| problem | robot | mravi | nevenka | chip |
|---|---|---|---|---|
| source file | `robot.pas`<br>`robot.c`<br>`robot.cpp` | `mravi.pas`<br>`mravi.c`<br>`mravi.cpp` | `nevenka.pas`<br>`nevenka.c`<br>`nevenka.cpp` | `chip.pas`<br>`chip.c`<br>`chip.cpp` |
| input data | library | stdin | | |
| output data | library | stdout | | |
| memory limit (heap) | 16 MB | 16 MB | 16 MB | 128 MB |
| memory limit (stack) | 2 MB | | | |
| time limit (pentium4 1.6 ghz) | 1 second | 2 seconds | 7 seconds | 4 seconds |
| points | **40** | **60** | **60** | **90** |
| | **250** | | | |

A thief Dave wish to plunder an empty house. To find out more about arrangement of rooms in the house, he teleported a robot inside it.

The robot is somewhere **inside the house** and Dave can move it by sending a simple command `Hodaj` that also tells the robot a direction (up, right, down and left). That way Dave can find all he needs aboud ground-plan of the house.

The ground-plan can be represented as a rectangular matrix consisting of small squares organized in rows and columns. Some squares represent **empty space** while other represent **walls** (i.e. places where the robot can't enter). The robot occupies one square representing floor. A room is defined as a **maximal set of floor squares mutually reachable** by left, right, up and down moves (diagonal moves are not allowed!).

Write a program that will determine **area of the room** (a number of its floor squares) where the robot is teleported using at most 10,000 calls of a function `Hodaj`. Any room area will be at most 1000.

## library

To solve the problem, you have to use a library `robotlib` that have three functions:

**`Init`** - this function should be called **only once at the beginning** of your program, without arguments. This function does not return any value.

```
procedure Init;
void Init(void);
```

**`Hodaj`** - this function has one argument – a direction where the robot should make its next move: **1 - up**, **2 - right**, **3 - down**, **4 - left**. If the move was **successful**, the function **moves** the robot to the adjacent square in desired direction and returns value 1. If the move **failed** (i.e. the robot tried to enter a wall), its position remains the same and the function returns value **0**.

```
function Hodaj(direction : longint) : longint;
int Hodaj(int direction);
```

**`Rjesenje`** - this function should be called **at the end** of your program; its only argument should be the area of the observed room. This function does not return any value and **regularly finishes execution of your program**.

```
procedure Rjesenje(area : longint);
void Rjesenje(int area);
```

Ants of negligible dimensions move with **constant speed of 1 mm/s** along tight and long wire.

When an ant encounters an obstacle (**end of the wire or another ant**), it immediately **bounces** and continues to move **in the opposite direction** with the same speed.

Initial **positions** and **directions** (left or right) of ants **are given**. The ants are marked by numbers 1, 2, 3, ..., N **from left to right**. No two ants have the same initial position.

Write a program that will determine the positions of all ants after some given time.

## input data

The first line of input file consists of two integers, L and T, $2 \leq L \leq 200{,}000$ (the length of wire) and $1 \leq T \leq 1{,}000{,}000$ (time in seconds), separated by a space character.

The second line consists of integer N, $1 \leq N \leq 70{,}000$ (number of ants), $N < L$.

Each of the following N lines consists of initial position (the distance from the left end of the wire) and direction of each ant (from the ant number 1 up to the ant number N). Initial position is given by an integer and initial direction is given by a character 'L' (left) and 'R' (right).

## output data

The only output line should consist the final positions (the distances from the left end of the wire) of all ants, starting with the ant marked with number 1 and ending with the ant number N.

## examples

**input**

```
3 5
1
1 D
```

**output**

```
0
```

**input**

```
5 5
2
2 D
4 L
```

**output**

```
1 3
```

**input**

```
8 10
5
1 L
3 L
4 D
6 L
7 D
```

**output**

```
1 2 4 7 7
```

Greg and Stef both fell in love with their classmate Miranda.

They feel irresistible urge to be in her vicinity all the time including the Sunday morning when she goes to a church.

A town where they live can be represented by an **undirected connected graph**. Miranda's house, churches where she prefers to go on Sunday mornings and pubs where Greg and Stef like to drink their beer are represented by vertices of the graph.

Distance between two vertices are defined as **the number of edges in the shortest path** (i.e. the shortest sequence of vertices where adjacent vertices are connected by an edge) connecting them.

Greg and Stef **know** where **Miranda lives** and what are **her preferred churches**.

