# Exercise: Type System

[Problems for in-class lab for the "TypeScript Advanced" course @ SoftUni.](https://softuni.bg/trainings/2696/typescript-advanced-december-2019)

# 1. Sort Array

Write a function that **sorts a numeric array** in **ascending** or **descending** order, depending on an **argument** that is passed to it.

You will receive a **numeric array** and a **string** as arguments to the first function in your code.

* If the second argument is asc, the array should be sorted in **ascending order** (smallest values first).
* If it is desc, the array should be sorted in **descending order** (largest first).

### Input

You will receive a **numeric array** and a **string** as input parameters.

### Output

The output should be the **sorted array**.

### Examples

|  |  |
| --- | --- |
| Input | Output |
| [14, 7, 17, 6, 8], 'asc' | [6, 7, 8, 14, 17] |
| [14, 7, 17, 6, 8], 'desc' | [17, 14, 8, 7, 6] |

1. **Change Array**

Write a program, which reads an **array** of **numbers** and **commands**, which **manipulate** the array. Your program may receive the following commands:

* Delete {element} – delete all elements in the array, which are equal to the given element.
* Insert {element} {position} – insert an element and the given position.

You should stop the program when you receive the command "end". Print the numbers in the array separated by a **single** whitespace.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 2 3 4 5 5 5 6  Delete 5  Insert 10 1  Delete 5  end | 1 10 2 3 4 6 |
| 20 12 4 319 21 31234 2 41 23 4  Insert 50 2  Insert 50 5  Delete 4  end | 1. 2 50 319 50 21 31234 2 41 23 |

1. **Operations**

You will be given an **array** of **integer** numbers on the first line of input. You will be receiving operations you have to apply on the array until you receive the **"End"** command. The possible **commands** are:

* **Add {number} – add number at the end.**
* **Insert {number} {index} – insert number at given index.**
* **Remove {index} – remove at index.**
* **Shift left {count} – first number becomes last ‘count’ times.**
* **Shift right {count} – last number becomes first ‘count’ times.**

**Note: there is a possibility that the given index is outside of the bounds of the array. In that case print "Invalid index"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 23 29 18 43 21 20  Add 5  Remove 5  Shift left 3  Shift left 1  End | 43 20 5 1 23 29 18 |
| 5 12 42 95 32 1  Insert 3 0  Remove 10  Insert 8 6  Shift right 1  Shift left 2  End | Invalid index   1. 12 42 95 32 8 1 3 |

1. **Operations**

Write a function that composes an object by given properties. The input comes as an **array of strings**. Every **even index** of the array represents the **name of the food**. Every **odd index** is a **number** that is equal to the **calories in 100 grams of the given product**. Assign each value to its corresponding property and print it on the console.

The **input** comes as an **array of string** **elements**.

The **output** should be printed on the console.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['Yoghurt', 48, 'Rise', 138, 'Apple', 52] | { Yoghurt: 48, Rise: 138, Apple: 52 } |
| ['Potato', 93, 'Skyr', 63, 'Cucumber', 18, 'Milk', 42] | { Potato: 93, Skyr: 63, Cucumber: 18, Milk: 42 } |

1. **\*Racing**

Write a program to calculate the **winner of a car race**. You will receive an **array of numbers**. Each element of the array represents the **time needed to pass through that step** (the index). There are going to be **two cars**. **One** of them **starts** from the **left side** and the **other one starts from the right** **side**. **The middle index of the array is the finish line**. The **number of elements** in the array **will always be odd**. Calculate **the total time for each racer to reach the finish**, which is the **middle of the array**, and **print the winner with his total time** (the **racer with less time**). If you have a **zero in the array**, you have to **reduce the time of the racer that reached it by 20%** (**from his current time**).

Print the result in the following format **"The winner is {left/right} with total time: {total time}".**

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| 29 13 9 0 13 0 21 0 14 82 12 | The winner is left with total time: 53.8 |
| **Comment** | |
| The time of the left racer is (29 + 13 + 9) \* 0.8 (because of the zero) + 13 = 53.8  The time of the right racer is (82 + 12 + 14) \* 0.8 + 21 = 107.4  The winner is the left racer, so we print it | |