Measuring Problem Difficulty

`difficult.c`, `difficult.C`, `difficult.java`, `difficult.p`

If you are the lucky one to advance to the ACM-ICPC World Finals, one of the situations you will face is the world finals competition itself. Wait, isn't that the *main* reason to go there?

In the beginning of each ACM-ICPC competition, there are two separate goals each team tries to accomplish:

1. among the given set of problems, find the easiest one,
2. solve that problem as fast as possible.

To evaluate the performance of all teams in detail, we want to test your abilities for each of these two goals separately. This problem deals with the former goal (finding the easiest problem), the latter one (solving it) is analyzed in another problem (`easy`).

The main trouble with comparing problem difficulty is that the opinions of different people may vary. To satisfy everyone, we need to find some consensus. Let's start with determining all problems on which the opinions agree.

Your team is given a set of ICPC problems. Each team member sorts all of the problems in the order of their expected difficulty. Then we want to find all pairs of problems such that their relative order is the same according to all given orderings.

Input Specification

The input contains several tasks. Each task starts by one line containing single integer N, $2 \leq N \leq 150\,000$, the number of problems to consider.

After that, there are three blocks, each of them describing opinion of one team member. (ICPC teams have three members, right?) Every block specifies an arbitrary permutation of N numbers $1 \ldots N$ representing the problems. You should know from the school that the word "permutation" means each of the numbers will appear exactly once in each block.

Each block starts on a new line. For presentation reasons, the numbers inside a block may be split among any number of lines. If there are more than one number per line, they will be separated by at least one space. Empty lines may occur before and after blocks.

The last task is followed by a line containing zero.

Output Specification

For each task, print one line containing a single integer number: the number of all pairs of problems, whose mutual order is the same in all three permutations.

Please note that the result may be as high as $\frac{N \cdot (N-1)}{2}$ and may therefore exceed 2^{32}.

Sample Input

```
4
1 2 3 4
2 3 4 1
4 1 2 3

6
3 6 1 2 4 5
3 6 1 4 2 5
3 6 1 2 4 5

0
```

Output for Sample Input

```
1
14
```