Highway design

The Government decided to build a new highway system through ***N*** cities. Each city has already specified the desired location of its junction. The Government has the policy that highways may only be built along straight lines. Miraculously, two highways turned out to be enough to reach all cities, i.e. it is possible to define two lines, such that each of the junction points lies on one of the lines. Also, both of these lines contain at least three of the junctions, and there is no junction at the intersection of the two lines. It is easy to prove that under these conditions the lines are uniquely determined. Your company's job is to build this highway system, therefore you want to know the traces of the two lines. The coordinates of the junction points have been registered at the Land Office, but you are only allowed to submit queries to the office asking whether or not some given three junction points are collinear. Since a query has a significant processing fee (you know these bureaucrats…), you aim to ask as few queries as possible.

Since two junction points completely determine the trace of a straight line, your task is to determine two junction points for the each of the two lines.

Task

You are to write a program that determines two junctions on each of the two highways, with as few queries as possible.

Library

To perform queries, you are given the library **`office`** with three operations:

- **`GetN`**, to be called once at the beginning, without arguments; it returns ***N***, the number of cities. The corresponding junctions are numbered from ***1*** to ***N***.
- **`isOnLine`**, to be called with three junction identifiers as arguments. **`isOnLine(x,y,z)`** returns ***1*** if the junction points ***x***, ***y*** and ***z*** are collinear, and ***0*** otherwise. Note that two points are always collinear.
- **`Answer`**, to be called once at the end; it submits your solution and then properly terminates the execution of your program. **`Answer`** has four integer arguments: ***a1***, ***b1***, ***a2***, ***b2***. Here ***a1*** and ***b1*** are two junctions on one line, ***a2*** and ***b2*** are two junctions on the other line.

Instruction for Pascal programmers: include the import statement

```
    uses office;
```

in your source code.

Instructions for C/C++ programmers: use the instruction

```
    #include ”office.h”
```

Experimentation

You are provided with the source code of an implementation of the library "office" contained in the file **`sample.zip`** that you can download from the Contest System. The library reads data from the standard input. The first line must contain a single integer, the number of cities. The second line must contain space-separated integers, the identifiers of the junctions on the first highway. The junctions on the second highway are exactly the ones not listed in the second line.

Constraints

- For the number of cities ***N,*** we have 40 ≤ ***N*** ≤ 100 000.
- Pascal function declarations:
```
    function GetN: longint;
    function isOnLine(x, y, z: longint): integer;
    procedure Answer(a1, b1, a2, b2: longint);
```
- C/C++ function declarations:
```
    int GetN(void);
    int isOnLine(int x, int y, int z);
    void Answer(int a1, int b1, int a2, int b2);
```
- Your program must not read or write any file, including the standard input and output.
- Memory limit: 16 MB (the library uses no more than 1 MB)
- Time limit: 1.0 sec

Grading

The grader does not use a predetermined situation, but it will answer queries without contradicting itself. Your solution is only accepted if it is implied by the queries previously performed by your program. (There is no point in guessing.)
There are 25 test cases. If your answer is correct and the number of queries that your program has made is ***K***, then you get

- 4 points if ***K≤N/2+2,*** else
- 2 points if ***K≤2N/3,*** else
- 1 point if ***K≤N-3,*** else
- 0 point.

You may assume that the grader answers queries in a way that your program will not be able to provide a correct solution without making at least ***N/3*** queries.