# Homework: Control-Flow

Tasks with **\*** are considered hard or involve something we still haven't mentioned in the lectures. Try to solve them nevertheless. Do not worry about constraints how big the numbers can be, length of strings and so on. As we know JavaScript handles automatically the type and length of values. All tasks are submitted to GitHub. You can use HackerRank to check your solutions but the final tasks should be in GitHub.

## Compare the Triplets

Follow the link for full details. Try to solve it with and without loops.

<https://www.hackerrank.com/challenges/compare-the-triplets>

## CamelCase

Follow the link for full details.

<https://www.hackerrank.com/challenges/camelcase>

## A Very Big Sum

Follow the link for full details.

<https://www.hackerrank.com/challenges/a-very-big-sum>

## Staircase

Follow the link for full details.

<https://www.hackerrank.com/challenges/staircase>

## Min-Max Sum

Follow the link for full details.

<https://www.hackerrank.com/challenges/mini-max-sum>

## Caesar Cipher

Follow the link for full details.

<https://www.hackerrank.com/challenges/caesar-cipher-1>

## \* Super Reduced Strings

Follow the link for full details.

<https://www.hackerrank.com/challenges/reduced-string>

## Find the Largest from Undefined Count of Numbers

Find the largest number; **the input numbers might contain negatives**.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 3  7  -4  5 | Max: 7 |  | -3  -2  -17  -33  -1 | -1 |

## \*Count Letters

You will receive a single line from the standard input containing a word (or at least a set or characters). You need to print on the standard input **how many times each letter is found in order of the letter appearance**, in format {letter} -> {times}

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| apple | a -> 1  p -> 2  l -> 1  e -> 1 |  | appearance | a -> 3  p -> 2  e -> 2  r -> 1  n -> 1  c -> 1 |

An associative array is good for this purpose, but in JavaScript it is a bit different. Check this link – [associative arrays with JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects). It can keep each **letter as a key** and the **times it has appeared as value**. When you **find a letter for first time, set its value to 0**. This will **ensure you will never try to increment non-existing keys**. The main logic then will be “**increment the value by that key in the array**”. If it was just found, it will be 0 and incrementing it will become 1 (finding letter for first time means exactly that – it has occurred 1 time). If it was found before it will increment the old value e.g. from 1 to 2

Use **length** string property to iterate over a string.

## \*Count Letters – Sorted

As in the previous task, but the output should be **sorted by the times a letter has occurred in descending order**, then in order of appearance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| apple | p -> 2  a -> 1  l -> 1  e -> 1 |  | appearance | a -> 3  p -> 2  e -> 2  r -> 1  n -> 1  c -> 1 |

[Read about sorting an array](https://www.google.com/search?q=php+sorting+array). You should be using custom comparing function.