Problem A. Turtles

Input: Standard
Output: Standard
Judge: CodeForces

You've got a table of size $nx\,m$. We'll consider the table rows numbered from top to bottom 1 through n, and the columns numbered from left to right 1 through m. Then we'll denote the cell in row x and column y as (x, y).

Initially cell $(1, 1)$ contains two similar turtles. Both turtles want to get to cell (n, m). Some cells of the table have obstacles but it is guaranteed that there aren't any obstacles in the upper left and lower right corner. A turtle (one or the other) can go from cell (x, y) to one of two cells $(x + 1, y)$ and $(x, y + 1)$, as long as the required cell doesn't contain an obstacle. The turtles have had an argument so they don't want to have any chance of meeting each other along the way. Help them find the number of ways in which they can go from cell $(1, 1)$ to cell (n, m).

More formally, find the number of pairs of non-intersecting ways from cell $(1, 1)$ to cell (n, m) modulo 1000000007 $(10^9 + 7)$. Two ways are called non-intersecting if they have exactly two common points – the starting point and the final point.

Input

The first line contains two integers n, m $(2 \leq n, m \leq 3000)$. Each of the following n lines contains m characters describing the table. The empty cells are marked by ".", the cells with obstacles are marked by "#".

Output

In a single line print a single integer – the number of pairs of non-intersecting paths from cell (1,1) to cell (n, m) modulo 1000000007 $(10^9 + 7)$.

Samples

Input	Output
4 5 ..... .###. .###. 2 3	1 1

URL:
http://codeforces.com/problemset/problem/348/D

Problem B. Playing the ball

Input: Standard
Output: Standard
Judge: CodeForces

A coder cannot sit and code all day. Sometimes it is a good idea to rise from the desk, have a rest, have small talk with colleagues and even play. The coders of the F company have their favorite ball game.

Let's imagine the game on the plane with a cartesian coordinate system. The point (0, 0) contains the player who chooses an arbitrary direction and throws a ball in that direction. The ball hits the plane at distance d from the player's original position and continues flying in the same direction. After the ball hits the plane for the first time, it flies on and hits the plane again at distance $2 \cdot d$ from the player's original position and so on (it continue flying in the chosen direction and hitting the plane after each d units). All coders in the F company are strong, so the ball flies infinitely far away.

The plane has n circles painted on it. If a ball hits the plane and hits a circle that is painted on the plane (including its border), then the player gets one point. The ball can hit multiple circles at once and get one point for each of them (if the ball hits some circle x times during the move, the player also gets x points). Count the maximum number of points a player can get if he throws a ball in the arbitrary direction. Note that the direction may have real coordinates.

Input

The first line contains two space-separated integers – n, d ($1 \leq n \leq 2 \cdot 10^4; 5 \leq d \leq 10$) Next n lines contain the circles' description. The i-th line contains three space-separated integers x_i, y_i, r_i ($-10000 \leq x_i, y_i \leq 10000; 1 \leq r \leq 50$), where ($x_i$, y_i, r_i) are the coordinates of the center and the radius of the circle, correspondingly. The point (0, 0) is not inside or on the border of some circle.

Output

Print a single integer – the maximum number of points you can get.

Samples

Input	Output
2 5 1 1 1 5 0 1 2 5 4 0 3 5 3 1 1 10 20 0 10	1 2 3

URL:
http://codeforces.com/problemset/problem/420/E

Problem C. Nuts and Bolts

Input: Standard
Output: Standard
Judge: UVa

One afternoon your cell phone rings; it's your cousin Jimmy. "Hi Cuz," he says, "I need your help and I need it fast. I'm in the middle of a programming contest and however hard I try, I can't get one problem to finish within the two second time limit."

"Hmm... well..., isn't that a bit illegal?", you try to say to him. But he rattles on.

"I just snook out of the contest room and managed to send you my code and the sample I/O by email", he continues without pausing. "I will check my mail again in an hour, so please make it work for me."

"What about the problem description?", you ask.

"Can't do", he says, "Zoroman the Head Judge is already on my tail, so I got to go. Bye, ... and, eh, thanks."

Are you going to help him?

