PRACTICE SESSION – OVERVIEW

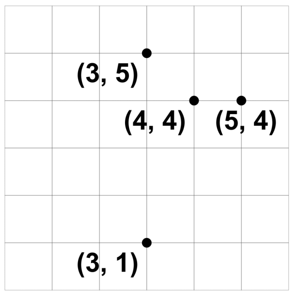
TASK	CHIP	FISH	ROBOT
type	output-only	batch	interactive
time limit (per test run)	–	1 second	1 second
memory limit (per test run)	–	64 MB	64 MB
points	**100**	**100**	**100**
	300		

CHIP

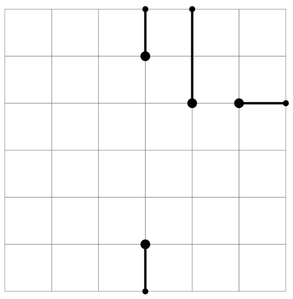
A chip is being produced on a square silicon plate.

The chip contains a number of power junctions, each described with a pair of integer coordinates. The first coordinate increases from left to right, while the second increases bottom up. The lower left corner of the chip is marked (0, 0).

For the chip to function properly, each power junction must be **connected to one of the four sides** of the chip using a **single straight** horizontal or vertical wire segment. Additionally, no two wires may overlap, intersect, or even touch.



Sample 6x6 chip with 4 power junctions



Connecting the power junctions to the chip's sides using 5 units of wire

TASK

You will be given the length of the sides of the chip and the locations of all power junctions. Find a way to connect the junctions to the sides, so that the **total length of wire used is the smallest possible**.

This is an output-only task. You will be given 10 input files and only need to produce the matching output files. You may download the input files from the contest system, on the page labeled "Tasks".

You need to submit each output file separately on the contest system. When submitting, the contest system will check the format of your output file. If the format is valid, the output file will be graded; otherwise, the contest system will report an error.

INPUT

The first line of input contains a single integer A ($2 \le A \le 30$), the length of the side of the chip.

The second line contains an integer N ($1 \le N \le 50$), the number of power junctions.

Each of the following N lines contains two integers X and Y ($1 \le X, Y \le A-1$), the coordinates of a power junction. No two power junctions will occupy the same position.

You may assume that there exists a solution for each input file.

OUTPUT

The first line of the output should contain the total length of wire used.

The following N lines should describe the connections. For each power junction, in the order in which they were given in the input, output one of `"up"`, `"down"`, `"left"` or `"right"`, the direction in which wire runs from that power junction.

If there is more than one optimal solution, output any one of them.

EXAMPLES

input

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

output

```
5
down
up
right
up
```

input

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

output

```
13
down
right
down
right
```

FISH

In a small coastal country, all towns are situated on a long coastline (which we will model as a straight line). A long straight road runs along the coast, connecting the towns. The position of each town can be described by a single non-negative integer – the distance (in kilometers) from the start of the road.

Most of the citizens are fishermen, and they catch great amounts of fish. After the fishing season is over and before the tourist season starts, the fish can be **transported** between different towns. A town can accommodate X tourists if it has X tons of fish available. The goal is to accommodate the **largest possible number of tourists** while **distributing them evenly** between towns. In other words, we want to find the **largest integer Y** for which it is possible to distribute fish so that each town can accommodate **at least Y tourists**.

In one shipment, an **integral** number of tons of fish is sent from one town to another. During transportation, **one ton of fish per kilometer traveled** is **lost** to hungry pillagers descending from the mountains. More formally, if a town ships F tons of fish to another town that is D kilometers away, then F-D tons will arrive at the destination; if F is less than D, then the entire shipment is lost.

It is possible to arbitrarily repackage and combine shipments in intermediate towns. For example, we can send shipments from towns A and B to town C, combine half of the remaining fish from both shipments with the fish originating in C and send it in a single large shipment from town C to town D.

