# More Exercises: Arrays

Problems for exercises and homework for the [“Programming Fundamentals” course @ SoftUni](https://softuni.bg/courses/programming-fundamentals).

You can check your solutions here: <https://judge.softuni.bg/Contests/422>.

## Last 3 Consecutive Equal Strings

Read an array of **strings** and find the **last three** consecutive equal strings.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| one one one one two hi hi my echo **last last last** pi | last last last |
| Gosho had a **little little little** lamb | little little little |
| no shirt shirt shirt no **shoes shoes shoes** no service | shoes shoes shoes |

## Array Elements Equal to Their Index

Read an array of **integers** and extract the elements which have the **same value** as their **index**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 **1** 1 1 **4** 2 1 3 **8** **9** 5 | 1 4 8 9 |
| 2 5 **2** 24 254 **5** **6** | 2 5 6 |
| **0 1 2 3 4 5 6 7 8** 8 | 0 1 2 3 4 5 6 7 8 |

## Phonebook

You will be given an array of **phone numbers** (strings) and an array of **names** (strings). After which, you will be given **strings on new lines**, **until** you receive the command “done”. Find the **numbers**, which correspond to the **names** and print them on the console in the following format:

* {name} -> {number}

A number corresponds to a name when it’s located on the **same position** as its corresponding **name** in both arrays.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 0888888888 0888123456 +359886001122  Nakov Ivan Maria  Ivan  Maria  done | Ivan -> 0888123456  Maria -> +359886001122 |
| (602)207-2544 +918671613200 1-661-335-8202  John Prakash Felix  Felix  done | Felix -> 1-661-335-8202 |
| 0882983983 0822223321 0898128398 0878223211  Vasil Hristo Georgi Luben  Vasil  Luben  done | Vasil -> 0882983983  Luben -> 0878223211 |

## \* Phone

This is an upgrade of the previous problem. Implement the **phonebook** functionality from the previous problem with this additional functionality:

* call {number/name} -> print “calling {name/number}…”
  + If the **sum of the digits** (ignore other characters) of the number is **odd**, print “no answer”.
  + If the **sum of the digits** (ignore other characters) of the number is **even**, print “call ended. duration: {duration}”. The duration is calculated from the **sum of the digits** in the format “mm:ss”
* message {number/name} -> print “sending sms to {name/number}...”
  + if the **difference** **of the digits** (ignore other characters) of the number is **odd**, print “busy”
  + if the **difference of the digits** (ignore other characters) of the number is **even**, print “meet me there”

### Constraints

* The numbers will contain the digits 0-9 and might contain the plus sign +. They will **always be valid** and in this format.
* The numbers and names will be **unique** – there will be no cases of **repeated numbers** or **repeated names**.
* The names and numbers will **always** be separated by exactly **one** whitespace.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 0888888888 0888123456 +359886001122  Nakov Ivan Maria  call 0888888888  call Ivan  message Maria  done | calling Nakov...  call ended. duration: 01:12  calling 0889123456...  no answer  sending sms to +359886001122...  busy |
| (602)207-2544 +918671613200 1-661-335-8202  John Prakash Felix  call Prakash  call John  call Felix  done | calling +918671613200...  call ended. duration: 00:44  calling (602)207-2544...  call ended. duration: 00:32  calling 1-661-335-8202...  no answer |
| 0882983983 0822223321 0898128398 0878223211  Vasil Hristo Georgi Luben  message Vasil  message 0878223211  message Hristo  done | sending sms to 0882983983...  meet me there  sending sms to Luben...  meet me there  sending sms to 0822223321...  busy |

## Char Rotation

You are given **two lines of input**: a string and an int array. Modify the elements in the first array by the second array, following these conditions:

* If the element at the position is **even**, **subtract** its value to the ASCII code of the character on the same position.
* If the element at the position is **odd**, **add** its value to the ASCII code of the character on the same position.

After that, **print** the resulting string.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| $wM+J\_H  47 8 25 73 11 15 33 | SoftUni |
| tszL  18 10 8 21 | bira |
| luFqfd~}  6 2 41 14 3 1 10 4 | fsociety |

### Hints

* You can use the string.ToCharArray() or the string.Split('') methods to turn the initial input from a string into an array.

## \* Power Plants

Ever heard of **Power Plants**? Power Plants are a special type of plant which collectively withers and blooms in strange ways.

You will receive an **array** of **integers**, holding the plants’ **power** **level**. If a plant’s power reaches **0**, it **dies**.

The plants get more and more withered each **day**, and so – their power **decreases** by 1, except for one special plant, which **blooms** instead. The special plant which blooms is the one whose **index** corresponds to the **current day**. Its power is increased by 1, but only if it **isn’t dead** (but since it has withered the same day, realistically its power stays the same).

When a season ends, all plants which are still alive **bloom**. The season ends when the **Nth** day is reached, **N** being the total number of plants.

Your task is to calculate how many days and seasons it will take for all the plants to **die**. Print them on the console in the following format: “survived {days} days ({seasons} seasons)”

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1 2 3 4 | survived 6 days (1 season) | x -> day index, y -> withering  initial: 1 2 3 4  day 0: 1 1 2 3  day 1: 0 1 1 2  day 2: 0 0 1 1  day 3: 0 0 0 1  season over: 0 0 0 2 (alive ++)  day 4: 0 0 0 1  day 5: 0 0 0 0  days survived 🡺 6, seasons 🡺 1 |
| 3 3 3 3 | survived 4 days (0 seasons) | x -> day index, y -> withering  initial: 3 3 3 3  day 0: 3 2 2 2  day 1: 2 2 1 1  day 2: 1 1 1 0  day 3: 0 0 0 0  days survived 🡺 4, seasons 🡺 0 |
| 10409 32 2 | survived 31224 days (10407 seasons) |  |

## \*\* Resizable Array

You will be given a series of commands on the console. Create an empty**,** 4 element **integer array**. Until you receive the command "end", process the following commands:

* **push {number}** – **add** the number to the **end** of the array.
* **pop** – **remove** the **last** number from the array
* **removeAt {index}** – **remove** the number at the given **index** and **shift** all elements to its right to the **left** to fill its room.
  + example: [1 2 3 4 5] -> removeAt 2 -> [1 2 4 5]
* **clear** – **erase** all the contents of the array

If at any point, the array becomes too small to hold the elements (such as when adding too many elements), create a new array at **twice the size** of the original one and fill it up with the elements of the original array. Then continue executing the commands. After you receive the “end” command, **print** the array in its final state, **separating** the elements by a **single space**. If there are no elements left in the array, print “empty array”

*Note: for this problem, you are not allowed to use the* ***List<T>*** *data structure or any similar resizable array solutions.*

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| push 1  push 2  push 3  push 4  push 5  removeAt 2  pop  end | 1 2 4 |
| push 2  push 4  push 3  push 8  push 2  clear  push 7  push 7  end | 7 7 |
| push 2  push 4  push 12  push 33  push 150  clear  push 11  push 12  removeAt 1  removeAt 0  end | empty array |

### Hints

* Think about how you would implement the removal of elements. For instance, if you **set them to** **0**, you could potentially get them **mixed up** with other **legitimate elements** that have a value of 0. Search the internet for a structure, which allows having an integer which can be set to **null**.