ACM Regional Programming Contest 1999

Problem No. 1 - Crossword
Executable Program: PROG1.EXE, PROG1.CLASS
Source Program: PROG1.CPP, PROG1.C, PROG1.JAVA, PROG1.PAS
Input file: PROG1.IN **Output file**: PROG1.OUT

A crossword can be stored as a matrix *m* x *n* of zeros and ones. Zero represents white squares and ones represents black squares. Some squares of the crossword are numbered and assigned to these numbers are the descriptions of the words that should be written either “across” or “down” into the crossword. A square is numbered if it is a white square and either (a) the square below it is white and there is no white square inmediately above, or (b) there is no white square inmediately to its left and the square to its right es white. Appropriate squares are numbered from left to right, from de top line to the bottom line.

From de matrix a crossword can be drawn. In the diagram each square is represented by a box 4 x 6 characters. Black square is represented by

```
******
******
******
******
```

White squares are represented as follows (numbered and not numbered square)

```
******          ******
*nnn *          *    *
*    *          *    *
******          ******
```

The remaining characters of the box are spaces. If black squares are given at the edges, they should be removed from the diagram (see the example). Only use spaces as necesary filling characters. Don't use any unnecessary spaces at the end of the line.

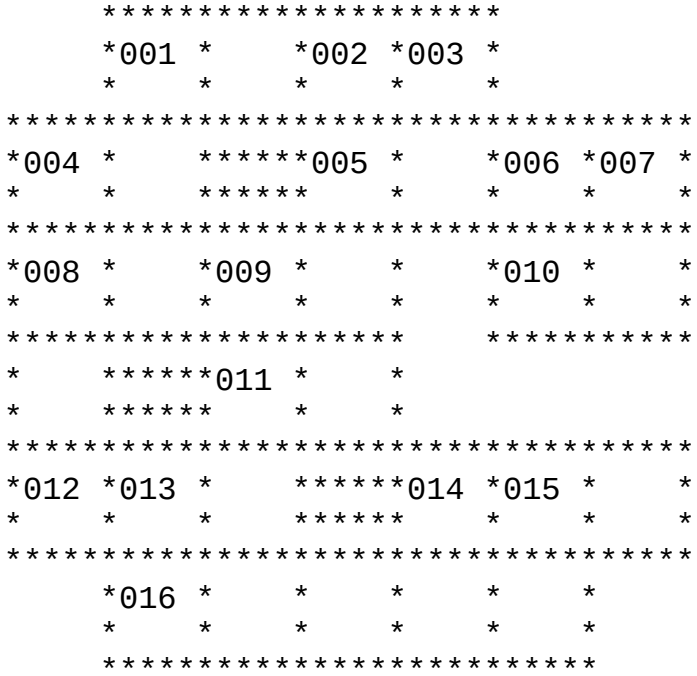
Input
The input consists of one block of lines representing a crossword. The first line containing two integers $m < 15$ and $n < 15$ separated by one space. In each of the next *m* lines there are *n* numbers 0 or 1, separated by one space.

Output
The output file contains the corresponding crossword diagram.

Example Input file

6 7
1 0 0 0 0 1 1
0 0 1 0 0 0 0
0 0 0 0 1 0 0
0 1 0 0 1 1 1
0 0 0 1 0 0 0
1 0 0 0 0 0 1

Exaample Output file



ACM Regional Programming Contest 1999

Problem No. 2 - Joseph's Problem
Executable Program: PROG2.EXE, PROG2.CLASS
Source Program: PROG2.CPP, PROG2.C, PROG2.JAVA, PROG2.PAS
Input file: PROG2.IN **Output file**: PROG2.OUT

The Joseph's problem is notoriously known. For those who are not familiar with the problem : from among *n* people, numbered 1, 2, ..., *n*, standing in circle every *m*th is going to be executed and only the life of the last remaining person will be saved. Joseph was smart enough to choose the position of the last remaining person, thus saving his life to give the message about the incident. For example when n=6 and m=5 then the people will be executed in the order 5, 4, 6, 2, 3 and 1 will be saved.

Suppose that there are *k* good guys and *k* bad guys. In the circle the first *k* are good guys and the last *k* bad guys. Ou have to determine such minimal *m* that all the bad guys will be executed before the first good guy.

