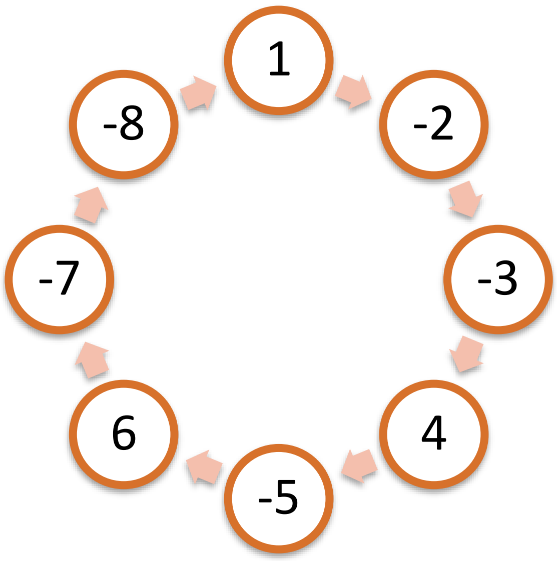
Problem 2 - Crypto Master

Let's say you have passed on the other side of the dangerous floor. Now the only thing that separates you from the real deal is the darn safe lock. Till now you had instructions for each task but not this time … this time your uncle have forgotten to leave you the combination for the safe so you have to figure out a way to unlock it! After hours of searching for any clue, the only thing you find is a sign on the side of the safe saying: "***The numbers of the code are in increasing order***". Yeah, you guessed that one - brute-force! Or maybe you haven't … anyway, time is all yours to have some fun now!

Your task is to find a brute-force algorithm to unlock the safe. You will **receive** a **sequence of numbers**, which represent the locker. This **sequence forms** a **circle**, so the **last** number is just **before** the **first** one and the **first** number is right **after** the **last** one - you have **no ending** of the sequence.

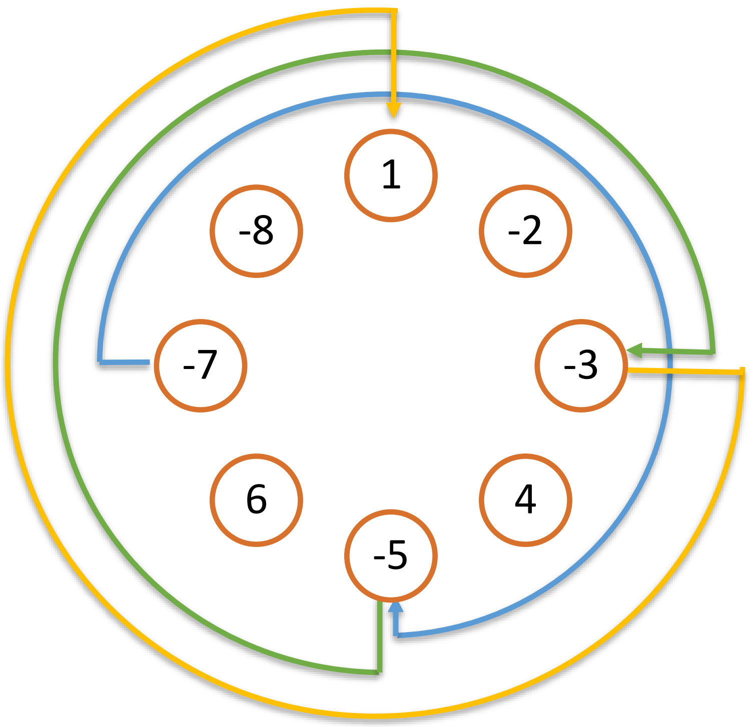


You can **start** from **any number** in the circle and **move to** another one and **choose** any **size** for the **step** from the **range below**. The **direction** must be from **left-to-right**. There are **four rules** you should keep:

- You should always jump to a **larger** number **than** the one you are **currently on**
- You **cannot jump on the same number more than once** per try
- The **size** of the jumping **step** can vary in the range **1** - the **size of** the **sequence** of numbers but must be constant within a single try
- The **size** of the **step** should be **constant** within a **single try**

Example:

In the sample above, the best route is **-7**, **-5**, **-3**, **1** with **length 4** and **step 6**.



Input

The input data should be read from the console.

On the **first line** you will be given the **numbers separated** with "**,** " (comma and space).

Output

The output data should be printed on the console.

You should print the **count of longest sequence**, which you have found.

Constraints

- **The count of numbers** will be between 1 and 2 500 inclusive.
- Each of the number will be **between -1000 <= n <= 1000**
- Allowed working time for your program: 0.2 seconds. Allowed memory: 16 MB.

Examples

Input example	**Output example**
`1, -2, -3, 4, -5, 6, -7, -8`	`4`
`1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 0`	`11`
`1, 1, 1`	`1`