Homework: Functions, Conditional statements (*if, if-else, switch*), Arrays, Loops

# **Exchange numbers**

### Write a function that takes two numbers a and b. Exchange their values if the a is greater than b. As a result return a string that contains a and b, separated by space. Console log the result.

### Hints:

### Declare 3rd party (3rd variable) that will help you with value transfer

### Use string interpolation or string concatenation for (console.log()) the result

### Bonus:

### Declare values with prompt()

* Use two script (.js) files, one to declare the function and one to call the function

### Examples:

|  |  |  |
| --- | --- | --- |
| a | b | result |
| 5 | 2 | 2 5 |
| 3 | 4 | 3 4 |
| 5.5 | 4.5 | 4.5 5.5 |

# **Bonus Score**

### Write a function that applies bonus score to given score in the range [1…9] by the following rules:

* + If the score is between 1 and 3, the program multiplies it by 10.
  + If the score is between 4 and 6, the program multiplies it by 100.
  + If the score is between 7 and 9, the program multiplies it by 1000.

Return the new value.

### Bonus:

### Declare score value with prompt()

* Use two script (.js) files, one to declare the function and one to call the function
* What if user’s input is out of the desired range (1-9)? Add a second function that validates user’s input and keeps asking for a valid input ***while*** provided values are out of range

Hints:

1. Regarding the valid input, until you get a good one, you may need to wait for a **while(){}!#!!@&!**
2. You’ll need to figure out how to check if it **is N**ot **a N**umber**()** when validating!

*Examples:*

|  |  |
| --- | --- |
| *Input (prompt())* | *Output (console.log() or alert())* |
| *2* | *20 (console.log())* |
| *7* | *7000 (console.log())* |
| *‘wat?’* | *`The value you entered "wat?" is not a number. Please enter a valid number!` (alert())* |
| *0* | *`The value you entered 0 is out of scope!`(alert())* |
|  |  |

# **Play card**

[Classical play cards](https://en.wikipedia.org/wiki/Playing_card) use the following signs to designate the card face: *2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K* and *A*. Write a function that takes a string and prints "yes" if it is a valid card sign or "no" otherwise.

* Take the user input using prompt()

Hints:

1. You may want to **switch** to something exotic here, or **if** you are all **if**fy about it you can try something **else**.

Examples:

|  |  |
| --- | --- |
| Input | Output |
| 1 | no |
| A | yes |
| a | no |
| 5 | yes |
| 11 | no |
| 0 | no |

# **Largest of 5**

Write a program that finds the biggest of 5 numbers using maximum 5 if statements.

* Don’t use arrays and the built-in sorting functionality.

Examples:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| firstNumber | secondNumber | thirdNumber | fourthNumber | fifthNumber | result |
| 1 | 2 | 5 | 99 | 2 | 99 |
| 1 | 3 | -2 | 0 | -5 | 3 |
| -1 | -3 | -2 | -5 | -11 | -1 |
| -2 | -3 | -6 | 8 | 9 | 9 |
| 0 | 5 | -7 | 4 | 2 | 5 |

# **Sort 3 Numbers**

Write a function that takes 3 real numbers and returns them sorted out in descending order as a string.

Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| firstNumber | secondNumber | thirdNumber | result |
| 1 | 2 | 5 | The largest is 5, then comes 2 and the last is 1 |
| 1 | 3 | -2 | The largest is 3, then comes 1 and the last is -2 |
| -1 | -3 | -2 | The largest is -1, then comes -2 and the last is -1 |
| -2 | -3 | -6 | The largest is -2, then comes -3 and the last is -6 |
| 0 | 5 | -7 | The largest is 5, then comes 0 and the last is -7 |

# **Number as Word**

Write a program that read an integer [0-9] from prompt(), and depending on the input, shows the digit as a word (in English).

* Print "invalid input" in case entered value is not 1-9.
* Use a switch statement.

|  |  |
| --- | --- |
| input | outpit |
| 1 | one |
| 0.1 | Invalid input |
| wat | Invalid input |
| 3 | three |
| -9 | Invalid input |

# **Numbers from 1 to N**

Declare a function that takes one number as input parameter **n (n must be a positive integer larger than 1)**. The function should return a string of all numbers from **1** to **n**. The numbers should be separated by comma and whitespace (1, 2, 3, 4, 5...) except for the last which should end with a dot (1, 2, 3.)

|  |  |
| --- | --- |
| input | outpit |
| 5 | 1, 2, 3, 4, 5. |

# **Min, Max, Sum, Average of N Numbers**

Declare a function that takes an array of floating-point numbers as strings and returns a string that contains the minimal, the maximal number, the sum and the average value of all numbers (displayed with 2 digits after the decimal point).