Input
The input consists of separates lines containing *k*. The last line in the input line contains 0. You can suppose that $0<k<10$

Output
The output file consist of separate lines containing *m* correspondig to *k* in the input file

`Example Input file`
3
4
0
Examples Output file
5
30

ACM Regional Programming Contest 1999

Problem No. 3 - Word Match
Executable Program: PROG3.EXE, PROG3.CLASS
Source Program: PROG3.CPP, PROG3.C, PROG3.JAVA, PROG3.PAS
Input file: PROG3.IN **Output file**: PROG3.OUT

As a secret service agent, you monitor all computer email traffic, searching for keywords that suggest subversive activities. However, some subversive elements in society have started scrambling the letters in some words of their email messages in an attempt to evade detection.
You, undeterred by such transparent tricks, decide to write a program that searches sentences for suspicous words whose letters have been rearranged.

Input

The input contains pairs of lists and sentences. The list gives the words to search for in the following sentence. All letters in the input are in lower case. Words are at most 20 letters long and are seperated by a single space. Each line is at most 80 characters long. Each list fits on a single line and contains at most 20 words. Sentences may occupy more than one line, and are terminated by a full stop.
The end of input is denoted by a list containing only a #.

Output

For each pair of list and sentence, find the words from the list that appear in the sentence with their letters shuffled. Print these words in the order they appear in the list, separated by a single space. If no words from the list match the words in the sentence, just print a blank line.

Input File Example

```
bomb
we twan ot ombb mericaa.
guns mines missiles
aameric ssell snug
dan iimsssle ot sit neeemis.
cat sat mat
cat sat tam mat.
#
```

Output File Example

```
bomb
guns missiles
cat sat mat
```

ACM Regional Programming Contest 1999

Problema No. 4 - Divisible por 11
Programa Ejecutable: PROG4.EXE, PROG4.CLASS
Programa Fuente: PROG4.CPP, PROG4.C, PROG4.JAVA, PROG4.PAS
Archivo de entrada: PROG4.IN **Archivo de salida**: PROG4.OUT

Escriba un programa que acepte como entrada un entero positivo y usando el algoritmo escrito abajo verifique si el entero es o no divisible por 11

Algoritmo:

Mientras el número que se prueba tenga más de dos dígitos, forme un nuevo número

- Borrando el dígito de las unidades
- Restando el dígito borrado del número truncado

El número original es divisible por 11 si y solo si, el número restante final es divisible por 11

Los ceros a la izquierda no son considerados parte del número y no deben ser impresos.

Como de costumbre, el primer número de la entrada indica el número de enteros positivos que siguen. Cada entero positivo tiene un máximo de 50 dígitos. Se puede asumir que no hay 0 a la izquierda en los enteros positivos. Por cada entero positivo en la entrada, la salida consiste de una serie de números formados cuando el último dígito es borrado y restado al número truncado, seguido por un mensaje que indica si el número original es o no divisible por 11. Las salidas para diferentes enteros positivos son separadas por líneas en blanco.

Ejemplo del archivo de entrada

```
2
12345678901234567900
896245630004
```

Ejemplo del archivo de salida

```
12345678901234567900
1234567890123456790
123456789012345679
12345678901234558
1234567890123447
123456789012337
12345678901226
1234567890116
123456789005
12345678895
1234567884
123456784
12345674
1234563
123453
12342
1232
121
11
El numero 12345678901234567900 es divisible por 11.

896245630004
89624562996
8962456293
896245626
89624556
8962449
896235
89618
8953
892
87
El numero 896245630004 no es divisible por 11.
```

ACM Regional Programming Contest 1999

Problema No. 5 - Estrellas
Programa Ejecutable: PROG5.EXE, PROG5.CLASS
Programa Fuente: PROG5.CPP, PROG5.C, PROG5.JAVA, PROG5.PAS
Archivo de entrada: PROG5.IN **Archivo de salida**: PROG5.OUT

Trabajas para los Laboratorios de Propulsión por Reacción Sputnik. En este momento es necesario que escribas un programa que lea una matriz no circular, la cual contiene una representación digitalizada de una fotografía del cielo nocturno.