TASK

Write a program that, given the positions of all towns and the amount of fish each town produces, determines the largest number of tourists that can be accommodated by each city after the fish has been distributed.

INPUT

The first line of input contains an integer N, $1 \leq N \leq 100\,000$, the number of towns.

Each of the following N lines contains two integers P and F, $0 \leq P, F \leq 10^{12}$, the position of a town (in kilometers) and the amount of fish it produces (in tons). The towns will be sorted in ascending order of position. The positions of all towns will be distinct.

OUTPUT

The first and only line of output should contain the largest number of tourists Y from the task description.

GRADING CRITERIA

In 50% of all test cases, N will be at most 100 and each town will produce at most 100 tons of fish.

DETAILED FEEDBACK WHEN SUBMITTING

Your first 10 submissions for this task will be evaluated during the contest (as soon as possible) on part of the official test data. After the evaluation is done, a summary of the results will be available on the contest system.

EXAMPLES

input

```
3
1 0
2 21
4 0
```

output

```
6
```

input

```
3
5 70
15 100
1200 20
```

output

```
20
```

input

```
4
20 300
40 400
340 700
360 600
```

output

```
415
```

ROBOT

Dave wishes to plunder an empty house. In order to discover the exact layout of the target house, he has teleported a remote-controlled robot inside.

The house is modeled as a grid composed of unit squares. Some squares are empty while others represent walls. The robot is initially **somewhere inside the house**, in an **empty square**.

Dave can **move the robot** by sending a simple command – the direction to move in (up, down, left or right). The robot can only move into an empty square; if it tries to move into a square occupied by a wall it will remain at its original location. After each command, it **reports back** whether move was successful or not.

TASK

Write a program that, given the ability to communicate with the robot, **determines the area of the room** into which the robot was teleported. The area of the room is the number of empty squares reachable from the robot's initial position, including the starting square. The area will be at most 1000 and you are allowed to use at most 5000 move commands.

INTERACTION

This is an interactive task. Your program sends commands to the robot using the standard output, and receives feedback from the robot by reading from the standard input.

To move the robot, you should output a single line containing one of the following commands: "`up`", "`down`", "`left`" or "`right`" (without the quotation marks), the direction in which the robot should move.

The robot will respond with a single line containing the word "`ok`" if the move was successful or the word "`fail`" otherwise.

When your program has found the solution, it should output a single line containing the area of the room and terminate its execution.

In order to interact properly with the grader, your program needs to flush the standard output after every write operation; the provided code samples show how to do this.

CODE SAMPLES

Code samples in all three languages are available for download on the "Tasks" page of the contest system. The code samples assume the room is rectangular and separately calculate the width and height of the room. This is not the correct solution and it will not score all points; the purpose of the samples is to show how to interact with the robot.

EXAMPLE

In the following example, commands are given in the left column, row by row. Feedback is given in the second column of the corresponding row.

output (command)	input (feedback)
up	fail
left	fail
right	ok
up	fail
right	ok
down	fail
up	fail
right	fail
left	ok
down	ok
down	fail
left	ok
left	fail
down	fail
5	

TESTING

You can use the TEST facility of the grading system to automatically run your solution with custom test cases. A single test case should be formatted as follows.

The first line of the test case should contain two integers R and C ($1 \le R, C \le 500$), the number of rows and columns, respectively.

The following R lines should contain C characters each. The character '.' represents an empty square, the character '#' represents a wall and the uppercase letter 'R' represents the starting location of the robot. There must be exactly one 'R' character on the map, and it needs to be inside of a room (i.e. it should be impossible for the robot to reach the edge of the map using valid moves).

Here is an example of a valid input file (corresponds to the example above). Notice that the robot is inside a room and that there is no way for it to move off the map.

```
6 8
........
.#####..
.#R..#..
.#..####
.###....
..#.....
```

Valid input for test facility

The grading system will provide you with a detailed log of the execution.