## Yandex.Algorithm 2011<br>Finals

# A. Domino

time limit per test: 0.5 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Little Gennady was presented with a set of domino for his birthday. The set consists of $28$ different dominoes of size $2 \times 1$. Both halves of each domino contain one digit from $0$ to $6$.

0-0 0-1 0-2 0-3 0-4 0-5 0-6
1-1 1-2 1-3 1-4 1-5 1-6
2-2 2-3 2-4 2-5 2-6
3-3 3-4 3-5 3-6
4-4 4-5 4-6
5-5 5-6
6-6

The figure that consists of $28$ dominoes is called *magic*, if it can be fully covered with $14$ non-intersecting squares of size $2 \times 2$ so that each square contained four equal numbers. Every time Gennady assembles a magic figure, some magic properties of the set appear — he wins the next contest. Gennady noticed that he can't assemble a figure that has already been assembled, otherwise someone else wins the contest.

Gennady chose a checked field of size $n \times m$ and put there rectangular chips of sizes $1 \times 2$ and $2 \times 1$. Each chip fully occupies exactly two neighboring squares of the field. Those chips do not overlap but they can touch each other. Overall the field has exactly $28$ chips, equal to the number of dominoes in the set. Now Gennady wants to replace each chip with a domino so that a magic figure appeared as a result. Different chips should be replaced by different dominoes. Determine in what number of contests Gennady can win over at the given position of the chips. You are also required to find one of the possible ways of replacing chips with dominoes to win the next Codeforces round.

## Input

The first line contains two positive integers $n$ and $m$ ($1 \le n, m \le 30$). Each of the following $n$ lines contains $m$ characters, which is the position of chips on the field. The dots stand for empty spaces, Latin letters from "a" to "z" and "A", "B" stand for the positions of the chips. There are exactly 28 chips on the field. The squares covered by the same chip are marked by the same letter, different chips are marked by different letters. It is guaranteed that the field's description is correct.

It is also guaranteed that at least one solution exists.

## Output

Print on the first line the number of ways to replace chips with dominoes to get a magic figure. That is the total number of contests that can be won using this arrangement of the chips. Next $n$ lines containing $m$ characters each, should contain a field from dots and numbers from $0$ to $6$ — any of the possible solutions. All dominoes should be different.

### Examples

| input |
| --- |
| 8 8<br>.aabbcc.<br>.defghi.<br>kdefghij<br>klmnopqj<br>.lmnopq.<br>.rstuvw.<br>xrstuvwy<br>xzzAABBy |
| **output** |
| 10080<br>.001122.<br>.001122.<br>33440055<br>33440055<br>.225566.<br>.225566.<br>66113344<br>66113344 |

# B. Superset

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

A set of points on a plane is called *good*, if for any two points at least one of the three conditions is true:

- those two points lie on same horizontal line;
- those two points lie on same vertical line;
- the rectangle, with corners in these two points, contains inside or on its borders at least one point of the set, other than these two. We mean here a rectangle with sides parallel to coordinates' axes, the so-called bounding box of the two points.

You are given a set consisting of $n$ points on a plane. Find any good superset of the given set whose size would not exceed $2 \cdot 10^5$ points.

## Input

The first line contains an integer $n$ ($1 \le n \le 10^4$) — the number of points in the initial set. Next $n$ lines describe the set's points. Each line contains two integers $x_i$ and $y_i$ ( $-10^9 \le x_i, y_i \le 10^9$) — a corresponding point's coordinates. It is guaranteed that all the points are different.

## Output

Print on the first line the number of points $m$ ($n \le m \le 2 \cdot 10^5$) in a good superset, print on next $m$ lines the points. The absolute value of the points' coordinates should not exceed $10^9$. Note that you should not minimize $m$, it is enough to find any good superset of the given set, whose size does not exceed $2 \cdot 10^5$.

All points in the superset should have integer coordinates.

### Examples

| input |
| --- |
| 2<br>1 1<br>2 2 |
| **output** |
| 3<br>1 1<br>2 2<br>1 2 |

# C. Winning Strategy

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

One university has just found out about a sport programming contest called ACM ICPC v2.0. This contest doesn't differ much from the well-known ACM ICPC, for example, the participants are not allowed to take part in the finals more than two times. However, there is one notable difference: the teams in the contest should consist of exactly $n$ participants.

