Codeforces Round #102 (Div. 1)

A. Help Farmer

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

Once upon a time in the Kingdom of Far Far Away lived Sam the Farmer. Sam had a cow named Dawn and he was deeply attached to her. Sam would spend the whole summer stocking hay to feed Dawn in winter. Sam scythed hay and put it into haystack. As Sam was a bright farmer, he tried to make the process of storing hay simpler and more convenient to use. He collected the hay into cubical hay blocks of the same size. Then he stored the blocks in his barn. After a summer spent in hard toil Sam stored $A \cdot B \cdot C$ hay blocks and stored them in a barn as a rectangular parallelepiped A layers high. Each layer had B rows and each row had C blocks.

At the end of the autumn Sam came into the barn to admire one more time the hay he'd been stacking during this hard summer. Unfortunately, Sam was horrified to see that the hay blocks had been carelessly scattered around the barn. The place was a complete mess. As it turned out, thieves had sneaked into the barn. They completely dissembled and took away a layer of blocks from the parallelepiped's front, back, top and sides. As a result, the barn only had a parallelepiped containing $(A - 1) \times (B - 2) \times (C - 2)$ hay blocks. To hide the evidence of the crime, the thieves had dissembled the parallelepiped into single $1 \times 1 \times 1$ blocks and scattered them around the barn. After the theft Sam counted n hay blocks in the barn but he forgot numbers A, B и C.

Given number n, find the minimally possible and maximally possible number of stolen hay blocks.

Input

The only line contains integer n from the problem's statement ($1 \le n \le 10^9$).

Output

Print space-separated minimum and maximum number of hay blocks that could have been stolen by the thieves.

Note that the answer to the problem can be large enough, so you must use the 64-bit integer type for calculations. Please, do not use the %lld specificator to read or write 64-bit integers in C++. It is preferred to use cin, cout streams or the %I64d specificator.

Sample test(s)

input
4
output
28 41

input
7
output
47 65

input
12
output
48 105

Note

Let's consider the first sample test. If initially Sam has a parallelepiped consisting of $32 = 2 \times 4 \times 4$ hay blocks in his barn, then after the theft the barn has $4 = (2 - 1) \times (4 - 2) \times (4 - 2)$ hay blocks left. Thus, the thieves could have stolen $32 - 4 = 28$ hay blocks. If Sam initially had a parallelepiped consisting of $45 = 5 \times 3 \times 3$ hay blocks in his barn, then after the theft the barn has $4 = (5 - 1) \times (3 - 2) \times (3 - 2)$ hay blocks left. Thus, the thieves could have stolen $45 - 4 = 41$ hay blocks. No other variants of the blocks' initial arrangement (that leave Sam with exactly 4 blocks after the theft) can permit the thieves to steal less than 28 or more than 41 blocks.

B. Help General

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Once upon a time in the Kingdom of Far Far Away lived Sir Lancelot, the chief Royal General. He was very proud of his men and he liked to invite the King to come and watch drill exercises which demonstrated the fighting techniques and tactics of the squad he was in charge of. But time went by and one day Sir Lancelot had a major argument with the Fairy Godmother (there were rumors that the argument occurred after the general spoke badly of the Godmother's flying techniques. That seemed to hurt the Fairy Godmother very deeply).

As the result of the argument, the Godmother put a rather strange curse upon the general. It sounded all complicated and quite harmless: " *If the squared distance between some two soldiers equals to* 5*, then those soldiers will conflict with each other!*"

The drill exercises are held on a rectangular $n \times m$ field, split into nm square 1×1 segments for each soldier. Thus, the square of the distance between the soldiers that stand on squares (x_1, y_1) and (x_2, y_2) equals exactly $(x_1 - x_2)^2 + (y_1 - y_2)^2$. Now not all nm squad soldiers can participate in the drill exercises as it was before the Fairy Godmother's curse. Unless, of course, the general wants the soldiers to fight with each other or even worse... For example, if he puts a soldier in the square $(2, 2)$, then he cannot put soldiers in the squares $(1, 4)$, $(3, 4)$, $(4, 1)$ and $(4, 3)$ — each of them will conflict with the soldier in the square $(2, 2)$.

Your task is to help the general. You are given the size of the drill exercise field. You are asked to calculate the maximum number of soldiers that can be simultaneously positioned on this field, so that no two soldiers fall under the Fairy Godmother's curse.

