G. Fall Down

Mauricio Carazas Segovia

Hay una cuadrícula con n Filas y m Columnas y tres tipos de celdas:

Una celda vacía, denotada con ".".

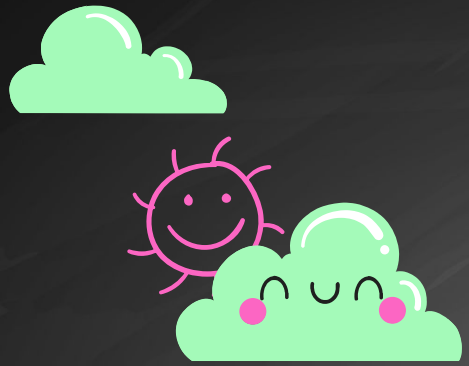
Una piedra, denotada con "*".

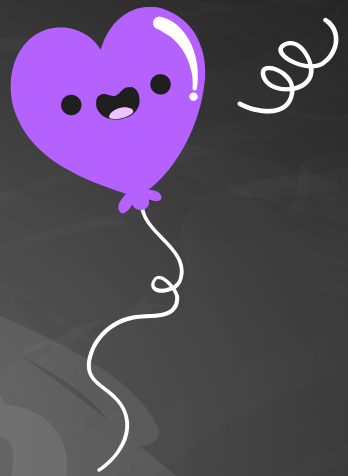
Un obstáculo, denotado con la letra latina minúscula 'o'.

Todas las piedras caen hasta que se encuentran con el suelo (la fila inferior), un obstáculo u otra piedra que ya es inamovible. (En otras palabras, todas las piedras se caen mientras puedan caer).

Simula el proceso. ¿Cómo es la cuadrícula resultante?

Input





T el numero de casos

- n el número de columnas
- m el número de consultas

Entonces N Siguen las líneas, cada una con m Caracteres. Cada uno de estos caracteres es '.', '*' o 'o', una celda vacía, una piedra o un obstáculo, respectivamente.

input Copy

```
3
6 10
.*.*....*.
.*.......*
...o....o.
.*.*....*.
..........
.o......o*
2 9
...***ooo
.*o.*o.*o
5 5
*****
*....
*****
....*
*****
```

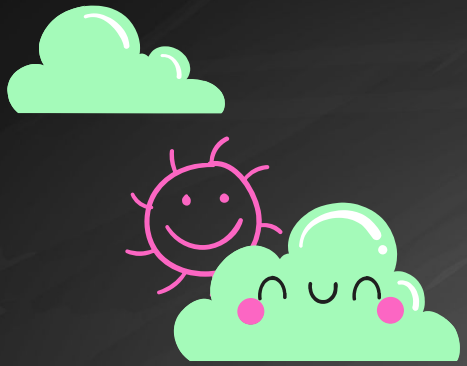
output Copy

```
..........
...*....*.
.*.o....o.
.*........
.*......**
.o.*....o*

....**ooo
.*o**o.*o

.....
*...*
*****
*****
*****
```

Output



Para cada caso de prueba, genera una cuadrícula con n Filas y m Columnas, que muestran el resultado del proceso

main.cpp

```cpp
#include <iostream>
using namespace std;

const char EMPTY = '.';
const char STONE = '*';
const char OBSTACLE = 'o';

int main() {
  int t;
  cin >> t;
  while (t--) {
    int n, m;
    cin >> n >> m;
    char g[n][m];
    for (int r = 0; r < n; r++) {
      for (int c = 0; c < m; c++) {
        cin >> g[r][c];
      }
    }
```

main.cpp

```
    }
  }
  for (int r = n - 1; r > -1; r--) {
    for (int c = 0; c < m; c++) {
      if (g[r][c] == STONE) {
        int nr = r;
        while (nr < n - 1 and g[nr + 1][c] == EMPTY) {
          nr++;
        }
        g[r][c] = EMPTY;
        g[nr][c] = STONE;
      }
    }
  }

  //cout <<endl<<endl<< "solution:"<<endl<<endl;
  for (int r = 0; r < n; r++) {
    for (int c = 0; c < m; c++) {
      cout << g[r][c];
    }
    cout << "\n";
  }
 }
 return 0;
}
```

There is a grid with n rows and m columns, and three types of cells:

- An empty cell, denoted with '.'.
- A stone, denoted with '*'.
- An obstacle, denoted with the lowercase Latin letter 'o'.

All stones fall down until they meet the floor (the bottom row), an obstacle, or other stone which is already immovable. (In other words, all the stones just fall down as long as they can fall.)

Simulate the process. What does the resulting grid look like?

Input

The input consists of multiple test cases. The first line contains an integer t ($1 \le t \le 100$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 50$) — the number of rows and the number of columns in the grid, respectively.

Then n lines follow, each containing m characters. Each of these characters is either '.', '*', or 'o' — an empty cell, a stone, or an obstacle, respectively.

Output

For each test case, output a grid with n rows and m columns, showing the result of the process.

You don't need to output a new line after each test, it is in the samples just for clarity.

Example

input

Copy

```
3
6 10
.*.*....*.
.*.......*
...o....o.
.*.*....*.
..........
.o......o*
2 9
...***ooo
.*o.*o.*o
5 5
*****
*....
*****
....*
*****
```

→ Virtual participation

Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you - solve these problems in the archive. If you just want to solve some problem from a contest, a virtual contest is not for you - solve this problem in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.

Start virtual contest

→ Clone Contest to Mashup

You can clone this contest to a mashup.

Clone Contest

→ Submit?

Language: GNU G++14 6.4.0

Choose file: Seleccionar archivo ning...ado

Submit

→ Last submissions

Submission	Tim	Verdict
181997781		Accepted

→ Problem tags

dfs and similar | implementation | *1200