WEB222 Assignment 2

Submission Deadline:
14th Feb, 2020, 11:59 pm

Assessment Weight: 5% of your final course Grade

Fill in and include the following content at the beginning of your file:

```
/**
 * WEB222 Assignment 02
 *
 * 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.
 *
 * Please update the following with your information:
 *
 *      Name: <YOUR_NAME>
 *      Student ID: <YOUR_STUDENT_ID>
 *      Date: <SUBMISSION_DATE>
 */
```

Included is an extra file in this assignment: `users.json`. This is a JSON file, which is a data format for representing Objects (NOTE: we'll talk more about JSON in later weeks, and you can read more about it at https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON) You can think of JSON files like mini databases for loading data into a program. We'll use it in this assignment to create a list of Users in an imaginary application.
This Users data is an Array where each item in the Array is a custom User Object. Users is really [user, user, user, ...], and each user Object has the following structure:

```
 {
   "id": 1,                          // A unique Number
   "name": {                         // An Object with user name info
     "first": "Paige",
     "last": "Bools"
   },
   "birthDate": "1995-02-04T07:34:45Z",   // A Date in String form
   "contact": {                      // An Object with contact info
     "phone": "8989068955",
     "email": "pbools0@webmd.com"
   },
```

```
  "address": {                    // An Object with address info
    "street": "476 Veith Parkway",
    "city": "Cuamba",
    "country": "Mozambique"
  },
  "accessCount": 776,              // A Number: access count to the app
  "isManager": false               // A Boolean: is this a manager?
}
```

Download the json file and add it to the same folder as your code. The following line of code will load and parse the file named `users.json`, setting the variable `users` to be an Array of user Objects. (NOTE: if you're interested in reading more about this, see: https://nodejs.org/api/modules.html#modules_require_id).

const users = require('./users.json');

Add this statement near the top of your code to set the variable users to be an array of Objects. As we discussed above, the `users` variable is now an Array where each item in the Array is a user Object: [user, user, user, ...].
To get you started, write a function that gets the first element in the users Array, and returns only the country String for this user.

```
function firstName() {
  // Get the first element from the users Array
  var firstUser = users[0];
  // TODO: fix this code to use dot notation to access the first name  portion only
  return firstUser.name;
}
```

If correctly added, this function should return the information for “Paige Bools”.

Task 1 - Write a function called **getAge(birthdate, currentdate)**
birthdate- The birthday of each of the users
currentdate- The date today.
The function will take these two parameters and return the age of the user.

Task 2 - Using the getAge() method, write a function called **getAvg()** that returns the average age of all the users in the array

The next problems use a custom Object named userUtils. You will be asked to add methods to this object in the problems below. Add the following line of code to define an object called userUtils**.**

const userUtils = {};

Task 3 - Add a new method to the “userUtils” Object called **getManagers**(). This method will return a new Array with all manager users. In other words, the new Array will include all user Objects where isManger is true.

Task 4 - Write a function called **getNameLength(firstname,lastname)** that takes as input the first name and last name of a user, concatenates the first name and last name (with a blank space in between) and returns the length of this string.

getNameLength(“Sharmin”,”Ahmed”) returns 13

Task 4 - Now using the getNameLength method**,** add a new method to the `userUtils` Object called **getLongestName().** This method will find the user in the users Array with the longest name and return his or her user Object.

Task 5 - Add a new method to the `userUtils` Object called **`searchbyName()`**. This method will allow us to search for users in the `users` Array by their name:
userUtils.searchbyName('Paige');
This will return an Array of all user Objects which have a firstname or lastname that matches 'Paige'.
userUtils.searchbyName('Pa', true);
This will return an Array of all user Objects which have a firstname or lastname that begins with the string 'Pa'. The second argument specifies that we want to do a fuzzy search (i.e., matches don't have to be exact). An exact search is one where the name given must match exactly with the firstname or lastname. A fuzzy search only has to begin with the name passed, and should work for both lower and UPPER case comparisons (e.g., it should not matter, and should match if the letters are the same regardless of case). Your function should return an Array with all user Objects that match the name criteria.
name - the name or name portion to search for
fuzzy - if true, do a fuzzy search; otherwise do an exact search

Task 6 - Add a new method to the ‘userUtils’ Object called “**mostCommonCountry(users)**” that returns the country with the most number of users. The method should take as input the users array and return the name of the country as well as the number of users in that country. To solve this problem, you may first create an array of all the user countries and the number of users in each of these countries. Now you can use this array to find the country with the maximum number of users.