Having taken part in several ACM ICPC v2.0 finals and having not won any medals, the students and the university governors realized that it's high time they changed something about the preparation process. Specifically, as the first innovation it was decided to change the teams' formation process. Having spent considerable amount of time on studying the statistics of other universities' performance, they managed to receive some interesting information: the dependence between the probability of winning a medal and the number of team members that participated in the finals in the past. More formally, we know $n + 1$ real numbers $p_0 \leq p_1 \leq \ldots \leq p_n$, where $p_i$ is the probability of getting a medal on the finals if the team has $i$ participants of previous finals, and other $n - i$ participants arrived to the finals for the first time.

Despite such useful data, the university governors are unable to determine such team forming tactics that would provide the maximum probability of winning a medal at ACM ICPC v2.0 finals on average (we are supposed to want to provide such result to the far future and we are also supposed to have an endless supply of students). And how about you, can you offer such optimal tactic? At the first stage the university governors want to know the value of maximum average probability.

More formally, suppose that the university sends a team to the $k$-th world finals. The team has $a_k$ participants of previous finals ($0 \leq a_k \leq n$). Since each person can participate in the finals no more than twice, the following condition must be true: $\forall k) a_k \leq \sum_{i=1}^{k-1}(n - 2a_i)$. Your task is to choose sequence $(a_m)_{m=1}^{\infty}$ so that the limit $\Psi$ exists and it's value is maximal:

$$\Psi = \lim_{k \to \infty} \frac{\sum_{i=1}^{k} a_i}{k}$$

As $(a_m)_{m=1}^{\infty}$ is an infinite sequence, you should only print the maximum value of the $\Psi$ limit.

## Input

The first line contains an integer $n$ ($3 \leq n \leq 100$), $n$ is the number of team participants. The second line contains $n + 1$ real numbers with no more than 6 digits after decimal point $p_i$ ($0 \leq i \leq n$, $0 \leq p_i \leq 1$) — the probability of that the team will win a medal if it contains $i$ participants who has already been on the finals. Also the condition $p_i \leq p_{i+1}$ should be fulfilled for all $0 \leq i \leq n - 1$.

## Output

Print the only real number — the expected average number of medals won per year if the optimal strategy is used. The result may have absolute or relative error $10^{-6}$.

### Examples

| input |
| --- |
| 3<br>0.115590 0.384031 0.443128 0.562356 |
| **output** |
| 0.4286122500 |

| input |
| --- |
| 3<br>1 1 1 1 |
| **output** |
| 0.9999999999 |

## Note

In the second test, no matter what participants the team contains, it is doomed to be successful.

# D. Robot in Basement

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Professor has lost his home robot yet again. After some thinking Professor understood that he had left the robot in the basement.

The basement in Professor's house is represented by a rectangle $n \times m$, split into $1 \times 1$ squares. Some squares are walls which are impassable; other squares are passable. You can get from any passable square to any other passable square moving through edge-adjacent passable squares. One passable square is the exit from the basement. The robot is placed exactly in one passable square. Also the robot may be placed in the exit square.

Professor is scared of going to the dark basement looking for the robot at night. However, he has a basement plan and the robot's remote control. Using the remote, Professor can send signals to the robot to shift one square left, right, up or down. When the robot receives a signal, it moves in the required direction if the robot's neighboring square in the given direction is passable. Otherwise, the robot stays idle.

Professor wrote a sequence of $k$ commands on a piece of paper. He thinks that the sequence can lead the robot out of the basement, wherever it's initial position might be. Professor programmed another robot to press the required buttons on the remote according to the notes on the piece of paper. Professor was just about to run the program and go to bed, when he had an epiphany.

Executing each command takes some energy and Professor doesn't want to get huge electricity bill at the end of the month. That's why he wants to find in the sequence he has written out the minimal possible prefix that would guarantee to lead the robot out to the exit after the prefix is fulfilled. And that's the problem Professor challenges you with at this late hour.

