A. Lever

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a description of a lever as string s. We'll represent the string length as record $|s|$, then the lever looks as a horizontal bar with weights of length $|s| - 1$ with exactly one pivot. We will assume that the bar is a segment on the Ox axis between points 0 and $|s| - 1$.

The decoding of the lever description is given below.

- If the i-th character of the string equals "^", that means that at coordinate i there is the pivot under the bar.
- If the i-th character of the string equals "=", that means that at coordinate i there is nothing lying on the bar.
- If the i-th character of the string equals digit c (1-9), that means that at coordinate i there is a weight of mass c on the bar.

Your task is, given the lever description, print if it will be in balance or not. Assume that the bar doesn't weight anything. Assume that the bar initially is in balance then all weights are simultaneously put on it. After that the bar either tilts to the left, or tilts to the right, or is in balance.

Input

The first line contains the lever description as a non-empty string s $(3 \le |s| \le 10^6)$, consisting of digits (1-9) and characters "^" and "=". It is guaranteed that the line contains exactly one character "^". It is guaranteed that the pivot of the lever isn't located in any end of the lever bar.

To solve the problem you may need 64-bit integer numbers. Please, do not forget to use them in your programs.

Output

Print "left" if the given lever tilts to the left, "right" if it tilts to the right and "balance", if it is in balance.

Sample test(s)

input
```
=^==
```
output
```
balance
```

input
```
9===^==1
```
output
```
left
```

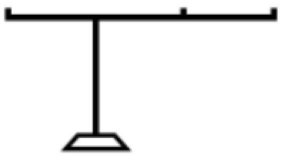
input
```
2==^7==
```
output
```
right
```

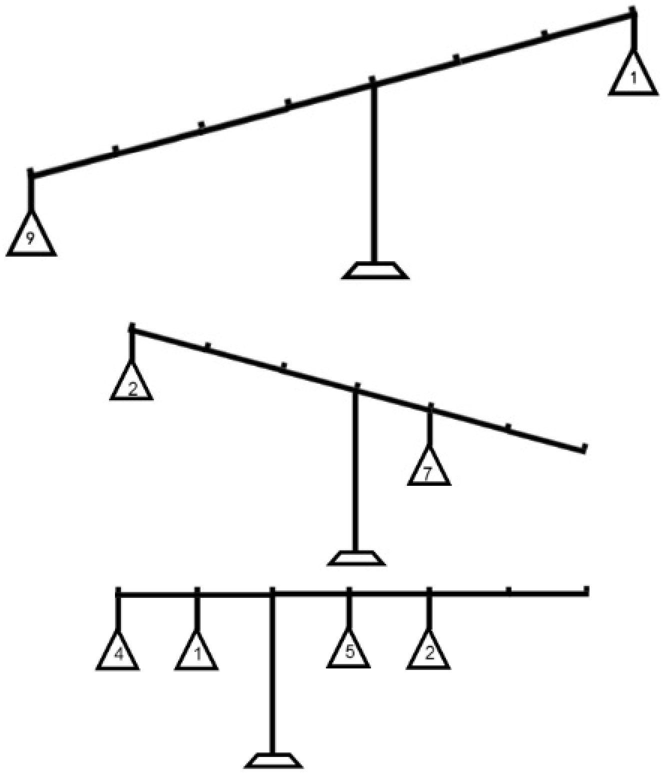
input
```
41^52==
```
output
```
balance
```

Note

As you solve the problem, you may find the following link useful to better understand how a lever functions: http://en.wikipedia.org/wiki/Lever.

The pictures to the examples:




9
1
2
7
4
1
5
2

B. I.O.U.

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Imagine that there is a group of three friends: A, B and C. A owes B 20 rubles and B owes C 20 rubles. The total sum of the debts is 40 rubles. You can see that the debts are not organized in a very optimal manner. Let's rearrange them like that: assume that A owes C 20 rubles and B doesn't owe anything to anybody. The debts still mean the same but the total sum of the debts now equals 20 rubles.

This task is a generalisation of a described example. Imagine that your group of friends has n people and you know the debts between the people. Optimize the given debts without changing their meaning. In other words, finally for each friend the difference between the total money he should give and the total money he should take must be the same. Print the minimum sum of all debts in the optimal rearrangement of the debts. See the notes to the test samples to better understand the problem.

Input

The first line contains two integers n and m $(1 \le n \le 100;\ 0 \le m \le 10^4)$. The next m lines contain the debts. The i-th line contains three integers a_i, b_i, c_i $(1 \le a_i, b_i \le n;\ a_i \ne b_i;\ 1 \le c_i \le 100)$, which mean that person a_i owes person b_i c_i rubles.

Assume that the people are numbered by integers from 1 to n.

It is guaranteed that the same pair of people occurs at most once in the input. The input doesn't simultaneously contain pair of people (x, y) and pair of people (y, x).

Output

Print a single integer — the minimum sum of debts in the optimal rearrangement.

Sample test(s)

input

```
5 3
1 2 10
2 3 1
2 4 1
```

output

```
10
```

input

```
3 0
```

output

```
0
```

input

```
4 3
1 2 1
2 3 1
3 1 1
```

output

```
0
```

Note

In the first sample, you can assume that person number 1 owes 8 rubles to person number 2, 1 ruble to person number 3 and 1 ruble to person number 4. He doesn't owe anybody else anything. In the end, the total debt equals 10.

In the second sample, there are no debts.

In the third sample, you can annul all the debts.

C. Divisible by Seven

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have number a, whose decimal representation quite luckily contains digits 1, 6, 8, 9. Rearrange the digits in its decimal representation so that the resulting number will be divisible by 7.

