Assignment 1: Javascript, Assemble!

Starter Code

See assignment dropbox.

- Put the solution to each problem in the appropriate function in **a1.js**.
- If you want to test your function, write your test code in the appropriate section of **main.js**. You are **not** required to submit main.js.

Group Size

This is an individual assessment.

Due Date

See course webpage.

Late Submissions

See course webpage > Course Info

Submission Checklist

❑ Add the following declaration at the top of your `a1.js` file. Fill in the blanks with your name, student id, and date.

```
/********************************************************************************
*  WEB222 – Assignment 1
*  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: ________________
*
********************************************************************************/
```

❑ In the course dropbox, submit `a1.js.`

Academic Integrity

1. Please review the college's Academic Integrity policy, found here: https://www.senecacollege.ca/about/policies/academic-integrity-policy.html

2. Learners are reminded that using full or partial solutions found on the Internet is not permitted

    a. Distributing "help" or "source code" to other learners is not okay
    b. Receiving "help" or "source code" from others is not okay

Problems (5 questions x 2% each)

For the following questions, you are NOT permitted to use any of the ES6 higher order array functions, such as: find(), filter(), map(), reduce(), closest(), foreach()

Question 1

Create a function called **`avengersAssemble(`*`names`*`)`**, where ***names*** represents an array of string values. You may assume that the ***names*** array will **always** contain a **minimum** of 3 names.

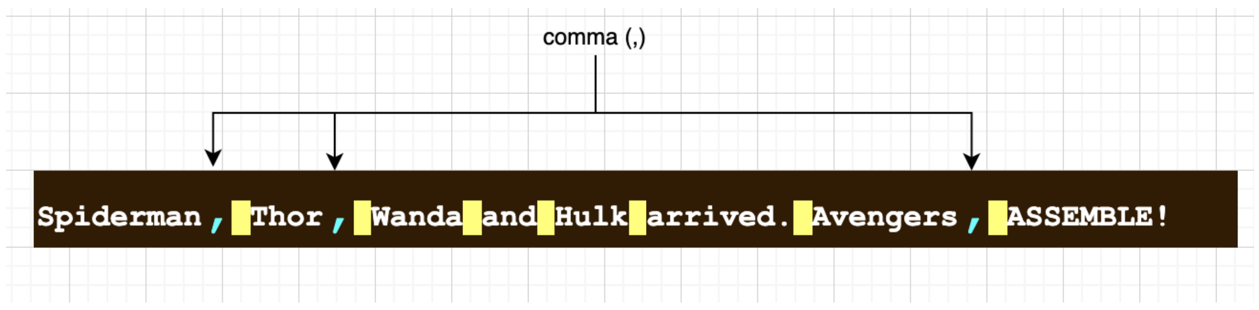
The function should behave as follows:

Function Call	`avengersAssemble(["Spiderman", "Thor", "Hulk"])`
Output	`Spiderman, Thor and Hulk arrived. Avengers, ASSEMBLE!`

Function Call	`avengersAssemble(["Wanda", "Nebula", "Shuri", "Wasp", "Captain Marvel"])`
Output	`Wanda, Nebula, Shuri, Wasp and Captain Marvel arrived. Avengers, ASSEMBLE!`

Note the positions of the commas and white spacing in the final output.
In the diagram below,

- , represents a single comma
- ▮ represents a single white space.


comma (,)
Spiderman, Thor, Wanda and Hulk arrived. Avengers, ASSEMBLE!

Question 2

Create a function called `toCamelCase(names)`, where ***names*** represents an array of string values. This function should convert each string in the array to camel case and then **return** an array containing the converted strings.

Example:

Function Call	`toCamelCase(["thor", "Iron Man", "BLACKwIdow"])`
Returns	`["thor", "ironMan", "blackwidow"]`

In camel case:

- The first letter of the first word in the string is lowercase. Remaining letters in the word are lowercase.
- The first letter of any remaining words are uppercase
- All spaces between words are removed

Example of camel case:

Example String	**String in Camel Case**
`thor`	`thor`
`Thor`	`thor`
`Iron Man`	`ironMan`
`IRONMAN`	`ironman`
`Black Widow`	`blackWidow`
`BlackWidoW`	`blackWidow`
`thE QuiCk brown fox juMps over a laZY Dog`	`theQuickBrownFoxJumpsOverTheLazyDog`

For this question:

- you are **not** permitted to use regular expressions.
- you **may** use Javscript's built-in toUpperCase() and toLowerCase() functions.

Questions 3-5

Questions 3-5 require you to work with an Avengers object.

Use the following definition of an Avengers object:

- Each Avenger has a name (string), attack power (number), and health points (number).
- A sample Avengers object looks like this:

```
{
   name: "Thor",
   attack: 25,
   hp: 50
}.
```

----

3. Write a function called `getWinner(a1,a2)`. The function accepts two Avenger objects (a1, a2). The function should compare the attack power of each Avenger and **return** a string containing the name of the Avenger with the higher attack power. In the case of a tie, output the name of the Avenger who is alphabetically first.

For example, if “Spiderman” and “Black Widow” have the same attack power, output “Black Widow”.

You are NOT permitted to use arrays in your solution.

4. Write a function called `findAvenger(name, avengersList)` that accepts the name of an Avenger and an array of Avenger objects. The function must search the array for an Avenger object with the provided name. If found, return the Avenger object. If not found, return an empty object (an empty object is {})

Reminder: You are not permitted to use higher order array functions in your solution.

5. Create a function called `fingerSnaps(avengersList)`. This function accepts an array of Avengers objects. The function loops through the array and decreases the health points of each Avenger by 15 points. The minimum number of health points for an Avenger is zero (0). After modifying each Avenger’s health points, return an array containing the Avengers with their updated health points.

Reminder: You are not permitted to use higher order array functions in your solution.