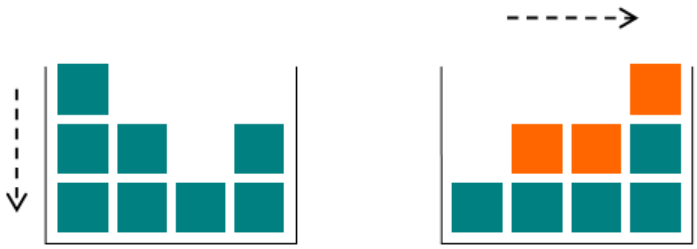
Codeforces Round #238 (Div. 2)

A. Gravity Flip

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

Little Chris is bored during his physics lessons (too easy), so he has built a toy box to keep himself occupied. The box is special, since it has the ability to change gravity.

There are n columns of toy cubes in the box arranged in a line. The i-th column contains a_i cubes. At first, the gravity in the box is pulling the cubes downwards. When Chris switches the gravity, it begins to pull all the cubes to the right side of the box. The figure shows the initial and final configurations of the cubes in the box: the cubes that have changed their position are highlighted with orange.



Given the initial configuration of the toy cubes in the box, find the amounts of cubes in each of the n columns after the gravity switch!

Input

The first line of input contains an integer n ($1 \le n \le 100$), the number of the columns in the box. The next line contains n space-separated integer numbers. The i-th number a_i ($1 \le a_i \le 100$) denotes the number of cubes in the i-th column.

Output

Output n integer numbers separated by spaces, where the i-th number is the amount of cubes in the i-th column after the gravity switch.

Sample test(s)

input
4 3 2 1 2
output
1 2 2 3

input
3 2 3 8
output
2 3 8

Note

The first example case is shown on the figure. The top cube of the first column falls to the top of the last column; the top cube of the second column falls to the top of the third column; the middle cube of the first column falls to the top of the second column.

In the second example case the gravity switch does not change the heights of the columns.

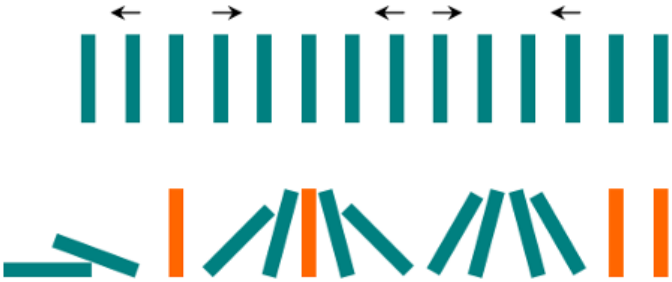
B. Domino Effect

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

Little Chris knows there's no fun in playing dominoes, he thinks it's too random and doesn't require skill. Instead, he decided to play *with* the dominoes and make a "domino show".

Chris arranges n dominoes in a line, placing each piece vertically upright. In the beginning, he simultaneously pushes some of the dominoes either to the left or to the right. However, somewhere between every two dominoes pushed in the same direction there is at least one domino pushed in the opposite direction.

After each second, each domino that is falling to the left pushes the adjacent domino on the left. Similarly, the dominoes falling to the right push their adjacent dominoes standing on the right. When a vertical domino has dominoes falling on it from both sides, it stays still due to the balance of the forces. The figure shows one possible example of the process.



Given the initial directions Chris has pushed the dominoes, find the number of the dominoes left standing vertically at the end of the process!

Input

The first line contains a single integer n ($1 \le n \le 3000$), the number of the dominoes in the line. The next line contains a character string s of length n. The i-th character of the string s_i is equal to

- "L", if the i-th domino has been pushed to the left;
- "R", if the i-th domino has been pushed to the right;
- ".", if the i-th domino has not been pushed.

