#” có nghĩa là đèn tắt.

```
###    #O#
### -> OOO
###    #O#

###    #O#
OOO -> ###
###    #O#
```

Người bạn của bạn thích tiết kiệm năng lượng và yêu cầu bạn viết một chương trình phát hiện xem có thể tắt hết các đèn hay không, và nếu có thể thì cho biết anh ta phải nhấn công tắc bao nhiêu lần để tắt tất cả các đèn.

INPUT

Có một vài test. Mỗi test được bắt đầu bằng một từ duy nhất biểu thị tên của test. Sau tên là 10 dòng, mỗi dòng chứa 10 kí tự "#" hoặc "O". Kết thúc Input là một xâu kí tự "end".

OUTPUT

Với mỗi test, in ra một dòng chứa tên test, một dấu cách, và số lần ít nhất mà anh bạn của bạn phải nhấn vào các công tắc. Nếu không thể tắt đèn thì ghi ra tên test, một dấu cách, và -1.

Sample Input	Sample Output
all_off ########## ########## ########## ########## ########## ########## ##########	all_off 0 all_on 44 simple 4

########## ########## ########## all_on OOOOOOOOOO OOOOOOOOOO OOOOOOOOOO OOOOOOOOOO OOOOOOOOOO OOOOOOOOOO OOOOOOOOOO OOOOOOOOOO OOOOOOOOOO simple #O######## OOO####### #O######## ####OO#### ###O##O### ####OO#### ########## ########O# #######OOO ########O# end	

10582. ASCII Labyrinth

Ta đang thử xây dựng một mê cung trên một bảng kích thước *m x n*. Ban đầu, trên mỗi hình vuông, ta có các miếng ghép *1 x 1*, trên miếng ghép đó, có ba loại hình dáng sau đây được vẽ trên nó.

Khi xây dựng mê cung, ta có thể xoay các miếng ghép nhưng mỗi miếng ghép phải bao phủ đúng một hình vuông trong bảng. Ta không được di chuyển một miếng sang ô lưới khác.

Cho sẵn một bảng gồm các miếng ghép, ta phải xoay các miếng ghép theo hướng : ít nhất có một đường nối từ góc trái trên tới góc phải dưới. Hình dưới đây mô tả trạng thái ban đầu của bảng 4x6 và một mê cung đã xây dựng từ bảng này.

Việc của bạn là đọc các thông tin ban đầu của bảng với các miếng ghép nằm trên đó, và chỉ ra xem liệu có thể xoay các miếng ghép để hình vẽ trên đó hợp thành một đường nối từ ô góc trái trên xuống ô góc phải dưới của bảng hay không.

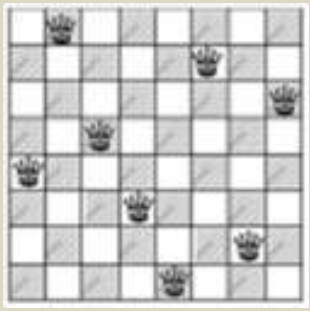
Dòng đầu tiên của input chứa số c, cho biết số trường hợp cần xét. Dữ liệu cho mỗi trường hợp bắt đầu với hai số nguyên m,n cho biết có bao nhiêu hàng và bao nhiêu cột trong bảng. Các dòng còn lại thể hiện bảng ban đầu bằng mã ASCII. Các dòng đó dùng các kí hiệu +,-,|,* và khoảng trống để kí hiệu. Xem mẫu input để biết thêm về cấu tạo. Kích thước của bảng thoả mãn $m \times n \leq 64$.

Với mỗi trường hợp, in ra một dòng, chỉ ra có bao nhiêu đường giải quyết vấn đề trên.

Sample Input	Sample Output
1 4 6 +---+---+---+---+---+---+ | | | | | | | |***|***|** |** |** |***| | | | * | * | * | | +---+---+---+---+---+---+ | | | | | | | | |** |** |***|** |** | | | * | * | | * | * | +---+---+---+---+---+---+ | | | | | | | |***|** |***|***|***|** | | | * | | | | * | +---+---+---+---+---+---+ | | | | | | | |** | |***| |** |** | | * | | | | * | * | +---+---+---+---+---+---+	Number of solutions: 2

11195. LẠI HẬU

Trong bài toán N-Queen này, bạn sẽ phải đếm số cách xếp n con hậu trên bàn cờ kích thước n × n. Trong bàn cờ, có 1 số ô xấu – không được đặt hậu vào các ô xấu. Chú ý các quân hậu vẫn có thể xuyên qua ô xấu để tấn công các quân khác. Chú ý 2 giải pháp dù có thể thu được từ nhau qua phép quay vẫn tính là khác nhau. Như vậy có 92 giải pháp cho bài toán 8-Queen.



INPUT

Có tối đa 10 test. Trong mỗi test, dòng đầu tiên là một số nguyên n ($3 < n < 15$) . Sau đó là n dòng, mỗi dòng có n ký tự biểu diễn bàn cờ. Ô trống là ký tự chấm (.). Ô xấu là ký tự sao (*). Test cuối có n=0 và không phải xử lý.

OUTPUT

Với mỗi test, in ra số thứ tự và đáp án.

Sample Input	Sample Output
8	Case 1: 92 Case 2: 1

…….. …….. 4 .*.. …. …. …. 0	