Task	MODULO	HERMAN	OKVIRI	SLIKAR	BOND	DEBUG
Input	standard input (**keyboard**)					
Output	standard output (**screen)**					
Memory limit (heap)	32 MB	32 MB	32 MB	32 MB	32 MB	32 MB
Memory limit (stack)	8 MB	8 MB	8 MB	8 MB	8 MB	8 MB
Time limit (per test)	1 sec	1 sec	1 sec	1 sec	1 sec	1 sec
Number of tests	10	10	10	10	10	15
Points per test	1	2	3	5	7	8
Total points	**10**	**20**	**30**	**50**	**70**	**120**
	300					

Note: The time limit is based on a computer running two AMD Athlon MP 2600+ processors and Linux operating system.

C and C++ programs will be compiled with the following options: -O2 -lm.
Pascal programs will be compiled with the following options: -So -O1 -XS.

1. MODULO

Given two integers A and B, A modulo B is the remainder when dividing A by B. For example, the numbers 7, 14, 27 and 38 become 1, 2, 0 and 2, modulo 3. Write a program that accepts 10 numbers as input and outputs the number of distinct numbers in the input, if the numbers are considered modulo 42.

Input

The input will contain 10 non-negative integers, each smaller than 1000, one per line.

Output

Output the number of distinct values when considered modulo 42 on a single line.

Sample tests

input	**input**	**input**
1 2 3 4 5 6 7 8 9 10	42 84 252 420 840 126 42 84 420 126	39 40 41 42 43 44 82 83 84 85
output 10	**output** 1	**output** 6

Clarification:
In the first example, the numbers modulo 42 are 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10.
In the second example all numbers modulo 42 are 0.
In the third example, the numbers modulo 42 are 39, 40, 41, 0, 1, 2, 40, 41, 0 and 1. There are 6 distinct numbers.

2. HERMAN

The 19th century German mathematician Hermann Minkowski investigated a non-Euclidian geometry, called the taxicab geometry. In taxicab geometry the distance between two points T1(x1, y1) and T2(x2, y2) is defined as:

$$D(T1,T2) = |x1 - x2| + |y1 - y2|$$

All other definitions are the same as in Euclidian geometry, including that of a circle:

A **circle** is the set of all points in a plane at a fixed distance (the radius) from a fixed point (the centre of the circle).

We are interested in the difference of the areas of two circles with radius R, one of which is in normal (Euclidian) geometry, and the other in taxicab geometry. The burden of solving this difficult problem has fallen onto you.

Input

The first and only line of input will contain the radius R, an integer smaller than or equal to 10000.

Output

On the first line you should output the area of a circle with radius R in normal (Euclidian) geometry. On the second line you should output the area of a circle with radius R in taxicab geometry.

Note: Outputs within ±0.0001 of the official solution will be accepted.

Sample tests

input 1 **output** 3.141593 2.000000	**input** 21 **output** 1385.442360 882.000000	**input** 42 **output** 5541.769441 3528.000000

"Peter Pan frames" are a way of decorating text in which every character is framed by a diamond-shaped frame, with frames of neigbhouring characters interleaving. A Peter Pan frame for one letter looks like this ('X' is the letter we are framing):

```
..#..
.#.#.
#.X.#
.#.#.
..#..
```

However, such a framing would be somewhat dull so we'll frame every third letter using a "Wendy frame". A Wendy frame looks like this:

```
..*..
.*.*.
*.X.*
.*.*.
..*..
```

When a Wendy frame interleaves with a Peter Pan frame, the Wendy frame (being much nicer) is put on top. For an example of the interleaving check the sample cases.

Input

The first and only line of input will contain at most 15 capital letters of the English alphabet.

Output

Output the word written using Peter Pan and Wendy frames on 5 lines.

Sample tests

input
```
A
```

output
```
..#..
.#.#.
#.A.#
.#.#.
..#..
```

input
```
DOG
```

output
```
..#...#...*..
.#.#.#.#.*.*.
#.D.#.O.*.G.*
.#.#.#.#.*.*.
..#...#...*..
```

input
```
ABCD
```

output
```
..#...#...*...#..
.#.#.#.#.*.*.#.#.
#.A.#.B.*.C.*.D.#
.#.#.#.#.*.*.#.#.
..#...#...*...#..
```

4. SLIKAR

The evil emperor Cactus has in his possession the Magic Keg and has flooded the Enchanted Forest! The Painter and the three little hedgehogs now have to return to the Beaver's den where they will be safe from the water as quickly as possible!

