# Problem 2 – Command Interpreter

Jagged arrays, regular expressions, asynchronous programming… Tough stuff. But simple structures like arrays are piece of cake, right? Let’s see how well you can manipulate data in a collection.

You will be given a **series of strings** on a single line, separated by **one or more whitespaces**. These represent the collection you’ll be working with.

On the next input lines, until you receive the command **"**end**"**, you’ll receive a **series of commands** in one of the following formats:

* **"**reverse from [start] count [count]**"** – this instructs you to reverse a **portion** of the array – just [count] elements starting at index [start];
* **"**sort from [start] count [count]**"** – this instructs you to sort a **portion** of the array - [count] elements starting at index [start];
* **"**rollLeft [count] times**"** – this instructs you to move **all** elements in the array to the left [count] times. On each roll, the first element is placed at the end of the array;
* **"**rollRight [count] times**"** – this instructs you to move **all** elements in the array to the right [count] times. On each roll, the last element is placed at the beginning of the array;

If any of the provided **indices** or **counts** is **invalid** (non-existent index or negative count), you should print a message on the console **"**Invalid input parameters.**"** and **keep the collection unchanged**.

After you’re done, print the resulting array in the following format: **"[arr0, arr1, …, arrN-1]"**. The examples should help you understand the task better.

### Input

* The input data should be read from the console.
* The first input line will hold **a series of strings**, separated by **one or more whitespaces**.
* The next lines will hold **commands** in the described formats (exactly).
* The input ends with the keyword **"end"**.
* The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

* The output should be printed on the console.
* Each time an **invalid parameters** are received, print the following line: **"**Invalid input parameters.**"**
* After receiving the "**end**" command, print the **resulting array** on the console in the format specified above.

### Constraints

* The **count of strings** in the collection will be in the range [1 … 50].
* The **number of commands** will be in the range [1 … 20].
* All commands will be in the described format; an invalid command is a command containing invalid [start] or [count], **there won’t be any missing or misspelled words**.
* [**start**] and [**count**] will be integers in the range [-231 … 231 - 1].
* Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1 2 5 8 7 3 10 6 4 9  reverse from 2 count 4  rollLeft 3 times  sort from 7 count 3  end | Invalid input parameters.  [7, 8, 5, 10, 6, 4, 9, 1, 2, 3] | The input holds 10 strings.  [1, 2, **3, 7, 8, 5**, 10, 6, 4, 9]  [7, 8, 5, 10, 6, 4, 9, 1, 2, 3]  Invalid input parameters.  Print the results. |