task	eeprom	tornjevi
input	standard input	
output	standard output	
time limit	5 seconds	1 second
memory limit	64 MB	
points	100	100
	200	

EEPROM

Mirko and Stanko each have one EEPROM rectangular memory module consisting of R×16 bits.

They also have the circuitry needed to write data into the memory module. In one operation they can **change the state of a contiguous sequence of bits in a row or column**.

Mirko and Stanko's dad asked them to write a particular arrangement of bits into their memory modules. Whoever is faster will get repeats for lunch so Mirko is looking for a program which will help him win.

Write a program which will determine the **smallest possible number of operations** needed for Mirko to **write the given data** into his memory module. Initially, all bits in the memory module are zero.

Input

The first line of input contains an integer R ($1 \leq R \leq 50$), the number of rows in the module.

Each of the following R rows contains a string of 16 digits '0' or '1', data to be written into the module.

Output

Output the smallest number of operations.

Example test cases

input

```
5
0000111111110000
0010000000000000
1101111111111111
0010000000000000
0000000000000000
```

output

```
3
```

input

```
4
0000001000000010
0000110111000011
0111001000001111
0000001000000011
```

output

```
6
```

We are being attacked on a map represented by a rectangular grid of R×S squares. The attackers are barefoot robbers, and we use small cannons on small wooden towers to defend ourselves.

Each tower is equipped with **two cannons**, placed to fire in a 90 degree angle. More precisely, cannons on one tower can be set to fire in one of the following four configurations:

1. fire left and down;
2. fire down and right;
3. fire right and up;
4. fire up and left.

A cannon ball that hits the attacker **destroys him and continues to fly** in the same direction. A cannon ball which hits a castle stops and does no damage to the castle (because castles are big and strong). But, when a cannon ball hits a tower, it destroys it (because towers are small and fragile).

We want to turn the cannons on the towers so that, when we fire exactly one shot **from every cannon, we destroy all the attackers**, and **all our towers remain undamaged**.

Input

The first line contains two integers R and S ($1 \leq R, S \leq 100$), the dimensions of the map.

The next R lines contain S characters each, the map.

Each character on the map can be the uppercase letter 'T' (tower), lowercase letter 'n' (attacker), the character '#' (castle) or the character '.' (empty).

Note: There will always be a solution, although not necessarily unique.

Output

Output the map in the same format as in the input, replacing 'T' characters with the orientations of the cannons – each tower should be replaced with one of the digits '1', '2', '3' or '4', corresponding to the four orientations as described above.

Example test cases

input

```
9 13
.............
...........n.
.n.T..nnnn#..
.............
.T#n..n....T.
.............
.n.T..T....n.
.............
......n......
```

output

```
.............
...........n.
.n.3..nnnn#..
.............
.4#n..n....4.
.............
.n.1..2....n.
.............
......n......
```

input

```
5 9
.n..T..n.
.T..n....
.n..#..n.
....n..T.
.n..T..n.
```

output

```
.n..4..n.
.2..n....
.n..#..n.
....n..4.
.n..3..n.
```

input

```
9 8
n.Tnnnnn
nnnnnnTn
nTnnnnnn
nnnnTnnn
Tnnnnnnn
..#nnTnn
nnnnnnnT
nnnTn.n.
.nTnnnnn
```

output

```
n.3nnnnn
nnnnnn1n
n2nnnnnn
nnnn1nnn
3nnnnnnn
..#nn4nn
nnnnnnn4
nnn4n.n.
.n3nnnnn
```