Codeforces Beta Round #44 (Div. 2)

A. Triangular numbers

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

A triangular number is the number of dots in an equilateral triangle uniformly filled with dots. For example, three dots can be arranged in a triangle; thus three is a triangular number. The n-th triangular number is the number of dots in a triangle with n dots on a side. $T_n = \frac{n(n+1)}{2}$. You can learn more about these numbers from Wikipedia (http://en.wikipedia.org/wiki/Triangular_number).

Your task is to find out if a given integer is a triangular number.

Input
The first line contains the single number n ($1 \le n \le 500$) — the given integer.

Output
If the given integer is a triangular number output `YES`, otherwise output `NO`.

Sample test(s)

input
1
output
YES

input
2
output
NO

input
3
output
YES

B. Coins

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

One day Vasya came across three Berland coins. They didn't have any numbers that's why Vasya didn't understand how their denominations differ. He supposed that if one coin is heavier than the other one, then it should be worth more. Vasya weighed all the three pairs of coins on pan balance scales and told you the results. Find out how the deminations of the coins differ or if Vasya has a mistake in the weighting results. No two coins are equal.

Input

The input data contains the results of all the weighting, one result on each line. It is guaranteed that every coin pair was weighted exactly once. Vasya labelled the coins with letters «A», «B» and «C». Each result is a line that appears as (letter)(> or < sign)(letter). For example, if coin "A" proved lighter than coin "B", the result of the weighting is A<B.

Output

It the results are contradictory, print Impossible. Otherwise, print without spaces the rearrangement of letters «A», «B» and «C» which represent the coins in the increasing order of their weights.

Sample test(s)

input
A>B C<B A>C
output
CBA

input
A<B B>C C>A
output
ACB

C. Crossword

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

Vasya trains to compose crossword puzzles. He can only compose crosswords of a very simple type so far. All of them consist of exactly six words; the words can be read only from top to bottom vertically and from the left to the right horizontally. The words are arranged in the form of a rectangular "eight" or infinity sign, not necessarily symmetrical.

The top-left corner of the crossword coincides with the top-left corner of the rectangle. The same thing is correct for the right-bottom corners. The crossword can't degrade, i.e. it always has exactly four blank areas, two of which are surrounded by letters. Look into the output for the samples for clarification.

Help Vasya — compose a crossword of the described type using the given six words. It is allowed to use the words in any order.

Input

Six lines contain the given words. Every word consists of no more than 30 and no less than 3 uppercase Latin letters.

Output

If it is impossible to solve the problem, print `Impossible`. Otherwise, print the sought crossword. **All** the empty squares should be marked as dots.

If there can be several solutions to that problem, print the lexicographically minimum one. I.e. the solution where the first line is less than the first line of other solutions should be printed. If the two lines are equal, compare the second lines and so on. The lexicographical comparison of lines is realized by the < operator in the modern programming languages.

Sample test(s)

input

```
NOD
BAA
YARD
AIRWAY
NEWTON
BURN
```

output

```
BAA...
U.I...
R.R...
NEWTON
..A..O
..YARD
```

input

```
AAA
AAA
AAAAA
AAA
AAA
AAAAA
```

output

```
AAA..
A.A..
AAAAA
..A.A
..AAA
```

input

```
PTC
JYNYFDSGI
ZGPPC
IXEJNDOP
JJFS
SSXXQOFGJUZ
```

output

```
JJFS....
Y..S....
N..X....
Y..X....
F..Q....
D..O....
S..F....
G..G....
IXEJNDOP
...U...T
...ZGPPC
```

D. Safe

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya tries to break in a safe. He knows that a code consists of n numbers, and every number is a 0 or a 1. Vasya has made m attempts to enter the code. After each attempt the system told him in how many position stand the right numbers. It is not said in which positions the wrong numbers stand. Vasya has been so unlucky that he hasn't entered the code where would be more than 5 correct numbers. Now Vasya is completely bewildered: he thinks there's a mistake in the system and it is self-contradictory. Help Vasya — calculate how many possible code variants are left that do not contradict the previous system responses.

Input

The first input line contains two integers n and m ($6 \le n \le 35, 1 \le m \le 10$) which represent the number of numbers in the code and the number of attempts made by Vasya. Then follow m lines, each containing space-separated s_i and c_i which correspondingly indicate Vasya's attempt (a line containing n numbers which are 0 or 1) and the system's response (an integer from 0 to 5 inclusively).

Output

Print the single number which indicates how many possible code variants that do not contradict the m system responses are left.

Sample test(s)

input
```
6 2
000000 2
010100 4
```
output
```
6
```

input
```
6 3
000000 2
010100 4
111100 0
```
output
```
0
```

input
```
6 3
000000 2
010100 4
111100 2
```
output
```
1
```

E. Cannon

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

Bertown is under siege! The attackers have blocked all the ways out and their cannon is bombarding the city. Fortunately, Berland intelligence managed to intercept the enemies' shooting plan. Let's introduce the Cartesian system of coordinates, the origin of which coincides with the cannon's position, the Ox axis is directed rightwards in the city's direction, the Oy axis is directed upwards (to the sky). The cannon will make n more shots. The cannon balls' initial speeds are the same in all the shots and are equal to V, so that every shot is characterized by only one number $alpha_i$ which represents the angle at which the cannon fires. Due to the cannon's technical peculiarities this angle does not exceed 45 angles ($\pi / 4$). We disregard the cannon sizes and consider the firing made from the point $(0, 0)$.

The balls fly according to the known physical laws of a body thrown towards the horizon at an angle:

$$v_x(t) = V \cdot cos(alpha)$$
$$v_y(t) = V \cdot sin(alpha) - g \cdot t$$
$$x(t) = V \cdot cos(alpha) \cdot t$$
$$y(t) = V \cdot sin(alpha) \cdot t - g \cdot t^2 / 2$$

Think of the acceleration of gravity g as equal to 9.8.

Bertown defends m walls. The i-th wall is represented as a vertical segment $(x_i, 0)$ - (x_i, y_i). When a ball hits a wall, it gets stuck in it and doesn't fly on. If a ball doesn't hit any wall it falls on the ground ($y = 0$) and stops. If the ball exactly hits the point (x_i, y_i), it is considered stuck.

Your task is to find for each ball the coordinates of the point where it will be located in the end.

Input

The first line contains integers n and V ($1 \le n \le 10^4$, $1 \le V \le 1000$) which represent the number of shots and the initial speed of every ball. The second line contains n space-separated real numbers $alpha_i$ ($0 < alpha_i < \pi / 4$) which represent the angles in radians at which the cannon will fire. The third line contains integer m ($1 \le m \le 10^5$) which represents the number of walls. Then follow m lines, each containing two real numbers x_i and y_i ($1 \le x_i \le 1000$, $0 \le y_i \le 1000$) which represent the wall's coordinates. All the real numbers have no more than 4 decimal digits. The walls may partially overlap or even coincide.

Output

Print n lines containing two real numbers each — calculate for every ball the coordinates of its landing point. Your answer should have the relative or absolute error less than 10^{-4}.

Sample test(s)

input

```
2 10
0.7853
0.3
3
5.0 5.0
4.0 2.4
6.0 1.9
```

output

```
5.000000000 2.549499369
4.000000000 0.378324889
```

input

```
2 10
0.7853
0.3
2
4.0 2.4
6.0 1.9
```

output

```
10.204081436 0.000000000
4.000000000 0.378324889
```

Codeforces (c) Copyright 2010-2015 Mike Mirzayanov
The only programming contests Web 2.0 platform