Input

The single line contains space-separated integers n and m $(1 \le n, m \le 1000)$ that represent the size of the drill exercise field.

Output

Print the desired maximum number of warriors.

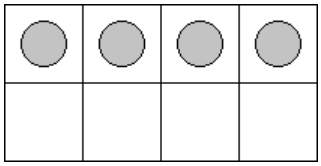
Sample test(s)

input
2 4
output
4

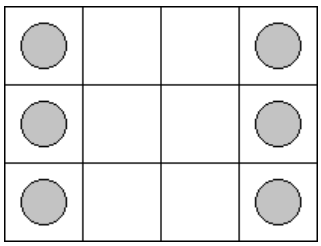
input
3 4
output
6

Note

In the first sample test Sir Lancelot can place his 4 soldiers on the 2×4 court as follows (the soldiers' locations are marked with gray circles on the scheme):



In the second sample test he can place 6 soldiers on the 3×4 site in the following manner:



C. Help Caretaker

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Autumn came late to the kingdom of Far Far Away. The harvest was exuberant and it is now time to get ready for the winter. As most people celebrate the Harvest festival, Simon the Caretaker tries to solve a very non-trivial task of how to find place for the agricultural equipment in the warehouse.

He's got problems with some particularly large piece of equipment, which is, of course, turboplows. The problem is that when a turboplow is stored, it takes up not some simply rectangular space. It takes up a T-shaped space like on one of the four pictures below (here character "#" stands for the space occupied by the turboplow and character "." stands for the free space):

```
###    ..#    .#.    #..
.#.    ###    .#.    ###
.#.    ..#    ###    #..
```

Simon faced a quite natural challenge: placing in the given $n \times m$ cells warehouse the maximum number of turboplows. As one stores the turboplows, he can rotate them in any manner (so that they take up the space like on one of the four pictures above). However, two turboplows cannot "overlap", that is, they cannot share the same cell in the warehouse.

Simon feels that he alone cannot find the optimal way of positioning the plugs in the warehouse that would maximize their quantity. Can you help him?

Input

The only line contains two space-separated integers n and m — the sizes of the warehouse ($1 \le n, m \le 9$).

Output

In the first line print the maximum number of turboplows that can be positioned in the warehouse. In each of the next n lines print m characters. Use "." (dot) to mark empty space and use successive capital Latin letters ("A" for the first turboplow, "B" for the second one and so on until you reach the number of turboplows in your scheme) to mark place for the corresponding turboplows considering that they are positioned in the optimal manner in the warehouse. The order in which you number places for the turboplows does not matter. If there are several optimal solutions for a warehouse of the given size, print any of them.

Sample test(s)

input
3 3

output
1 AAA .A. .A.

input
5 6

output
4 A..C.. AAAC.. ABCCCD .B.DDD BBB..D

input
2 2

output
0 .. ..

D. Help Shrek and Donkey 2

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Having learned (not without some help from the Codeforces participants) to play the card game from the previous round optimally, Shrek and Donkey (as you may remember, they too live now in the Kingdom of Far Far Away) have decided to quit the boring card games and play with toy soldiers.

The rules of the game are as follows: there is a battlefield, its size equals $n \times m$ squares, some squares contain the toy soldiers (the green ones belong to Shrek and the red ones belong to Donkey). Besides, each of the n lines of the area contains not more than two soldiers. During a move a players should **select** not less than 1 and not more than k soldiers belonging to him and make them either *attack* or *retreat*.

An *attack* is moving all of the selected soldiers along the lines on which they stand **in the direction of** an enemy soldier, if he is in this line. If this line doesn't have an enemy soldier, then the selected soldier on this line can move in any direction during the player's move. Each selected soldier has to move at least by one cell. Different soldiers can move by a different number of cells. During the attack the soldiers are not allowed to cross the cells where other soldiers stand (or stood immediately before the attack). It is also not allowed to go beyond the battlefield or finish the attack in the cells, where other soldiers stand (or stood immediately before attack).

