Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2014

Self-Intersecting Path

`self.c`, `self.cpp`, `Self.java`

Karel wants to register his robot Karel for a robot contest. The aim of the contest is to escape from a maze using a program consisting of N instructions. Each instruction is in the form: "MOVE a_i METERS FORWARD, THEN TURN 90° TO THE RIGHT", where a_i is a positive integer. We can simply encode the whole program for the robot as the sequence of integers a_1 a_2 a_3 ... a_N representing the lengths of particular steps.

For example, if the robot starts at coordinates $[0, 0]$ facing north and the encoded program is 1 2 3 4 5, the robot would end up at coordinates $[-2, 3]$ facing east. An important property of any valid program is that the path the robot takes is not allowed to intersect itself at any point.

This sounds like a nice contest problem, doesn't it? We want to give you an idea what it is like to organize a programming contest. Therefore, your task is to write a validator for the problem described above. (You may read more about validators in the *validate* problem.)

Input Specification

The input contains several test cases. Each test case consists of two lines. The first line contains a single integer N ($1 \leq N \leq 10^6$), the number of instructions that form the robot's program. The second line contains N space-separated integers a_1 a_2 a_3 ... a_N ($1 \leq a_i \leq 10^9$), an encoded program for the robot, as described above.

Output Specification

For each test case, print exactly one line. If the given program describes a path that does not intersect itself, print "`OK`". Otherwise output a single integer M ($0 \leq M < N$), the maximum number of instructions from the beginning of the program that describe a valid path. That means the path described by the program consisting of instructions a_1 a_2 a_3 ... a_M does not intersect itself and M is maximal with this property.

Sample Input

```
7
3 1 1 3 2 2 6
3
2 1 1
6
2 1 4 4 4 3
```

Output for Sample Input

```
3
OK
5
```