It is guaranteed that if $s_i = s_j =$ "L" and $i < j$, then there exists such k that $i < k < j$ and $s_k =$ "R"; if $s_i = s_j =$ "R" and $i < j$, then there exists such k that $i < k < j$ and $s_k =$ "L".

Output

Output a single integer, the number of the dominoes that remain vertical at the end of the process.

Sample test(s)

input
14 .L.R...LR..L..
output
4

input
5 R....
output
0

input
1 .
output
1

Note

The first example case is shown on the figure. The four pieces that remain standing vertically are highlighted with orange.

In the second example case, all pieces fall down since the first piece topples all the other pieces.

In the last example case, a single piece has not been pushed in either direction.

C. Unusual Product

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

Little Chris is a huge fan of linear algebra. This time he has been given a homework about the *unusual square* of a square matrix.

The *dot product* of two integer number vectors x and y of size n is the sum of the products of the corresponding components of the vectors. The *unusual square* of an $n \times n$ square matrix A is defined as the sum of n dot products. The i-th of them is the dot product of the i-th row vector and the i-th column vector in the matrix A.

Fortunately for Chris, he has to work only in $GF(2)$! This means that all operations (addition, multiplication) are calculated modulo 2. In fact, the matrix A is binary: each element of A is either 0 or 1. For example, consider the following matrix A:

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

The unusual square of A is equal to $(1 \cdot 1 + 1 \cdot 0 + 1 \cdot 1) + (0 \cdot 1 + 1 \cdot 1 + 1 \cdot 0) + (1 \cdot 1 + 0 \cdot 1 + 0 \cdot 0) = 0 + 1 + 1 = 0$.

However, there is much more to the homework. Chris has to process q queries; each query can be one of the following:

1. given a row index i, flip all the values in the i-th row in A;
2. given a column index i, flip all the values in the i-th column in A;
3. find the unusual square of A.

To flip a bit value w means to change it to 1 - w, i.e., 1 changes to 0 and 0 changes to 1.

Given the initial matrix A, output the answers for each query of the third type! Can you solve Chris's homework?

Input

The first line of input contains an integer n ($1 \le n \le 1000$), the number of rows and the number of columns in the matrix A. The next n lines describe the matrix: the i-th line contains n space-separated bits and describes the i-th row of A. The j-th number of the i-th line a_{ij} ($0 \le a_{ij} \le 1$) is the element on the intersection of the i-th row and the j-th column of A.

The next line of input contains an integer q ($1 \le q \le 10^6$), the number of queries. Each of the next q lines describes a single query, which can be one of the following:

- 1 i — flip the values of the i-th row;
- 2 i — flip the values of the i-th column;
- 3 — output the unusual square of A.

Note: since the size of the input and output could be very large, don't use slow output techniques in your language. For example, do not use input and output streams (cin, cout) in C++.

Output

Let the number of the 3rd type queries in the input be m. Output a single string s of length m, where the i-th symbol of s is the value of the unusual square of A for the i-th query of the 3rd type as it appears in the input.

Sample test(s)

input

```
3
1 1 1
0 1 1
1 0 0
12
3
2 3
3
2 2
2 2
1 3
3
3
1 2
2 1
1 1
3
```

output

```
01001
```

D. Toy Sum

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

Little Chris is very keen on his toy blocks. His teacher, however, wants Chris to solve more problems, so he decided to play a trick on Chris.

There are exactly s blocks in Chris's set, each block has a unique number from 1 to s. Chris's teacher picks a subset of blocks X and keeps it to himself. He will give them back only if Chris can pick such a non-empty subset Y from the remaining blocks, that the equality holds:

$$\sum_{x \in X}(x - 1) = \sum_{y \in Y}(s - y)$$

"Are you kidding me?", asks Chris.

For example, consider a case where $s = 8$ and Chris's teacher took the blocks with numbers 1, 4 and 5. One way for Chris to choose a set is to pick the blocks with numbers 3 and 6, see figure. Then the required sums would be equal: $(1 - 1) + (4 - 1) + (5 - 1) = (8 - 3) + (8 - 6) = 7$.