Jimmy's Code

```
#include <stdio.h>

#define MAX_BOLTS 500
#define MAX_NUTS  500

/* global copy of the input data */
int nuts,bolts;
int fits[MAX_BOLTS][MAX_NUTS];

void read_input_data(void){
   int n,b;

   scanf("%d%d",&bolts,&nuts);
   for(b=0;b<bolts;b++){
      for(n=0;n<nuts;n++){
         scanf("%d",&fits[b][n]);
         }
      }
   }

/* data used to match nuts with bolts */
int nut_used[MAX_NUTS];
int bestmatched;

void init_match(void){
   int n;

   bestmatched=0;
   for(n=0;n<nuts;n++) nut_used[n]=0;
   }

void match_bolt(int boltno, int matched){
   int n;

   if(boltno==bolts){
      if(matched>bestmatched) bestmatched=matched;
      return;
      }
```

```
39
40     /* don't match this bolt */
41     match_bolt(boltno+1,matched);
42
43     /* match with all unused nuts that fit this bolt */
44     for(n=0;n<nuts;n++) if(!nut_used[n] && fits[boltno][n]){
45        nut_used[n]=1;
46        match_bolt(boltno+1,matched+1);
47        nut_used[n]=0;
48        }
49     }
50
51  int main(){
52     int cases,caseno;
53
54     scanf("%d",&cases);
55     for(caseno=1;caseno<=cases;caseno++){
56        read_input_data();
57        init_match();
58        match_bolt(0,0);
59        printf("Case %d: ",caseno);
60        printf("a maximum of %d nuts and bolts ",bestmatched);
61        printf("can be fitted together\n");
62        }
63
64     return 0;
65     }
```

Samples

Input	Output
2 3 4 0 0 1 0 1 1 0 1 0 0 1 0 5 5 1 0 0 1 1 1 1 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1	Case 1: a maximum of 2 nuts and bolts can be fitted together Case 2: a maximum of 5 nuts and bolts can be fitted together

URL:
http://uva.onlinejudge.org/index.php?
option=onlinejudge&page=show_problem&problem=2079

Problem D. Software Allocation

Input: `Standard`
Output: `Standard`
Judge: `UVa`

A computing center has ten different computers (numbered 0 to 9) on which applications can run. The computers are not multi-tasking, so each machine can run only one application at any time. There are 26 applications, named A to Z. Whether an application can run on a particular computer can be found in a job description (see below).

Every morning, the users bring in their applications for that day. It is possible that two users bring in the same application; in that case two different, independent computers will be allocated for that application.

A clerk collects the applications, and for each different application he makes a list of computers on which the application could run. Then, he assigns each application to a computer. Remember: the computers are *not* multi-tasking, so each computer must handle at most one application in total. (An application takes a day to complete, so that sequencing i.e. one application after another on the same machine is not possible).

A job description consists of

1. one upper case letter $A \ldots Z$, indicating the application.
2. one digit $0 \ldots 9$, indicating the number of users who brought in the application.
3. a blank (space character.)
4. one or more different digits $0 \ldots 9$, indicating the computers on which the application can run.
5. a terminating semicolon ";".
6. an end-of-line.

Input

The input for your program is a textfile. For each day it contains one or more job descriptions, separated by a line containing only the end-of-line marker. The input file ends with the standard end-of-file marker. For each day your program determines whether an allocation of applications to computers can be done, and if so, generates a possible allocation.

Output

The output is also a textfile. For each day it consists of one of the following:

- ten characters from the set $A \ldots Z$, _, indicating the applications allocated to computers 0 to 9 respectively if an allocation was possible. An underscore "_" means that no application is allocated to the corresponding computer.
- a single character "!", if no allocation was possible.

Samples

Input	Output
A4 01234; Q1 5; P4 56789; A4 01234; Q1 5; P5 56789;	AAAA_QPPPP !

URL:
http://uva.onlinejudge.org/index.php?
option=onlinejudge&page=show_problem&problem=195

Credits

LaTeX edition by:

Daniel Cañizares Corrales.
Professor, Universidad Católica de Oriente.

All problems can be found on the provided URLs.
This compilation was made for educational purposes.