Number a doesn't contain any leading zeroes and contains digits 1, 6, 8, 9 (it also can contain another digits). The resulting number also mustn't contain any leading zeroes.

Input

The first line contains positive integer a in the decimal record. It is guaranteed that the record of number a contains digits: 1, 6, 8, 9. Number a doesn't contain any leading zeroes. The decimal representation of number a contains at least 4 and at most 10^6 characters.

Output

Print a number in the decimal notation without leading zeroes — the result of the permutation.

If it is impossible to rearrange the digits of the number a in the required manner, print 0.

Sample test(s)

input
```
1689
```
output
```
1869
```

input
```
18906
```
output
```
18690
```

D. Maximum Submatrix 2

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a matrix consisting of digits zero and one, its size is $n \times m$. You are allowed to rearrange its rows. What is the maximum area of the submatrix that only consists of ones and can be obtained in the given problem by the described operations?

Let's assume that the rows of matrix a are numbered from 1 to n from top to bottom and the columns are numbered from 1 to m from left to right. A matrix cell on the intersection of the i-th row and the j-th column can be represented as (i, j). Formally, a submatrix of matrix a is a group of four integers d, u, l, r $(1 \le d \le u \le n;\ 1 \le l \le r \le m)$. We will assume that the submatrix contains cells (i, j) $(d \le i \le u;\ l \le j \le r)$. The area of the submatrix is the number of cells it contains.

Input

The first line contains two integers n and m $(1 \le n, m \le 5000)$. Next n lines contain m characters each — matrix a. Matrix a only contains characters: "0" and "1". Note that the elements of the matrix follow without any spaces in the lines.

Output

Print a single integer — the area of the maximum obtained submatrix. If we cannot obtain a matrix of numbers one, print 0.

Sample test(s)

input

```
1 1
1
```

output

```
1
```

input

```
2 2
10
11
```

output

```
2
```

input

```
4 3
100
011
000
101
```

output

```
2
```

E. Circling Round Treasures

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a map as a rectangle table. Each cell of the table is either an obstacle, or a treasure with a certain price, or a bomb, or an empty cell. Your initial position is also given to you.

You can go from one cell of the map to a side-adjacent one. At that, you are not allowed to go beyond the borders of the map, enter the cells with treasures, obstacles and bombs. To pick the treasures, you need to build a closed path (starting and ending in the starting cell). The closed path mustn't contain any cells with bombs inside. Let's assume that the sum of the treasures' values that are located inside the closed path equals v, and besides, you've made k single moves (from one cell to another) while you were going through the path, then such path brings you the profit of $v - k$ rubles.

Your task is to build a closed path that doesn't contain any bombs and brings maximum profit.

Note that the path can have self-intersections. In order to determine if a cell lies inside a path or not, use the following algorithm:

1. Assume that the table cells are points on the plane (the table cell on the intersection of the i-th column and the j-th row is point (i, j)). And the given path is a closed polyline that goes through these points.
2. You need to find out if the point p of the table that is not crossed by the polyline lies inside the polyline.
3. Let's draw a ray that starts from point p and does not intersect other points of the table (such ray must exist).
4. Let's count the number of segments of the polyline that intersect the painted ray. If this number is odd, we assume that point p (and consequently, the table cell) lie inside the polyline (path). Otherwise, we assume that it lies outside.

Input

The first line contains two integers n and m $(1 \le n, m \le 20)$ — the sizes of the table. Next n lines each contains m characters — the description of the table. The description means the following:

- character "`B`" is a cell with a bomb;
- character "`S`" is the starting cell, you can assume that it's empty;
- digit c (`1`-`8`) is treasure with index c;
- character "`.`" is an empty cell;
- character "`#`" is an obstacle.

Assume that the map has t treasures. Next t lines contain the prices of the treasures. The i-th line contains the price of the treasure with index i, v_i $(-200 \le v_i \le 200)$. It is guaranteed that the treasures are numbered from 1 to t. It is guaranteed that the map has not more than 8 objects in total. Objects are bombs and treasures. It is guaranteed that the map has exactly one character "`S`".

Output

Print a single integer — the maximum possible profit you can get.

Sample test(s)

input

```
4 4
....
.S1.
....
....
10
```

output

```
2
```

input

```
7 7
.......
.1###2.
.#...#.
.#.B.#.
.3...4.
..##...
......S
100
100
100
100
```

output

```
364
```

input

```
7 8
........
```

```
........
....1B..
.S......
....2...
3.......
........
100
-100
100
```

output

```
0
```

input

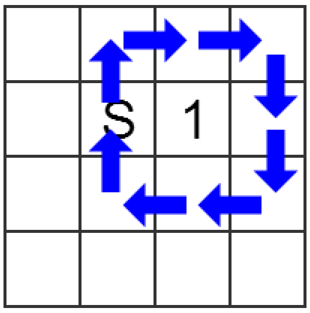
```
1 1
S
```

output

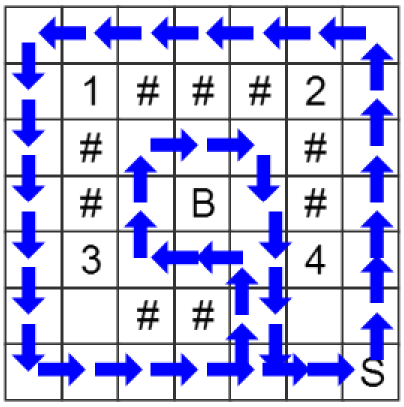
```
0
```

Note

In the first example the answer will look as follows.



In the second example the answer will look as follows.



In the third example you cannot get profit.

In the fourth example you cannot get profit as you cannot construct a closed path with more than one cell.