## Input

The first line contains three integers $n$, $m$ and $k$ ($3 \le n, m \le 150$, $1 \le k \le 10^5$). Next $n$ lines contain $m$ characters each — that is the Professor's basement's description: "#" stands for a wall, "." stands for a passable square and "E" stands for the exit from the basement (this square also is passable). It is possible to get from each passable square to the exit, all squares located by the $n \times m$ rectangle's perimeter are the walls. Exactly one square is the exit from the basement. The last line contains $k$ characters, the description of the sequence of commands that Professor has written out on a piece of paper. "L", "R", "U", "D" stand for commands left, right, up and down correspondingly.

## Output

Print in the output file the length of the smallest possible prefix that will lead the robot to the exit square. In other words, wherever the robot had been positioned initially, it should be positioned in the exit square **after all** the commands from the prefix are fulfilled (during doing commands the robot can come and leave the exit square, but only the last position of the robot is interesting for us). If Professor is mistaken and no prefix (including the whole sequence) can bring the robot to the exit, print "-1" (without the quotes). If there is the only passable square and it is the exit, print "0" (without the quotes).

### Examples

| input |
| --- |
| 5 5 7<br>#####<br>#...#<br>#...#<br>#E..#<br>#####<br>UULLDDR |
| **output** |
| 6 |

| input |
| --- |
| 5 5 7<br>#####<br>#.#.#<br>#...#<br>#E..#<br>#####<br>UULLDDR |
| **output** |
| -1 |

| input |
| --- |
| 5 3 2<br>###<br>#.#<br>#.#<br>#E# |

```
###
DD
```

**output**

```
2
```

# E. Leaders

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

After a revolution in Berland the new dictator faced an unexpected challenge: the country has to be somehow ruled. The dictator is a very efficient manager, yet he can't personally give orders to each and every citizen. That's why he decided to pick some set of leaders he would control. Those leaders will directly order the citizens. However, leadership efficiency turned out to vary from person to person (i.e. while person A makes an efficient leader, person B may not be that good at it). That's why the dictator asked world-famous berland scientists for help. The scientists suggested an innovatory technology — to make the leaders work in pairs.

A relationship graph is some undirected graph whose vertices correspond to people. A simple path is a path with no repeated vertices. Long and frighteningly expensive research showed that a pair of people has maximum leadership qualities if a graph of relationships has a simple path between them with an odd number of edges. The scientists decided to call such pairs of different people *leader pairs*. Secret services provided the scientists with the relationship graph so that the task is simple — we have to learn to tell the dictator whether the given pairs are leader pairs or not. Help the scientists cope with the task.

## Input

The first line contains integers $n$ and $m$ ($1 \le n \le 10^5, 0 \le m \le 10^5$) — the number of vertices and edges in the relationship graph correspondingly. Next $m$ lines contain pairs of integers $a$ and $b$ which mean that there is an edge between the $a$-th and the $b$-th vertices (the vertices are numbered starting from $1$, $1 \le a, b \le n$). It is guaranteed that the graph has no loops or multiple edges.

Next line contains number $q$ ($1 \le q \le 10^5$) — the number of pairs the scientists are interested in. Next $q$ lines contain these pairs (in the same format as the edges, the queries can be repeated, a query can contain a pair of the identical vertices).

## Output

For each query print on a single line "Yes" if there's a simple odd path between the pair of people; otherwise, print "No".

### Examples

| input |
| --- |
| 7 7<br>1 3<br>1 4<br>2 3<br>2 4<br>5 6<br>6 7<br>7 5<br>8<br>1 2<br>1 3<br>1 4<br>2 4<br>1 5<br>5 6<br>5 7<br>6 7 |

| output |
| --- |
| No<br>Yes<br>Yes<br>Yes<br>No<br>Yes<br>Yes<br>Yes |

## Note

Notes to the samples:

1) Between vertices 1 and 2 there are 2 different simple paths in total: 1-3-2 and 1-4-2. Both of them consist of an even number of edges.

2) Vertices 1 and 3 are connected by an edge, that's why a simple odd path for them is 1-3.

5) Vertices 1 and 5 are located in different connected components, there's no path between them.