**Exercise 1 - Numbers, loops, arrays, functions and recursion**

You will find the results of this exercise in a file named exercice1.js

Question 1A: write a function named 'factorielIt' which calculates the factorial of a number in an iterative way (with a loop)

Question 1 b: write a function named 'factorielRec' which calculates the factorial recursively

Question 1 c: write a 'factorielTableau' function that takes an array of numbers and returns the table of applications of a function at these numbers (without using map). You will test with the service of 1 b.

Question 1 d: write a function named 'factorielMap' is equivalent to the function of the 1 c using map.

1a

function factorielIt(x)

{

var factorial=1;

for(i=1; i<=x; i++)

{

factorial=factorial\*I;

}

return factorial;

}

1b

function factorielRec(x)

{

return factorielRec(x-1)\*x;

}

1c

function factorielTableau(arr)

{

var factorial=[ ];

for (var x in arr)

{ factorial.push(factorielIt(x));}

return factorial;

}

1d

function factorielMap(arr)

{

**var** factorial = **new** Map();

for (var x in arr)

{

factorial.set(x,factorielIt(x));

}

return factorial;

}

**Exercise 2 - Strings of characters, objects, anonymous functions**

You will find the results of this exercise in a file named exercice2.js

Question 2(a): write a 'countWords' function that for each word in a string, counts the number of occurrences of this word in this chain. The function will return a structured data containing these results and to easily obtain the number of occurences of a given word. Ensure that this function works on a string of at least 500 words, containing potentially punctuation.

Question 2b: write a 'WordList' constructor that takes as input a string and returns an object with the following methods: 'maxCountWord()' (resp. 'minCountWord()') that returns the word with the largest (resp. lower) number of occurrences, getWords() which returns an array of words (without duplicates) present in the original text, and 'getCount (word)', which gives the number of occurrence for a given word.

Question 2c: Add a method 'applyWordFunc()' to apply a function to WordList any every word and return a table of results.

Question 2d: call the method 'applyWordFunc' with an anonymous function returning an object for each word with like properties: the number of occurrence and its length.

2a

<p id="demo"></p>

<p id="result"></p>

<input type="text" id="paragraph" class="field",>

<button onclick="input()"><b>Submit</b></button>

<script>

function input() {

var test = document.getElementById("paragraph").value;

var arr =[];

arr = test.toLowerCase().replace(/[^a-zA-Z ]/g, "").split(/\s+/);

/\*remove all special characters: there remove anything that's not letter\*/

/\*or replace(/[&\/\\#,+()$~%.'":\*?<>{}]/g, '') --> place all characters you want to avoid here \*/

document.getElementById("demo").innerHTML = arr;

return arr;

}

function countWords(text) {

var wf=new Map();

var fw=new Map();

for (var word in text)

{

if (frequency.has(word)==false)

{

wf.set(word,1);

fw.set(1,word);

}

else

{

wf.set(word, frenquecy.get(word)+1);

fw.set(frenquecy.get(word)+1,word);

}

return frequency;

}

2b

var WordList={

function maxCountWord()

{

var frequency=[ ];

for(var i in wf)

{

frequency.push=i[2];

}

maxfre=alert(Math.max.apply(null,frequency));//not sure

return fw.get(maxfre);

}

getWords()

{

var words=[ ];

for(var i in wf)

{

words.push=i[1];

}

return words;

}

getCount (word)

{

return wf.get(word);

}

}

2c

function applyWordFunc()

**Exercise 3 - programming in JavaScript object-oriented**

You will find the results of this exercise in a file named exercice3.js

Question 3A: create a "Student" constructor that takes parameters variables "name", "firstName" and "id", so that we can write the following code:

var student = new Student("Dupond", "Jean", 1835);

Question 3B: Add a "print method" which takes no parameters, and in the console, which displays a character string that is built from the properties of the object in the form:

"student: Dupond, Jean, 1835"

Question 3 c: create a derived class ForeignStudent that allows to give a student a nationality. Add a print method that takes the result of the base class and adds nationality, as follows:

"student: Doe, John, 432, American"

Question 3d: create a module for the Student and ForeignStudent classes. Test this module to verify that you have exported what it takes to make it usable.

Question 3d: create a module for a class Promotion, which may contain any number of Students or ForeignStudent objects and with the following methods: add, remove, list, saveToFile readFromFile. Add take citizenship as the optional parameter. The last 2 methods will backup and read using the JSON format. Test this module and all of its features with an example to 20 + students.