Hints:

* What is floating-point number? 1,02 1,978 5,5
* What is an array of floating-point numbers as strings? [“1,32”, “1,00”, “3,456”]
* You’ll need to use [parseFloat](https://www.w3schools.com/jsref/jsref_parsefloat.asp)
* How to calculate sum? 1,32 + 1 + 3.456
* How to calculate average? (1,32 + 1 + 3.456 + 2 + 3)/5
* You’ll need to use [toFixed()](https://www.w3schools.com/jsref/jsref_tofixed.asp)

|  |  |
| --- | --- |
| input | outpit |
| [“2”, “5”,”1”] | Min is 1, max is 5, sum is 8, average is 2.67 |

# **Allocate(create new) array**

Write a function that allocates array of N integers, and initializes each element by its index multiplied by N. Function should return the newly allocated array.

|  |  |
| --- | --- |
| input | result |
| 5 | [0, 5, 10, 15, 20] |

# **Check if array is symmetrical**

Write a function that takes an array and checks if it’s symmetrical. Return boolean value (true or false) that depends from the outcome

|  |  |
| --- | --- |
| input | result |
| [5,3,1,1,3,5] | true |
| [2,3,3,1] | false |
| [2,1,2] | true |

# **Count elements in the array that are divisible by 7 and 3**

Write a function that takes an array of numbers and counts number of elements that are divisible by 7 and 3.

Hints:

* For a number n to be divisible by 7 and 3 it means that there is no rest after the division

Example:

21 % 7 = 0 and 21 % 3 = 0 → 21 is divisible by 7 and 3

15 % 7 = 1 and 15 % 3 = 0 → 15 is divisible by 3 but there is rest 1 when divided with 7

28 % 7 = 0 and 28 % 3 = 1 → 28 is divisible by 7 but there is rest 1 when divided with 3

# **Maximal sequence and Maximal increasing sequence \***

Write a function that takes a parameter n, which represents the number of the elements in the array (the length of the array), and will allocate (return new array). Positions in the array should be filled up by the user manually at run time through prompt window.

* Write a second function that will take an array and return the length of max sequence of repetition of a single element

Example:

|  |  |
| --- | --- |
| input | output |
| [1, 3, 4, 3, 3, 3, 3, 3, 2, 1, 5, 5] | 5 (3 repeat 5 times) |

* Write a third function that will take an array and return the length of the max increasing numbers

Example:

|  |  |
| --- | --- |
| input | output |
| [1, 3, 4, 5, 6, 7, 8, 3, 2, 1, 5, 5] | 6 |

# **Larger than neighbours**

* Write a function that takes an array and number **n (which is existing in the array)** and checks if the value at the n index is larger that it’s neighbour(s). The method should return a boolean value (true or false)

|  |  |  |
| --- | --- | --- |
| Input arr | input n | output |
| [1, 3, 4, 2, 3, 3, 3, 3, 2, 1, 5, 5] | 4 | true (4 is larger than 3 (on the left) and 2 (on the right)) |
| [1, 2, 2] | 2 | false (2 is equal to 2 on the right) |
| [1, 2, 2] | 1 | false (2 is larger than 1 on the left but equal to 2 on the right) |

Hint:

* You’ll need to handle values at the ends of the array differently
* You’ll need to find a way to **break** out of the loop cicle when you find the first equal value, (or maybe we don’t need to **break** stuff we already **return**ed)

# **Count larger than neighbors**

* Write a function that takes an array and counts how many elements in the array are larger than its neighbor(s). Return the count result

|  |  |
| --- | --- |
| Input arr | output |
| [1, 3, 4, 2, 3, 7, 3, 3, 2, 5] | 3 |

Hint:

* You’ll need to handle values at the ends of the array differently
* Hmm this task looks awfully familiar, maybe we can **reuse** some code? [who would know right? ;) ]

# **Century in a Year**

* Given a year, return the century it is in. The first century spans from the year 1 up to and including the year 100, the second - from the year 101 up to and including the year 200, etc.

Example

For year = 1905, the output should be

centuryFromYear(year) = 20;

For year = 1700, the output should be

centuryFromYear(year) = 17.

Hint:

* + What was that **Math** object we talked about in class? Maybe we can use some of its methods here

# **Adjacent elements product**

* Given an array of integers, find the pair of adjacent elements that has the largest product and return that product.

Example

For inputArray = [3, 6, -2, -5, 7, 3], the output should be

adjacentElementsProduct(inputArray) = 21.

7 and 3 (7 \* 3) produce the largest product.

Hint:

* Sometimes it’s better not to go the distance/**length**