One Sunday they went to **two different pubs**.

Miranda is acquainted with waitresses of all pubs in town and she asked them to tell her in which pubs Greg and Stef are drinking beer.

Since she dislikes them, she waited until she got the information about their whereabouts and then went to a church so that the distance between her and them would be **maximal possible** all the time. (Greg and Stef will try to come as close as possible to her in a "hunting" manner that will be explained later.)

Greg's and Stef's good friend Mike is Miranda's neighbour. He overheard when she told her parents which church she is going to that Sunday and he **immediately** told that by mobile phone to his friends. Greg and Stef leave pubs **at the same moment** Miranda leaves her home.

They all move with the speed of **one edge per minute**, although it may happen that Greg or Stef **do not move** sometimes (i.e. they are waiting at some vertices). Miranda goes along her chosen path to the chosen church (which is known to both friends) no matter what Greg and Stef are doing. They both are trying to get to her **as close as possible**. Greg and Stef stop with their hunt when Miranda enters the church.

A pair of different pubs is said to grant minimal distance X if for any church Miranda chooses and for any path she goes to it, the minimal distance to her that Greg or Stef, starting their hunt from those pubs, can achieve is X.

Write a program that will help Greg and Stef to determine **a pair of different pubs** that will grant them **the minimal possible distance X**.

## input data

The first line of input file consists of one integer V ($5 \leq V \leq 1000$), number of vertices of a graph.

The second line consists of one integer denoting vertex representing Miranda's house. The third line consists of certain number of integers determining Miranda's preferred churches. The first number ($1 \leq N \leq 200$) is number of Miranda's preferred churches which is followed by N numbers denoting vertices representing those churches. The fourth line consists of certain number of integers. The first number ($1 \leq M \leq 200$) is number of pubs that Greg and Stef like to visit which is followed by M numbers in ascending order denoting vertices representing those pubs. The fifth line consists of one integer B denoting the number of edges in the graph.

The next B lines consists of two integers V1 and V2 (V1 < V2) each denoting vertices connected by corresponding edge.

## output data

The first and only line of output file should contain two numbers (in ascending order) denoting vertices representing pubs granting minimal possible distance.

If there exist more than one solution it suffices to output any one of them.

## examples

**input**

```
14
4
3 8 9 11
3 1 7 14
13
1 2
2 3
3 4
4 5
5 6
5 7
6 8
3 9
3 10
10 11
10 12
12 13
13 14
```

**output**

```
1 7
```

**input**

```
17
17
2 4 15
3 1 9 11
18
1 2
2 3
2 10
3 4
4 5
5 6
5 17
6 7
7 8
8 9
10 11
10 17
11 12
12 13
13 14
14 15
15 16
16 17
```

**output**

```
9 11
```

A **two dimensional square plate** is located in a rectangular coordinate system so that its **lower left corner** coincides with the **origin** and **upper right corner** coincides with a point **with integer coordinates**.

Some points on the plate **are marked**. Each marked point has to be connected to a side of the plate by exactly one horizontal or vertical line. **None of those lines can contain any other of given points** on the plate and **no two lines may intersect**.

Write a program that will find described lines for given set of points on a plate so that **the sum of their lengths will be minimal possible**.

## input data

The first line of input file consists of an integer A ($2 \leq A \leq 30$), length of a plate side.

The second line consists of an integer N ($1 \leq N \leq 100$), number of given points on the plate.

Each of the following N lines consists of two integers X and Y ($1 \leq X,Y \leq A-1$) determining coordinates of a point on the plate. There will be no two different lines consisting of the same pair of coordinates (i.e. no two points will coincide).

## output data

The first line of output file should contain **the sum of lengths** of lines described above.

Each of the following N lines should contain one of the words '**GORE**' (up), '**DOLJE**' (down), '**LIJEVO**' (left) or '**DESNO**' (right) determining a side of the plate to which corresponding point is connected by a line.

If there are more than one solution, output **any one**.

If a solution does not exist then first and only line should contain integer **-1**.

## examples

**input**

```
6
4
3 1
3 5
5 4
1 4
```

**output**

```
4
DOLJE
GORE
DESNO
LIJEVO
```

**input**

```
10
5
8 1
4 3
1 2
3 9
8 5
```

**output**

```
8
DOLJE
DOLJE
LIJEVO
GORE
DESNO
```

**input**

```
10
4
5 1
5 2
4 3
6 3
```

**output**

```
13
DOLJE
DESNO
DOLJE
DESNO
```