Cada elemento de la matriz representa la cantidad de luz que existe en determinada región de la imagen digitalizada. El rango de intensidad va de 0 a 20. El programa permitirá localizar las regiones donde se ubica una estrella, partiendo de la siguiente información:

Una estrella se encuentra en el área cubierta por el elemento i,j de la matriz si se cumple la siguiente condición:

(MD(i,j) + suma de intensidades circundantes)/5 > 10.0

Donde MD representa la matriz digitalizada.

Entrada

La primera línea son dos enteros menores que 10 que indican el número de filas y columnas de la matriz,las siguientes líneas contienen enteros que representan los de cada fila de la matriz de intensidad, separados por un blanco.

Salida

La salida deseada es la matriz que contiene un asterisco en la posición donde esta localizada una estrella, y un blanco donde no la hay. La matriz debe estar circundada por un borde que indique las coordenadas de cada estrella. La coordenada inicial es 1.

Ejemplo del archivo de entrada

```
6 8
0 3 4 20 15 0 6 8
5 13 6 8 2 0 2 3
2 6 2 2 3 0 10 0
0 0 4 15 4 1 1 20
0 0 7 2 6 9 10 4
5 0 6 10 6 4 8 0
```

Ejemplo del archivo de salida

```
 12345678
1  **    1
2  **    2
3  *     3
4      * 4
5   ** * 5
6        6
 12345678
```

ACM Regional Programming Contest 1999

Problema No. 6 - Canales de Comunicación
Programa Ejecutable: PROG6.EXE, PROG6.CLASS
Programa Fuente: PROG6.CPP, PROG6.C, PROG6.JAVA, PROG6.PAS
Archivo de entrada: PROG6.IN **Archivo de salida**: PROG6.OUT

Cuando una estación de radio está transmitiendo sobre un área muy grande, se utilizan repetidoras para retransmitir la señal con el propósito de que cada receptor tenga una señal fuerte. Sin embargo, los canales usados por cada repetidora deben ser cuidadosamente seleccionados de manera que las repetidoras cercanos no interfieran una con otra. Esta condición se satisface si las repetidoras adyacentes usan canales de comunicación diferentes. Ya que el espectro de radiofrecuencia es un recurso valioso, el número de canales requeridos por una red de repetidoras dada deber ser minimizado. Escriba un programa que lea una descripción de varias redes de repetidoras y determine el número mínimo de canales requeridos.

Entrada

La entrada consiste de varios mapas de redes de repetidoras. Cada mapa inicia con una línea que contiene el número de repetidoras en el rango 1 a 26, seguida de una lista de repetidoras con sus respectivas adyacentes. Las repetidoras son referidas por letras mayúsculas consecutivas del alfabeto iniciando con la A. Por ejemplo, 10 repetidoras son identificadas por las letras A, B, C, …,I y J.

Cada línea tiene la forma:

```
A : BCDH
```

la cual indica que las repetidoras B, C, D y H son adyacentes a la repetidora A. La primera línea describe las adyacentes a la repetidora A, la segunda las adyacentes a B, y así para todas las repetidoras. Si una repetidora no es adyacente a alguna otra, su línea tiene la forma :

```
A :
```

Las repetidoras son listadas en orden alfabético.

Nota que la adyacencia es una relación simétrica; si A es adyacente a B, entonces B es adyacente a A. Por otro lado, como las repetidoras están en un plano, la gráfica formada al conectar repetidoras adyacentes no tiene segmentos de línea que se crucen.

Una red A con cero repetidoras indica el fin de la entrada.

Salida

Para cada mapa excepto el último (que indica el fin de la entrada), imprime una línea que contiene el número mínimo de canales de comunicación requeridos para que no exista interferencia entre repetidoras adyacentes. El ejemplo de salida muestra el formato de esta línea, ten cuidado con mostrar el mensaje adecuado en inglés, y según sea el caso utilice la forma singular o plural.

Ejemplo del archivo de entrada

```
2
A:
B:
4
A:BC
B:ACD
C:ABD
D:BC
4
A:BCD
B:ACD
C:ABD
D:ABC
0
```

Ejemplo del archivo de salida

```
1 channel is needed
3 channel are needed
4 channel are needed
```