However, now Chris has exactly $s = 10^6$ blocks. Given the set X of blocks his teacher chooses, help Chris to find the required set Y!

Input

The first line of input contains a single integer n ($1 \le n \le 5 \cdot 10^5$), the number of blocks in the set X. The next line contains n distinct space-separated integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le 10^6$), the numbers of the blocks in X.

Note: since the size of the input and output could be very large, don't use slow output techniques in your language. For example, do not use input and output streams (cin, cout) in C++.

Output

In the first line of output print a single integer m ($1 \le m \le 10^6 - n$), the number of blocks in the set Y. In the next line output m distinct space-separated integers $y_1, y_2, \ldots, y_m$ ($1 \le y_i \le 10^6$), such that the required equality holds. The sets X and Y should not intersect, i.e. $x_i \ne y_j$ for all i, j ($1 \le i \le n$; $1 \le j \le m$). It is guaranteed that at least one solution always exists. If there are multiple solutions, output any of them.

Sample test(s)

input
3 1 4 5
output
2 999993 1000000

input
1 1
output
1 1000000

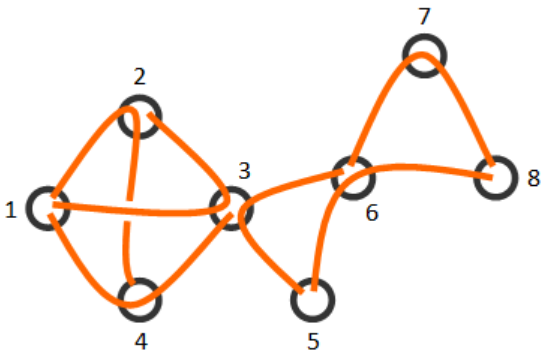
E. Graph Cutting

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

Little Chris is participating in a graph cutting contest. He's a pro. The time has come to test his skills to the fullest.

Chris is given a simple undirected connected graph with n vertices (numbered from 1 to n) and m edges. The problem is to cut it into edge-distinct paths of length 2. Formally, Chris has to partition all edges of the graph into pairs in such a way that the edges in a single pair are adjacent and each edge must be contained in exactly one pair.

For example, the figure shows a way Chris can cut a graph. The first sample test contains the description of this graph.



You are given a chance to compete with Chris. Find a way to cut the given graph or determine that it is impossible!

Input

The first line of input contains two space-separated integers n and m ($1 \le n, m \le 10^5$), the number of vertices and the number of edges in the graph. The next m lines contain the description of the graph's edges. The i-th line contains two space-separated integers a_i and b_i ($1 \le a_i, b_i \le n$; $a_i \ne b_i$), the numbers of the vertices connected by the i-th edge. It is guaranteed that the given graph is simple (without self-loops and multi-edges) and connected.

Note: since the size of the input and output could be very large, don't use slow output techniques in your language. For example, do not use input and output streams (cin, cout) in C++.

Output

If it is possible to cut the given graph into edge-distinct paths of length 2, output $\frac{m}{2}$ lines. In the i-th line print three space-separated integers x_i, y_i and z_i, the description of the i-th path. The graph should contain this path, i.e., the graph should contain edges (x_i, y_i) and (y_i, z_i). Each edge should appear in exactly one path of length 2. If there are multiple solutions, output any of them.

If it is impossible to cut the given graph, print "No solution" (without quotes).

Sample test(s)

input

```
8 12
1 2
2 3
3 4
4 1
1 3
2 4
3 5
3 6
5 6
6 7
6 8
7 8
```

output

```
1 2 4
1 3 2
1 4 3
5 3 6
5 6 8
6 7 8
```

input

```
3 3
1 2
2 3
3 1
```

output

```
No solution
```

input

```
3 2
1 2
2 3
```

output

```
1 2 3
```