A. Breadth-first Search

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

You are given a graph of n vertices. You are to find a path from vertex v_1 to vertex v_2, and you **have to** use breadth-first search. You need to code it in such a way that for each vertex it tries all adjacent to it vertices in the order of increasing indices. In other words, you need to code a "standard" breadth-first search.

The path should be restored using the "from" markers.

Input

The first line contains three integers n, v_1, v_2 ($1 \le n \le 100, 1 \le v_1, v_2 \le n$).

Each of the next n lines contain n integers each — the adjacency matrix of the graph. '1' denotes that there is an arc between these vertices, and '0' denotes that there is no arc.

Output

In the first line print the length of the path (the number of arcs in it). In the second line print the path, starting from vertex v_1 and ending with vertex v_2. If there is no such path, print -1.

Examples

input Copy

```
5 1 5
0 1 1 0 1
1 0 0 1 1
1 0 0 0 1
0 1 0 0 1
1 1 1 1 0
```

output Copy

```
1
1 5
```

input Copy

```
2 1 2
0 0
0 0
```

output Copy

```
-1
```

B. Strange Game

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

Vitya plays a game for one player. He chooses three integers n, a and b, then, starting from the integer a he wants to reach b with minimum possible number of steps. In each step he can move from integer x to one of the integers $(x+1) \bmod n$, $(x \cdot x + 1) \bmod n$ or $(x \cdot x \cdot x + 1) \bmod n$. Operation $x \bmod y$ denotes taking the remainder in division of x by y.

Your task is to determine the minimum possible number of steps and the sequence of integers that appear on the steps (including initial a and the target b).

Note that in this problem you should use 64-bit integers.

Input

The first line contains integers n, a and b ($2 \le n \le 100000$, $0 \le a, b < n$).

Output

If there is no way to reach integer b from the integer a by the steps mentioned in the statements, then print -1 in the first line.

Otherwise, print the minimum possible number of steps l in the first line. In the second line print the sequence $a = a_0, a_1, \ldots, a_l = b$ of integers in the order they appear during the steps.

Examples

input Copy

```
7 5 2
```

output Copy

```
2
5 6 2
```

input Copy

```
9 2 2
```

output Copy

```
0
2
```

input Copy

```
100000 99999 0
```

output Copy

```
1
99999 0
```

C. Clocks

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

Electronic clocks show h_1 hours and m_1 minutes. To edit the time, there are two buttons. If the first button is pressed, the hours increase by a_1, the minutes increase by b_1. If the second button is pressed, the hours increase by a_2, the minutes increase by b_2.

The clocks can only show hours from 0 to 23, so the next value after 23 is 0 again. Similarly, the next value after 59 minutes is 0 minutes. If the minutes go from 59 to 0, after some button is pressed, the hours are not increased because of that.

Find the minimum number of presses needed to set h_2 hours and m_2 minutes.

Input

The first line contains integers a_1, b_1, a_2 and b_2 ($0 \le a_1, a_2 \le 23; 0 \le b_1, b_2 \le 59$).

The second line contains integers h_1, m_1 ($0 \le h_1 \le 23; 0 \le m_1 \le 59$).

The third line contains integers h_2, m_2 ($0 \le h_2 \le 23; 0 \le m_2 \le 59$).

Output

Print the smallest number of presses needed. If there is no solution, print -1.

Examples

input Copy

```
1 0 0 1
1 10
2 1
```

output Copy

```
52
```

input Copy

```
2 5 1 1
0 0
5 11
```

output Copy

```
3
```

input Copy

```
3 58 17 8
1 59
1 0
```

output Copy

```
-1
```

D. Labyrinth

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

You are given a map of a labyrinth, your task is to find a path from a start cell to a finish cell with minimum number of moves. In one move you can go to any of 8 neighboring cells.

The map is a rectangular grid, consisting of characters '`.`' (an empty cell), '`#`' (a forbidden cell), '`S`' (start cell), '`F`' (finish cell).

You can't cross the bounds of the labyrinth, and you can't go to forbidden cells.

Let's define the direction from bottom to up as North direction, and the direction from left to right as East direction. Thus, in general, you can go from a cell to eight possible directions: "`N`", "`NE`", "`E`", "`SE`", "`S`", "`SW`", "`W`", "`NW`".

Input

The first line contains two integers n and m ($1 \le n \le 50$, $1 \le m \le 50$) — the sizes of labyrinth.

n lines follow, each consisting of m characters — the map of the labyrinth, as described in the statements. It is guaranteed that there is exactly one stard cell and exactly one finish cell.

Output

Print -1 if there is no path from start cell to finish cell. Otherwise, in the first line print k — the smallest number of steps needed, and in the next k lines print the directions that describe the path.

Examples

input Copy

```
5 5
.###.
..S..
..##.
###.#
....F
```

output Copy

```
4
E
SE
SW
SE
```

input Copy

```
4 3
###
#S#
###
.F.
```

output Copy

```
-1
```

E. Treasure Hunter

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

Vasya the treasure hunter found a map of an ancient underground dungeon. The dungeon is a labyrinth of size $n \times m$. Each cell is either empty and passable, or is a wall. From a cell you can only go to cells that share an edge with the current cell (in other words, each cell has at most 4 adjacent cells).

One of the cells contains a treasure that Vasya wants to obtain. There are k entrances in the labyrinth, and Vasya can start his path in any of them.

You are to determine which entrance Vasya should use to make the distance to the treasure as small as possible. If there are several such entrances, find the one with the minimum number.

Input

The first line contains two integers n and m ($1 \le n, m \le 1000$) that define the sizes of the labyrinth.

The next n lines with m characters in each describe the labyrinth. Character '0' denotes an empty cell, and character '1' denotes a wall. The character '*' denotes the cell with the treasure (there is exactly one such cell in the labyrinth).

The $(n+2)$-th line contains a single integer k ($1 \le k \le n \times m$) — the number of entrances to the labyrinth. After that, k lines contain the positions of the entrances. The i-th of these lines contains two integers x_i and y_i ($1 \le x_i \le n$, $1 \le y_i \le m$), meaning that the i-th entrance is located in the x_i-th row in the y_i-th column. It is guaranteed that the entrances' positions are distinct, and no entrance is located in a wall nor in the cell with the treasure.

Output

You are to print a single integer — the index of the optimal entrance (the entrances are enumerated from 1). If it is impossible to reach the treasure, print -1.

Examples

input Copy

```
5 5
00000
00000
10*00
01111
00000
4
1 1
1 5
4 1
5 5
```

output Copy

```
1
```

F. Colored Graph

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

You are given a directed graph, each arc in which is colored into one of three colors. You are to find the length of the shortest path from the vertex 1 to the vertex n, if the path can't contain two arcs of the same color in a row.

Input

The first line contains two integers n and m ($2 \le n \le 200$, $0 \le m \le n \cdot n$).

Then m lines with arcs description follow. Each arc is described with three integers x, y, c ($1 \le x, y \le n$, $1 \le c \le 3$), it is an arc from vertex x to vertex y, colored in color c. There is at most one arc in each direction between any two vertices.

Output

Print the length of the shortest path from the vertex 1 to the vertex n. If there is no such path, print -1.

Examples

input Copy

```
4 4
1 2 1
2 3 2
3 4 3
2 4 1
```

output Copy

```
3
```

input Copy

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

output Copy

```
-1
```