Harbour - BFS

A. Civilization

1.5 seconds, 256 megabytes

The world map in the "Civilization" version of the 1 computer game is a rectangle divided into squares. Each square can have one of several possible reliefs, for simplicity we will limit ourselves to three types of reliefs — field, forest, and water. The settler moves around the map, while moving to a cell occupied by a field takes one unit of time, moving to a forest — two units of time, and moving to a cell with water is impossible.

You have one settler, you have determined the place where you need to build a city in order to take over the whole world as soon as possible. Find the route of the settler, which leads him to the place of construction of the city, requiring a minimum of time. On each move, a migrant can move to a cell that has a common side with the cell where he is currently located.

Input

The input file contains two natural numbers N and M, not exceeding 1000 — dimensions of the world map (N — number of rows in the map, M — number of columns). Then the coordinates of the initial position of the settler x and y are given, where x — row number, y — row number on the map ($1 \le x \le N, 1 \le y \le M$), lines are numbered from top to bottom, columns — from left to right. Then, the coordinates of the cell where you want to bring the settler are set in the same way.

Next comes the description of the world map in the form of N lines, each of which contains M characters. Each character can be either "." (dot) for field, "W" for forest, or "#" for water. It is guaranteed that the start and end cells of the migrant's path are not water.

Output

In the first line of the output file print the number of units of time needed to move the settler (moving to a cell with a field takes 1 unit of time, moving to a cell with a forest — 2 time units). In the second line of the output file print a sequence of characters that define the migrant's route. Each character must be one of the four following: "N" (move up), "E" (move right), "S" (move down), "W" (move left). If there are several such routes, print any of them.

If it is impossible to get from the initial cell to the final one, print the number -1.

input
```
4 8 1 1 4 8
....WWWW
.######.
.#..W...
...WWWW.
```
output
```
13
SSSEENEEEEES
```

B. One hungry horse

1 second, 256 megabytes

On the chessboard $N \times N$ in the cell (x_1, y_1) there is a hungry chess knight. He wants to get into the cell (x_2, y_2), where delicious chess grass grows. What is the minimum number of moves he must make to do this?

Input

The program receives five numbers as input: N, x_1, y_1, x_2, y_2 ($5 \le N \le 20, 1 \le x_1, y_1, x_2, y_2 \le N$). The top left cell of the board has coordinates $(1, 1)$, and the bottom right cell — (N, N).

Output

In the first line print a single number K — the least necessary number of knight's moves. Each of the next $K + 1$ lines should contain 2 numbers — the coordinates of the next cell in the knight's path.

input
```
5
1 1
3 2
```
output
```
1
1 1
3 2
```

C. Beads

2 seconds, 256 megabytes

The little boy is making beads. He has many numbered beads. Each bead has a unique number — an integer in the range 1 to N. He lays out all the beads on the floor and connects the beads to each other in an arbitrary way so that no closed loops are formed. Each of the beads in this case is connected to some other bead. It is required to determine what is the maximum number of series-connected beads present in the resulting figure.

Input

The first line contains the number N ($1 \le N \le 5 \times 10^5$) — the number of beads. The next $N - 1$ lines contain two integers — the numbers of the connected beads.

Output

Output a single number — the desired number of beads.

input
```
2
1 2
```
output
```
2
```

input
```
5
2 1
2 3
2 4
2 5
```
output
```
3
```

D. Toy maze

1 second, 256 megabytes

The toy labyrinth is a transparent flat rectangular box, inside of which there are obstacles and the ball moves. The labyrinth can be tilted to the left, right, towards itself, or away from itself, after each tilt, the ball moves in a given direction to the nearest obstacle or to the wall of the labyrinth, after which it stops. The goal of the game is to drive the ball into one of the special holes — exits. The ball falls into the hole if it meets on its way (the ball does not have to stop in the hole).

Initially, the ball is in the upper left corner of the maze. It is guaranteed that the solution exists and the upper left corner is not occupied by an obstacle or a hole.

Input

The first line of the input file contains numbers N and M — dimensions of the maze (positive integers not exceeding 100). Then comes N lines of M numbers each — description of the labyrinth. The number 0 in the description means free space, the number 1 is — an obstacle, the number 2 is — a hole.

Output

Print a single integer – the minimum number of tilts you need to make for the ball to leave the maze through one of the holes.

```
input
4 5
0 0 0 0 1
0 1 1 0 2
0 2 1 0 0
0 0 1 0 0
```

```
output
3
```

E. Buses

1 second, 256 megabytes

Buses run between some villages in the Vasyuki region. Since the passenger traffic here is not very large, the buses run only a few times a day.

Maria Ivanovna needs to get from village d to village v as quickly as possible (it is assumed that at time 0 she is in village d).

Input

First enter the number N — the total number of villages ($1 \le N \le 100$), then the village numbers d and v, followed by the number of bus trips R ($0 \le R \le 10000$). The following are descriptions of bus routes. Each flight is given by the departure village number, departure time, destination village, and arrival time (all times — integers from 0 to 10000). If at time t a passenger arrives at some village, then he can leave it at any time starting from t.

Output

Print the minimum time when Maria Ivanovna can be in the village v. If she can't get from d to v using the specified bus routes, print -1.

```
input
3
1 3
4
1 0 2 5
1 1 2 3
2 3 3 5
1 1 3 10
```

```
output
5
```

F. Least multiple

1 second, 256 megabytes

Given a number X and a set of digits D. It is required to add to X the minimum number of digits from D so that the resulting number is divisible by k. In this case, the resulting number should be as small as possible.

Input

The first line of the input file contains two natural numbers X and k ($1 \le X \le 10^{1000}$, $2 \le k \le 10^5$). The second line contains the number of digits in D. The third line contains these numbers separated by a space.

Output

The only line must contain the minimum number obtained from X by adding digits from D and divisible by k. If there is no such number, print -1.

```
input
102 101
3
1 0 3
```

```
output
10201
```

G. Path in graph

1 second, 256 megabytes

In an undirected graph, it is required to find the length of the minimum path between two vertices.

Input

The first input is the number N — the number of vertices in the graph ($1 \le N \le 100$). Then the adjacency matrix is written (0 denotes the absence of an edge, 1 — the presence of an edge). Next, the numbers of two vertices — initial and final are given.

Output

You need to print the length of the path in edges. If there is no path, print -1.

```
input
5
0 1 0 0 1
1 0 1 0 0
0 1 0 0 0
0 0 0 0 0
1 0 0 0 0
3 5
```

```
output
3
```