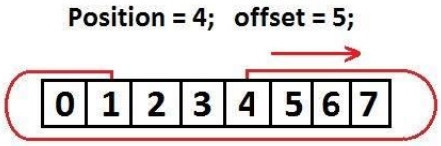
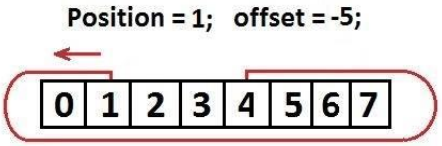
# Problem 2 – Array Slider

Fil is not a nice guy. You think that the array problem on your exam will be the easiest, but he goes out of his way to make it harder. Also, Fil likes people who follow orders blindly, so he'll give you an array of integer numbers and a series of operations to perform on it.

On the first input line you'll be given some **non-negative integer numbers separated by whitespaces (one or more)**.

On the following lines, until the "**stop**" command is received, you'll receive commands which will describe what you need to do. The commands will be in **format "[offset] [operation] [operand]"**. Offset will be a number which shows which element you need to manipulate. **You start with the element at position 0 and add the offset at each step**. If you receive the command "2 * 2", this means you need to work with element at position 2 = that is 0 (initial index) + 2 (offset). As the next command, you receive "-2 * 2", this means you need to operate on the element at index 0 = 2 (current index) + -2 (offset). If you receive a **positive offset and end up out of range**, **start at the beginning**. The same applies for **negative offsets; if you end up with target index less than 0, start at the end of the array**.




Keep track of the current index, you start at 0, but **every offset is applied to the index you were manipulating previously**.

The **operation** will be one of the following symbols: **'&' (bitwise AND), '|' (bitwise OR), '^' (bitwise XOR), '+' (add), '-' (subtract), '*' (multiply), '/' (divide)**. The **operand** will be a **non-negative integer number**. What you need to do is: **1) find the target index** following the rules described above; **2) perform the operation** on the selected element. Do all the math, follow your orders, and **print the array on the console** when you're done in format: **"[arr0, arr1, …, arrN]"**.

**The resulting array should contain only non-negative numbers**! This means that when **subtracting**, if you end up with a negative number, the resulting number should be 0. All other operations should result in **non-negative** numbers.

## Input

- The input data should be read from the console.
- The first input line will hold **a series of integers**, separated by **one or more whitespaces**.
- The next lines will contain **commands** in format **"[offset] [operation] [operand]"**.
- On the final input line you'll receive the **command "stop"**.
- 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.
- After all commands have been processed, print **the final array in format "[arr1, arr2, …]"**. No negative numbers are allowed in the output!

## Constraints

- The **count of integers** in the collection will be in the range [1 … 50].
- The **number of commands** will be in the range [1 … 20].
- **All numbers in the input array** and the **operands** will be integers in the range [0 … $2^{31}$ - 1].
- The **offset** will be in the range [-$2^{31}$ … $2^{31}$ - 1].
- Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

## Examples

| Input | Output | Comments |
|---|---|---|
| 1 2 3 4 5<br>2 * 2<br>2 / 2<br>1 - 2<br>-2 & 1<br>stop | [0, 2, 6, 0, 2] | Start at index 0. First command has offset 2, we end up at index 2 (which is the number 3). Multiply it by 2 to get 6. Next command has offset 2; current index is 2, so target index is 2 + 2 = 4 (which is the number 5). Divide by 2 (integer division) to get 2. Next command has offset 1, current index is 4, target index is 5 (1 + 4) which ends up out of bounds, so we start at 0 again, thus target index is 0 which is the number 1. Subtract 2 to get -1, which is not valid, so it becomes 0 instead. Final command has offset -2, target index is -2, which is out of bounds, so start at the end of the array to arrive at index 3 which is the number 4. Apply bitwise AND with 1 which results with 0 at this position. Final array is [0, 2, 6, 0, 2]. |