BTI225 Assignment 1

Submission Deadline:

Thursday, February 9th, 2017 @ 11:59 PM

Assessment Weight:

5% of your final course Grade

Objective:

Practice JavaScript syntax and user defined functions.

Specification:

Write a series of JavaScript functions (7) to perform varying tasks using both **function definition** and **function expression notations**. To begin, download the following .js file:

Shared file: assignment1.js

This file includes a block of code (labeled "TEST DATA") that will invoke your functions with several different values and display their output to the console so that you can verify that they produce the correct output. You will be writing your functions ABOVE this block of code.

Once you have downloaded the file, open assignment1.js in Firefox Scratchpad and create comment line(s) for each of the Functions in assignment 1 using block comments. Here, you will indicate the name and purpose of each Function. e.g.

```
/*****************************************
*   someFunction(param1, param2)
*
*  Purpose:  description of what the function does
*****************************************/
```

Once this is complete, go through each function and code them per the specification below. If you try to run the code before all the functions are created, however, you will get an error in Scratchpad along the lines of "someFunction is not defined". To fix this error, simply define all your functions first (according to the specification) and then code them in any order you choose. Once again, be sure to write your functions ABOVE the "TEST DATA" Block of code, so that your functions are correctly invoked. This code does not tell you whether your functions are right or wrong, it simply invokes them several times so that you can evaluate them for correctness. You are not permitted to modify the "TEST DATA" block of code.

Function 1: grader

Write a function called "grader" using **function expression notation** with the following parameters & return value:

Parameters:

1. **mark**: a positive, whole number representing a percentage (%) mark that was received on an assignment, ie (98, 73, 61, etc).

Return value:

- The calculated letter grade (string) based on the provided percentage grade (**mark**). This will be either A (80% – 100%), B (70%-79%), C (60%-69%), D (50% - 59%) or F (0%-49%). For example, if the value of the **mark** parameter is **71**, the function would return the string "**B**" (since 71 is between 70 & 79). Similarly, if the value of the **mark** parameter is **35**, the function would return the string "**F**" (since 35 is between 0 and 49).

Function 2: showMultiples

Write a function called "showMultiples" using **function declaration notation** with the following parameters & return value:

Parameters:

1. **num**: Any whole number (ie, 4, 57, 99, etc).
2. **numMultiples**: Any whole number (ie, 12, 46, 73, etc).

Return value:

- A string that lists the first **numMultiples** multiples of **num**, separated by newline (**\n**) characters (to make the output more readable): For example: if **num** is **5** and **numMultiples** is **4**, the function would output the following string:

  "5 x 1 = 5\n
  5 x 2 = 10\n
  5 x 3 = 15\n
  5 x 4 = 20\n"

Function 3: largerNum

Write a function called "largerNum" using **function expression notation** with the following parameters & return value:

Parameters:

1. **num1**: Any whole number (ie: 2,567,3947, 4, etc.) or string representing a whole number (ie: "25", "65", "95", etc).
2. **num2**: Any whole number (ie: 2,567,3947, 4, etc.) or string representing a whole number (ie: "25", "65", "95", etc).

Return value:

- The larger of the two numbers (number), or NaN if either num1 or num2 cannot be converted to a number (ie: "a","abc", etc). For example, if **num1** is **"56"** and **num2** is **33**, the function will return **56**

(because 56 is larger than 33. Similarly, if **num1** is **95** and **num2** is **"a"**, the function will return **NaN** (because "a" cannot be converted to a number).

Function 4: tempConvert

Write a function called "tempConvert" using **function declaration notation** with the following parameters & return value:

Parameters:

1. **temperature**: The temperature to be converted (number), ie 78, 26, 37, etc.
2. **convert**: The type of conversion (string) to be done. This will always be either "CF" (to convert Celsius to Fahrenheit) or "FC" (to convert to Fahrenheit to Celsius)

Return value:

- The converted temperature (number) based on the provided **temperature** (ie, 78, 26, 37, etc) and the provided "**convert**" type (ie, "**CF**" for Celsius-to-Farenheit or "**FC**" for Fahrenheit-to-Celsius). For example, if the **temperature** is **22** and the **convert** is "**CF**", the function will return **71.6** (because we're converting 22 C to F, which is 71.6 F), Similarly, if the **temperature** is is **76** and the **convert** is "**FC**", the function will return **24.4** (because we're converting 76 F to C, which is 24.4 C).

  Note: visit **www.manuelsweb.com/temp.htm** for temperature conversion formula.

Function 5 evenNumbers

Write a function called "evenNumbers" using **function expression notation** with the following parameters & return value:

Parameters:

1. **minNumber**: Any whole number representing the minimum number to count from, ie 4,3,9, etc.
2. **maxNumber**: The maximum number to count to, ie 15, 19, 33, etc

Return value:

A string containing all even numbers (inclusive) starting from the provided **minNumber** (ie, 4, 2, 9, etc) to the provided **maxNumber** (ie, 15, 33, 19, etc.) separated by a comma. For example, if the provided **minNumber** is **8** and the **maxNumber** is **21**, your function will return the string "**8, 10, 12, 14, 16, 18, 20**". You may assume that the function will always be called with a **minNumber** value that is less than a **maxNumber** value and that the values will be whole numbers.

Function 6 passingAverage

Write a function called "passingAverage" using **function expression notation** with the following parameters & return value:

Parameters:

- This function takes an **unknown** number of parameters representing a list of grades (positive, whole numbers). For example, it may be called with 3 parameters (ie: passingAverage(75,42,98)), or 5 parameters (ie: passingAverage(34, 93, 77, 89, 49).

Return value:

true if the **average** of all provided grades is **greater than 49** and **false** if the **average** of all provided grades is **less than or equal to 49**. For example, if the provided parameters are **59**, **42**, **94** and **72**, the function will return **true** (since the average of 59,42,94 and 72 is greater than 49). Similarly, if the provided parameters are **43**, **53** and **39** the function would return **false** (since the average of 43, 53 and 39 is less than or equal to 49).

Function 7 counter

Write a function called "counter" using **function declaration notation** with the following parameters & return value:

Parameters:

- This function has no parameters.

Return value:

A new **function** that returns a single number – one (1). However, every time this function is invoked (called) in the future, this function will return a number that is one greater than the previous time it was invoked (called). For example:

```
var count = counter(); // call the "counter" function and assign the return value (function) to count
count(); // returns 1;
count(); // returns 2;
count(); // returns 3;
```

NOTE: You are not permitted to use any global variables in your solution

Assignment Submission:

- Add the following declaration at the top of your code:

```
/*********************************************************************************
* BTI225 – Assignment 01
* I declare that this assignment is my own work in accordance with Seneca  Academic Policy.  No part of this
* assignment has been copied manually or electronically from any other source (including web sites) or
* distributed to other students.
*
* Name: _____ Student ID: _____ Date: _____
*
*********************************************************************************/
```

- Submit your **assignment1.js** file to My.Seneca under **Assignments** -> **Assignment** 1