11230 - Annoying painting tool

Time limit: 3.000 seconds

Maybe you wonder what an annoying painting tool is? First of all, the painting tool we speak of supports only black and white. Therefore, a picture consists of a rectangular area of pixels, which are either black or white. Second, there is only one operation how to change the colour of pixels:

Select a rectangular area of **r** rows and **c** columns of pixels, which is completely inside the picture. As a result of the operation, each pixel inside the selected rectangle changes its colour (from black to white, or from white to black).

Initially, all pixels are white. To create a picture, the operation described above can be applied several times. Can you paint a certain picture which you have in mind?

Input Specification

The input contains several test cases. Each test case starts with one line containing four integers **n**, **m**, **r** and **c**. ($1 \le r \le n \le 100$, $1 \le c \le m \le 100$), The following **n** lines each describe one row of pixels of the painting you want to create. The **i**th line consists of **m** characters describing the desired pixel values of the **i**th row in the finished painting ('0' indicates white, '1' indicates black).

The last test case is followed by a line containing four zeros.

Output Specification

For each test case, print the minimum number of operations needed to create the painting, or -1 if it is impossible.

Sample Input

```
3 3 1 1
010
101
010
4 3 2 1
011
110
011
110
3 4 2 2
0110
0111
0000
0 0 0 0
```

Sample Output

```
4
6
-1
```

11231 - Black and white painting

Time limit: 3.000 seconds

You are visiting the Centre Pompidou which contains a lot of modern paintings. In particular you notice one painting which consists solely of black and white squares, arranged in rows and columns like in a chess board (no two adjacent squares have the same colour). By the way, the artist did not use the tool of problem A to create the painting.

Since you are bored, you wonder how many 8×8 chess boards are embedded within this painting. The bottom right corner of a chess board must always be white.

Input Specification

The input contains several test cases. Each test case consists of one line with three integers **n**, **m** and **c**. ($8 \leq n, m \leq 40000$), where **n** is the number of rows of the painting, and **m** is the number of columns of the painting. **c** is always *0* or *1*, where *0* indicates that the bottom right corner of the painting is black, and *1* indicates that this corner is white.

The last test case is followed by a line containing three zeros.

Output Specification

For each test case, print the number of chess boards embedded within the given painting.

Sample Input

```
8 8 0
8 8 1
9 9 1
40000 39999 0
0 0 0
```

Sample Output

```
0
1
2
799700028
```

11232 - Cylinder

Time limit: 3.000 seconds

Using a sheet of paper and scissors, you can cut out two faces to form a cylinder in the following way:

1. Cut the paper horizontally (parallel to the shorter side) to get two rectangular parts.
2. From the first part, cut out a circle of maximum radius. The circle will form the bottom of the cylinder.
3. Roll the second part up in such a way that it has a perimeter of equal length with the circle's circumference, and attach one end of the roll to the circle. Note that the roll may have some overlapping parts in order to get the required length of the perimeter.

Given the dimensions of the sheet of paper, can you calculate the biggest possible volume of a cylinder which can be constructed using the procedure described above?

Input Specification

The input consists of several test cases. Each test case consists of two numbers **w** and **h** ($1 \leq w \leq h \leq 100$), which indicate the width and height of the sheet of paper.

The last test case is followed by a line containing two zeros.

Output Specification

For each test case, print one line with the biggest possible volume of the cylinder. Round this number to 3 places after the decimal point.

Sample Input

```
10 10
10 50
10 30
0 0
```

Sample Output

```
54.247
785.398
412.095
```

In the first case, the optimal cylinder has a radius of about 1.591549, in the second case, the optimal cylinder has a radius of 5, and in the third case, the optimal cylinder has a radius of about 3.621795.

11234 - Expressions

Time limit: 3.000 seconds

Arithmetic expressions are usually written with the operators in between the two operands (which is called infix notation). For example, *(x+y)*(z-w)* is an arithmetic expression in infix notation. However, it is easier to write a program to evaluate an expression if the expression is written in postfix notation (also known as reverse polish notation). In postfix notation, an operator is written behind its two operands, which may be expressions themselves. For example, *x y + z w - ** is a postfix notation of the arithmetic expression given above. Note that in this case parentheses are not required.

To evaluate an expression written in postfix notation, an algorithm operating on a stack can be used. A stack is a data structure which supports two operations:

1. **push**: a number is inserted at the top of the stack.
2. **pop**: the number from the top of the stack is taken out.

During the evaluation, we process the expression from left to right. If we encounter a number, we push it onto the stack. If we encounter an operator, we pop the first two numbers from the stack, apply the operator on them, and push the result back onto the stack. More specifically, the following pseudocode shows how to handle the case when we encounter an operator O:

```
a := pop();
b := pop();
push(b O a);
```

The result of the expression will be left as the only number on the stack.

Now imagine that we use a queue instead of the stack. A queue also has a push and pop operation, but their meaning is different:

1. **push**: a number is inserted at the end of the queue.
2. **pop**: the number from the front of the queue is taken out of the queue.

Can you rewrite the given expression such that the result of the algorithm using the queue is the same as the result of the original expression evaluated using the algorithm with the stack?

Input Specification

The first line of the input contains a number **T** (*T* ≤ *200*). The following **T** lines each contain one expression in postfix notation. Arithmetic operators are represented by uppercase letters, numbers are represented by lowercase letters. You may assume that the length of each expression is less than *10000* characters.

Output Specification

For each given expression, print the expression with the equivalent result when using the algorithm with the queue instead of the stack. To make the solution unique, you are not allowed to assume that the operators are associative or commutative.

Sample Input

```
2
xyPzwIM
abcABdefgCDEF
```

Sample Output

```
wzyxIPM
gfCecbDdAaEBF
```

11236 - Grocery store

Time limit: 3.000 seconds

A cashier in a grocery store seems to have difficulty in distinguishing the multiplication symbol and the addition symbol. To make things easier for him, you want to buy goods in such a way that the product of their prices is the same as the sum of their prices.

Of course, if you buy only one item, this is always true. With two items and three items, it still seems quite a boring task to you, so now you are interested in finding possible prices of four items such that the sum of the four prices is equal to the product of the four prices. You should consider the prices are in € with two digits after the decimal point. Obviously, each product costs at least one cent.

Input Specification

This problem has no input.

Output Specification

Print all solutions which have a sum of the four items of at most **20.00 €**. For each solution, print one line with the prices of the four items in non-decreasing order, with one space character between them. You may print the solutions in any order, but make sure to print each solution only once.

Sample Output

```
0.50 1.00 2.50 16.00
1.25 1.60 1.75 1.84
1.25 1.40 1.86 2.00
...
```