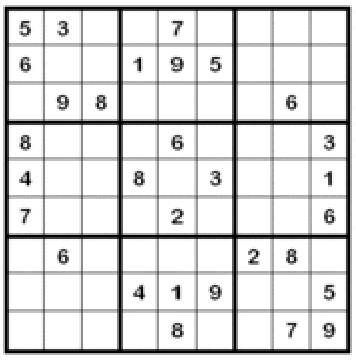
Test Subject – JAVASCRIPT – Algorithm

Type : Pratical work – Time allowed 240 mn
(All documents are authorized)

Subject

We propose to manage a table capable of containing grids of numbers (SUDOKU)

Sudoku reminder



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

A 9 × 9 grid of sudoku
The game grid is a square with nine side squares, subdivided into as many identical squares, called regions (see figure). The rules of the game are simple: each row, column and region must not contain only once all the digits from one to nine. Otherwise formulated, each of these sets must contain all the digits from one to nine

The number of different grid possibilities is 6,67.10^{21}.

Typically, a game grid comes up with some numbers unconvered, and the player must discover the other hidden figures.

➲ The objective of the test is to create a set of JAVASCRIPT components that will allow to verify, for a test table provided by the examiner in the form of a file, if the constraints listed above are actually respected.

➲ The test consists of 3 exercises that allow you to reach the set objective.

Principe de travail

☑ Each of the exercises proposed must be performed in a JAVASCRIPT file which you will name:

<your name>_<your first name>_test_javascript_algo_ex**N**.js
N being the exercise number.

You will create a main HTML script that you will name

<your name>_<your first name>_test_javascript_algo_main.htm
which will use the processing coded in the * .js files as you progress through the exercises and which you will use to test them.

Reminder: to be able to use JAVASCRIPT code contained in an external file, a line of type must be inserted in the HTML script

```
<SCRIPT LANGUAGE="Javascript" SRC="john_smith_test_javascript_algo_ex1.js"> </SCRIPT>
```
john_smith_test_javascript_algo_ex1.js in this example being the name of the file containing JAVASCRIPT code

☑ The main objective of the test is to validate your knowledge of algorithmics. Do not waste time on formatting the results as this is not the objective of the test.
You can use the concepts "object" or not, as you like. The use of forms is also not required.

☑ Particular attention will be paid by the corrector to the readability of your code, the note obtained for the exercises (out of 50) being increased according to the following criteria to give a final note out of 100:

- Name of variables (max :+10%).
- Indentation (max :+20%)
- Alignment of{ and } (max :+15%)
- Clarity and readability (max :+15%)
- Modularity and reusability (max :+10%)
- Developtment standardization (max :+10%)
- Relevant comments (max :+20%)

Note, for example, that the exercises sometimes refer to variables "A", "B", etc. and functions "F1", "F2", etc. . You can of course give them more explicit names.

☑ If you get stuck, don't hesitate to call the examiner who can help you on occasion. The examiner will recover your 4 files (exN.js + main.htm) at the end of the test.

Environmental preparation

The file provided by the reviewer is called "**javascript_test_je1.js**". It contains the data of the table to be checked in the following format:

```
var array_number = new Array(9);
array_number[0]= "4 2 7 3 5 1 9 8 6";
array_number[1]= "9 8 3 7 6 2 5 1 4";
array_number[2]= "1 5 6 8 9 4 7 2 3";
array_number[3]= "2 3 9 1 8 5 4 6 7";
array_number[4]= "7 4 1 6 3 9 2 5 8";
array_number[5]= "5 6 8 2 4 7 1 3 9";
array_number[6]= "8 7 2 9 1 3 6 4 5";
array_number[7]= "3 9 5 4 2 6 8 7 1";
array_number[8]= "6 1 4 5 7 8 3 9 2";
```

Of course, the exact value of the figures in the table may vary from this example, but the structure will remain exactly the same.It is a 1-dimensional array, the numbers in each row being separated by whites.

The table is assumed to always be correct and you do not have to check it.

You must refer to this file in your JAVASCRIPT code and not copy it into your program.

During your tests, you will be able to use a personal file concerning your own table, formatted by yourself and of the same structure, but containing anomalies in order to check the correct functioning of the program.

*Exercise 1 (10 points) : (Reminder : to be done in a specific *.js file)*

1- (1pt) In this script, create a "to_verify" JAVASCRIPT table of 9 boxes out of 9.

2- (4pt) Insert a JAVASCRIPT "F11" function allowing you to read the table which has been provided to you (table_digits, 1 dimension) and to transfer it to the table defined in 1 ("to_check", 2 dimensions).

3- (3pt) Then insert a function "F12" allowing to display the content of the table "to be verified", in the form of an HTML table (9x9).

4- (2pt) Run the script via your HTML file "Main" which will use the 2 functions and check the display.

Example of expected display :

Table to check :

4	**9**	**5**	**2**	**7**	**3**	**6**	**8**	**1**
6	**1**	**3**	**9**	**4**	**8**	**7**	**5**	**2**
8	**2**	**7**	**5**	**6**	**1**	**4**	**3**	**9**
7	**6**	**1**	**8**	**2**	**4**	**5**	**9**	**3**
9	**5**	**8**	**6**	**3**	**7**	**2**	**1**	**4**
2	**3**	**4**	**1**	**5**	**9**	**8**	**6**	**7**
1	**7**	**9**	**4**	**8**	**5**	**3**	**2**	**6**
3	**8**	**2**	**7**	**9**	**6**	**1**	**4**	**5**
5	**4**	**6**	**3**	**1**	**2**	**9**	**7**	**8**

*Exercise 2 (10 points) : (Reminder : to be done in a 2nd specific *.js file)*

1- (2pt) Create a JAVASCRIPT "F21" function which accepts 1 input parameter "A" of table type with 9 positions as input.

2- (7pt) Within the function, insert the necessary JAVASCRIPT code making it possible to verify that all the positions are numbers (from 1 to 9) and all different from each other.

3- (1pt) The function must return true if the input array is good, false otherwise .

*Exercise 3 (20 points) : (reminder: to be done ine a 3rd specific *.js file)*

1- (4pt) Create a new function "F31" which calls the previous function "F21" for each line of the table "to_check". Display a relevant error message in the event of an anomaly, indicating in particular the line number in error and the values of the stations on the line. (see question n ° 4 and example of the anomaly table in the annex).

2- (4pt) Repeat the previous operation with a "F32" function which processes each **column** in the "checkbox" table. (see question n ° 4 and example of the anomaly table in the annex).

3- (4pt) Repeat the operation again with a function "F33" which deals with each **region** of the table "checkbox". (see question n ° 4 and example of the anomaly table in the annex)

Table regions :

	1			2			3	
	4			5			6	
	7			8			9	

4- (1+2+3pt) "F21" accepting an array as input, for each of questions 1 to 3 you will need to find a simple algorithm allowing you to put the values of the boxes to be checked in the form of a table of 9 positions before calling this function.

5- (2pt) Execute these functions via your HTML file "Main" which will use the 3 functions and check the display. (also leave the call to the functions created in exercise 1).

Example of expected display (if anomaly)

Line 2 incorrect	6	6	3	2	8	7	4	5	9
Column 3 incorrect	3	3	2	6	5	9	1	8	7
Region 1 incorrect	8	2	3	6	6	3	9	7	2

Display example expected otherwise

➔ the table is correct