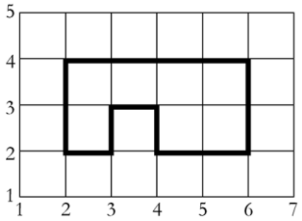
The fully automated lunar station's upkeep is the responsibility of the service robot M1RK8. The robot is moving along a **circular** track (a track forming a cycle). The track consists of **straight segments** with north, south, east, or west orientations, and in-place 90-degree **turns** to the left or right. The length of each straight segment is a positive integer number of metres, so the entire track can be shown in a Cartesian plane with segments passing through **lattice** (integer-coordinates) points and **parallel** to the coordinate axes. The track **never intersects** with itself, and no two segments touch or overlap (except, of course, at adjacent endpoints, i.e. turns).

Here is an example track with 8 turns:



A meteorite has struck the Moon near the station, so most sensors went offline. Thus, neither the **position** of the robot nor the **direction** that it is facing is currently known. We cannot begin repair work on the station until we have established the robot's position along the track.

This task will not be easy since the only command available for moving the robot is "**move K**", where **K** is the positive integer number of metres that the robot must move along the track. After executing the command, the robot returns **two integers**: **L** and **R**. The number **L** is the number of **left** turns that the robot had to make **while executing the last move command**, and **R** is the number of **right** turns.

Notes: The robot's starting position, as well as the position after each command, is always a **lattice** (integer-coordinates) point. If the robot ends the move command at a turn point, it will turn in the direction of the rest of the track **before** ending the move command, so the next move command will start with going straight.

For example, if the robot is at the coordinates (3, 3) and facing east, after executing the command "move 4" it will end up at (6, 2) facing north and return the result "2 1", meaning that it has made two left and one right turns while executing the command. After that, the command "move 1" will move the robot to (6, 3) and return "0 0".

Write a program that will determine the position of the robot using **at most 5000** commands of the form "move **K**". The program has to finish with the command "**stop X Y**", where **X** and **Y** are the **final coordinates of the robot** after executing all comands.

INTERACTION

Before interacting with the robot, the program needs to read the following data from the input:

- The first line of input contains the positive integer **N** ($6 \leq \mathbf{N} \leq 10\,000$), the number of turns on the track, which is also equal to the number of straight segments.
- The following **N** lines of input contain the coordinates of turns in order that they appear along the track, one per line, given as pairs of integers **X** and **Y** ($1 \leq \mathbf{X} \leq 100\,000$, $1 \leq \mathbf{Y} \leq 100\,000$).

Notes: The **direction** of the robot along the track can be in order that the coordinates are given, or the opposite order. The track will be such that tke robot's position is uniquely determinable, i.e. the test data will never contain two track points with different orientation that give the same result for every possible "move" command sequence.

After reading the data above, it is possible to give commands to the robot using standard output. Make sure to *flush* the output after each "move **K**" command; after flushing, you can read in the two numbers **L** and **R** from standard input. **K** must be between 1 and 1 000 000 000, inclusive.

After determining the coordinates of the robot, your program must output "stop **X Y**" to standard output, *flush* the output, and regularly finish execution.

SCORING

The following constraints will hold in a subset of the test data:

- worth a total of 40 points: **N** ≤ 100 and **X**, **Y** ≤ 50
- worth a total of 60 points: **N** ≤ 100 and **X**, **Y** ≤ 100 000
- worth a total of 80 points: **N** ≤ 2000 and **X**, **Y** ≤ 100 000

TESTING

Your solution can be tested either locally or using the evaluator system. For both methods you first need to create a file describing the desired test case.

- The first line must contain the positive integer **N**, the number of turns.
- The following **N** lines must contain the coordinates of turns in the order that they appear along the track: two integers, **X** and **Y**, for each line.
- The following line must contain the two integers **X0** and **Y0** and the word **S**. **X0** and **Y0** represent the starting coordinates of the robot, while **S** describes the robot's direction. **S** can be one of the following: "**same**" if the robot is moving along the track in the order of turns given in input, or "**opposite**" if the robot is moving in the opposite direction. The starting coordinates can be either at a turn or along a straight segment of the track.

An example of a test input file corresponding to the track example given earlier:

```
8
2 4
6 4
6 2
4 2
4 3
3 3
3 2
2 2
3 3 opposite
```

If you wish to test using the evaluator system, you first need to submit the source code using the SUBMIT page, and then send the test input file using the TEST page.

Local testing (Unix only) can be carried out using the mjesec_test file available for download from the evaluator system. The testing is done using the following command:

./mjesec_test *./solution_executable test_input_file*

Whatever testing method you choose, the output will contain feedback on whether your program has correctly solved the test case, as well as information about the commands issued by your program, responses received, and robot positions after each command.