A *retreat* is moving all of the selected soldiers along the lines on which they stand **in the direction from** an enemy soldier, if he is in this line. *The other rules repeat the rules of the attack.*

For example, let's suppose that the original battlefield had the form (here symbols "`G`" mark Shrek's green soldiers and symbols "`R`" mark Donkey's red ones):

```
-G-R-
-R-G-
```

Let's suppose that $k = 2$ and Shrek moves first. If he decides to *attack*, then after his move the battlefield can look like that:

```
--GR-     --GR-     -G-R-
-RG--     -R-G-     -RG--
```

If in the previous example Shrek decides to *retreat*, then after his move the battlefield can look like that:

```
G--R-     G--R-     -G-R-
-R--G     -R-G-     -R--G
```

On the other hand, the followings fields cannot result from Shrek's correct move:

```
G--R-     ---RG     --GR-
-RG--     -R-G-     GR---
```

Shrek starts the game. To make a move means to attack or to retreat by the rules. A player who cannot make a move loses and his opponent is the winner. Determine the winner of the given toy soldier game if Shrek and Donkey continue to be under the yellow pills from the last rounds' problem. Thus, they always play optimally (that is, they try to win if it is possible, or finish the game in a draw, by ensuring that it lasts forever, if they cannot win).

Input

The first line contains space-separated integers n, m and k ($1 \le n, m, k \le 100$). Then n lines contain m characters each. These characters belong to the set {"`-`", "`G`", "`R`"}, denoting, respectively, a battlefield's free cell, a cell occupied by Shrek's soldiers and a cell occupied by Donkey's soldiers.

It is guaranteed that each line contains no more than two soldiers.

Output

Print "`First`" (without the quotes) if Shrek wins in the given Toy Soldier game. If Donkey wins, print "`Second`" (without the quotes). If the game continues forever, print "`Draw`" (also without the quotes).

Sample test(s)

input
2 3 1 R-G RG-
output
First

input
3 3 2 G-R R-G G-R
output
Second

input

```
2 3 1
-R-
-G-
```

output

```
Draw
```

input

```
2 5 2
-G-R-
-R-G-
```

output

```
First
```

E. Help Greg the Dwarf 2

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Greg the Dwarf has been really busy recently with excavations by the Neverland Mountain. However for the well-known reasons (as you probably remember he is a very unusual dwarf and he cannot stand sunlight) Greg can only excavate at night. And in the morning he should be in his crypt before the first sun ray strikes. That's why he wants to find the shortest route from the excavation point to his crypt. Greg has recollected how the Codeforces participants successfully solved the problem of transporting his coffin to a crypt. So, in some miraculous way Greg appeared in your bedroom and asks you to help him in a highly persuasive manner. As usual, you didn't feel like turning him down.

After some thought, you formalized the task as follows: as the Neverland mountain has a regular shape and ends with a rather sharp peak, it can be represented as a cone whose base radius equals r and whose height equals h. The graveyard where Greg is busy excavating and his crypt can be represented by two points on the cone's surface. All you've got to do is find the distance between points on the cone's surface.

The task is complicated by the fact that the mountain's base on the ground level and even everything below the mountain has been dug through by gnome (one may wonder whether they've been looking for the same stuff as Greg...). So, one can consider the shortest way to pass not only along the side surface, but also along the cone's base (and in a specific case both points can lie on the cone's base — see the first sample test)

Greg will be satisfied with the problem solution represented as the length of the shortest path between two points — he can find his way pretty well on his own. He gave you two hours to solve the problem and the time is ticking!

Input

The first input line contains space-separated integers r and h $(1 \le r, h \le 1000)$ — the base radius and the cone height correspondingly. The second and third lines contain coordinates of two points on the cone surface, groups of three space-separated real numbers. The coordinates of the points are given in the systems of coordinates where the origin of coordinates is located in the centre of the cone's base and its rotation axis matches the OZ axis. In this coordinate system the vertex of the cone is located at the point $(0, 0, h)$, the base of the cone is a circle whose center is at the point $(0, 0, 0)$, lying on the XOY plane, and all points on the cone surface have a non-negative coordinate z. It is guaranteed that the distances from the points to the cone surface do not exceed 10^{-12}. All real numbers in the input have no more than 16 digits after decimal point.

Output

Print the length of the shortest path between the points given in the input, with absolute or relative error not exceeding 10^{-6}.

Sample test(s)

input
2 2 1.0 0.0 0.0 -1.0 0.0 0.0
output
2.000000000

input
2 2 1.0 0.0 0.0 1.0 0.0 1.0
output
2.414213562

input
2 2 1.0 0.0 1.0 -1.0 0.0 1.0
output
2.534324263

input
2 2 1.0 0.0 0.0 0.0 1.0 1.0
output
3.254470198