The map of the Enchanted Forest consists of R rows and C columns. Empty fields are represented by '.' characters, flooded fields by '*' and rocks by 'X'. Additionally, the Beaver's den is represented by 'D' and the Painter and the three little hedgehogs are shown as 'S'.

Every minute the Painter and the three little hedgehogs can move to 4 neighbouring fields (up, down, left or right). Every minute the flood expands as well so that all empty fields that have at least one common side with a flooded field become flooded as well. Neither water nor the Painter and the three little hedgehogs can pass through rocks. Naturally, the Painter and the three little hedgehogs cannot pass through flooded fields, and water cannot flood the Beaver's den.

Write a program that will, given a map of the Enchanted Forest, output the **shortest** time needed for the Painter and the three little hedgehogs to safely reach the Beaver's den.

Note: The Painter and the three little hedgehogs cannot move into a field that is about to be flooded (in the same minute).

Input

The first line of input will contain two integers, R and C, smaller than or equal to 50.
The following R lines will each contain C characters ('.', '*', 'X', 'D' or 'S'). The map will contain exactly one 'D' character and exactly one 'S' character.

Output

Output the **shortest** possible time needed for the Painter and the three little hedgehogs to safely reach the Beaver's den. If this is impossible output the word "KAKTUS" on a line by itself.

Sample tests

input 3 3 D.* ... .S. **output** 3	**input** 3 3 D.* ... ..S **output** KAKTUS	**input** 3 6 D...*. .X.X.. ....S. **output** 6

Clarification of the second sample test: The best they can do is to go along the lower border and then the left border, and get flooded one minute before reaching the den.

5. BOND

Everyone knows of the secret agent double-oh-seven, the popular Bond (James Bond). A lesser known fact is that he actually did not perform most of his missions by himself; they were instead done by his cousins, Jimmy Bonds. Bond (James Bond) has grown weary of having to distribute assign missions to Jimmy Bonds every time he gets new missions so he has asked you to help him out.

Every month Bond (James Bond) receives a list of missions. Using his detailed intelligence from past missions, for every mission and for every Jimmy Bond he calculates the probability of that particular mission being successfully completed by that particular Jimmy Bond. Your program should process that data and find the arrangement that will result in the **greatest** probability that **all** missions are completed successfully.

Note: the probability of all missions being completed successfully is equal to the product of the probabilities of the single missions being completed successfully.

Input

The first line will contain an integer N, the number of Jimmy Bonds and missions ($1 \leq N \leq 20$).
The following N lines will contain N integers between 0 and 100, inclusive. The j-th integer on the i-th line is the probability that Jimmy Bond i would successfully complete mission j, given as a percentage.

Output

Output the maximum probability of Jimmy Bonds successfully completing all the missions, as a percentage.

Note: Outputs within ±0.000001 of the official solution will be accepted.

Sample tests

input 2 100 100 50 50 **output** 50.000000	**input** 2 0 50 50 0 **output** 25.00000	**input** 3 25 60 100 13 0 50 12 70 90 **output** 9.10000

Clarification of the third example: If Jimmy bond 1 is assigned the 3rd mission, Jimmy Bond 2 the 1st mission and Jimmy Bond 3 the 2nd mission the probability is: 1.0 * 0.13 * 0.7 = 0.091 = 9.1%. All other arrangements give a smaller probability of success.

6. DEBUG

While debugging a program Mirko noticed that a bug in the program may be linked with the existence of so called square killers in the program memory. The program memory is a matrix composed of R rows and C columns consisting only of zeroes and ones. A square killer is a square submatrix in memory, consisting of more than one character, that, when rotated 180 degrees looks exactly the same. For example, the following matrix contains 3 square killers:

101010 111001 101001 memory	....10 ....01 ...... killer	....00. ...00. killer	101... 111... 101... killer

Mirko is wondering if there is a connection between the size of the largest square killer and the bug in the program. Help Mirko by writing a program that, given the layout of the memory, outputs the size of the largest square killer. The size of the square killer is the number of rows (or columns) that the killer consists of. In the example above the killer sizes are 2, 2 and 3, respectively.

Input

The first will contain two integers, R and C, smaller than or equal to 300.
The next R lines will each contain C characters ('0' or '1') with no spaces.

Output

Output the size of the largest killer on a single line, or output -1 if there are no square killers.

Sample tests

input 3 6 101010 111001 101001 **output** 3	**input** 4 5 10010 01010 10101 01001 **output** 3	**input** 3 3 101 111 100 **output** -1