Task: TRI
Triangles

english

CEOI 2018 Warsaw POLAND

CEOI 2018, day 2. Available memory: 256 MB. *16.08.2018*

Byteland is a nice country with n ($n \geq 3$) cities, represented as n distinct points on a 2D plane. The cities are numbered from 1 to n. As a tourist, you do not know the exact locations of the cities in Byteland. From a tourist magazine you learnt that no three cities are colinear.

The convex hull of a set of n points is a convex polygon with the smallest possible area such that all the n points are inside or on the border of this polygon. A convex polygon has all angles less than 180 degrees and cannot have self-intersections.

Your task is to find the number of vertices on the border of the convex hull of the set of Byteland cities. You may only ask questions for triples of **different** numbers of cities i, j, k ($1 \leq i, j, k \leq n$). Such a question concerns a triangle with vertices in cities i, j, k. The answer to the question indicates if traversing the vertices of the triangle in the order i, j, k is clockwise or counterclockwise.

Communication

Your program should use a library which allows asking questions and announcing the final answer.

The library (`trilib.h` for C and C++) provides the following functions:

- `int get_n();`
  Returns the number of cities.
- `bool is_clockwise(int a, int b, int c);`
  Returns `true` if the vertices of the triangle a, b, c ($1 \leq a, b, c \leq n, a \neq b \neq c \neq a$) are given in a clockwise order and `false` if they are given in the a counterclockwise order.
- `void give_answer(int s);`

For Java, the class `trilib` provides the following methods:

- `static public int get_n();`
- `static public boolean is_clockwise(int a, int b, int c);`
- `static public void give_answer(int s);`

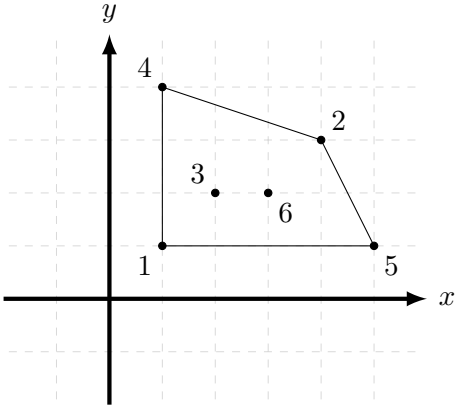
After your program calls `give_answer`, it is not allowed to ask more questions. It should call `give_answer` exactly once.

In this problem, you are not allowed to read from the standard input or to write to the standard output. After calling `give_answer`, your program should terminate immediately.

You may assume that the locations of points are established in advance and are not going to change during the execution of the program (that is, the library behaves in a completely deterministic way). For example, in the example test case (see below) calling `give_answer(4)` and immediately exiting would pass the test. Your program is allowed to try to guess the answer without being sure.

Example interaction

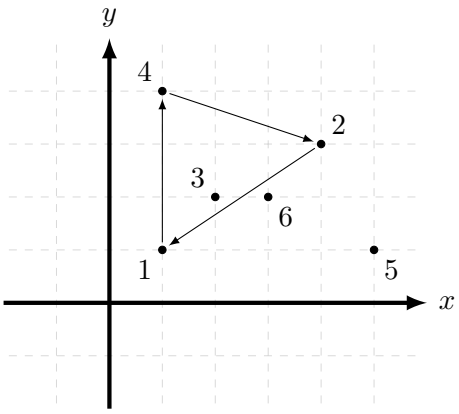
Consider $n = 6$ cities located at $(1, 1)$, $(4, 3)$, $(2, 2)$, $(1, 4)$, $(5, 1)$, $(3, 2)$ as shown in the picture below. The convex hull is marked with lines. It contains four vertices on its border, so the result is 4.



The following table shows an example interaction with a library that corresponds to this example.

Call	Returned value
`get_n()`	6
`is_clockwise(1, 4, 2)`	`true`
`is_clockwise(4, 2, 1)`	`true`
`is_clockwise(1, 2, 4)`	`false`
`is_clockwise(3, 6, 5)`	`true`
`give_answer(4)`	–

The picture below shows the triangle from the first query. The cities 1, 4, 2 are in a clockwise order, so the returned value is `true`.



Grading

The test set is divided into the following subtasks with additional constraints. Tests in each of the subtasks consist of one or more separate test groups. Each test group may contain one or more test cases.

In all tests $3 \leq n \leq 40\,000$. You may call `is_clockwise` at most 1 000 000 times.

Subtask	Constraints	Points
1	$n \leq 50$	15
2	$n \leq 500$	20
3	$n \leq 15\,000$	20
4	at most one point is not on the border of the convex hull	20
5	no additional constraints	25

Experiments

In `public` directory there is an example library allowing you to test formal correctness of your solution. The library reads a description of Byteland from the standard input in the following format:

- in the first line an integer n, the number of cities,
- in the next n lines: two integers each, the coordinates of the i-th city.

The provided library **does not** make all checks of your solution. It also does not check the correctness of the input. It is not the same as the (secret) library on the server.

An example input for the library is given in `tri0.in` file.

After `give_answer` is called, the library prints the given answer and the number of calls to `is_clockwise` to the standard output.

For compiling the solution with the example library you may use the following commands:

- **C**: `gcc -O2 -static trilib.c tri.c -lm -std=gnu99`
- **C++**: `g++ -O2 -static trilib.c tri.cpp -lm -std=c++11`

For Java you do not need to use any special command to compile the solution with the library.

The files with the solution and the library should be in the same directory.