Lab 5 Instructions, Selection, Group B

CS 133JS, Beginning Programming: JavaScript

Overview

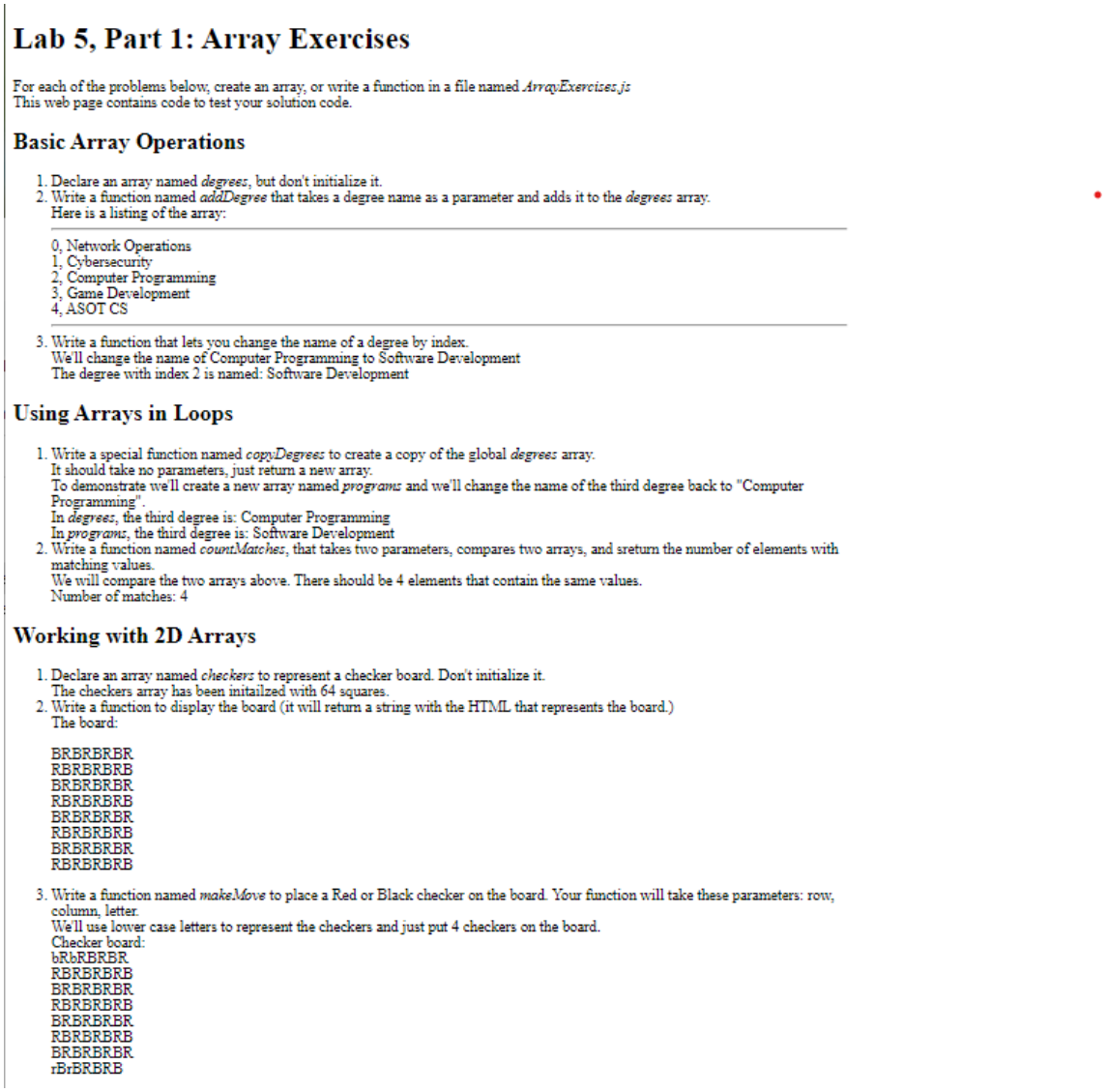
The objective of this lab is to give you practice using:

- Creating an array
- Adding values to an array
- Getting values from an array
- Calling array methods to do special operations on the array

Part 1: Array Exercises

A web page, *ArrayExercises.html*, has been written for you that contains code to call functions that you will write in a file named *ArrayExercises.js*. The instructions for writing your functions and the code to test your functions are *ArrayExercises.html*, but all the code you write will go in *ArrayExercises.js.*

Here is a screenshot of the finished ArrayExercises:



Lab 5, Part 1: Array Exercises

For each of the problems below, create an array, or write a function in a file named *ArrayExercises.js*
This web page contains code to test your solution code.

Basic Array Operations

1. Declare an array named *degrees*, but don't initialize it.
2. Write a function named *addDegree* that takes a degree name as a parameter and adds it to the *degrees* array.
   Here is a listing of the array:

   0, Network Operations
   1, Cybersecurity
   2, Computer Programming
   3, Game Development
   4, ASOT CS

   ---

3. Write a function that lets you change the name of a degree by index.
   We'll change the name of Computer Programming to Software Development
   The degree with index 2 is named: Software Development

Using Arrays in Loops

1. Write a special function named *copyDegrees* to create a copy of the global *degrees* array.
   It should take no parameters, just return a new array.
   To demonstrate we'll create a new array named *programs* and we'll change the name of the third degree back to "Computer Programming".
   In *degrees*, the third degree is: Computer Programming
   In *programs*, the third degree is: Software Development
2. Write a function named *countMatches*, that takes two parameters, compares two arrays, and sreturn the number of elements with matching values.
   We will compare the two arrays above. There should be 4 elements that contain the same values.
   Number of matches: 4

Working with 2D Arrays

1. Declare an array named *checkers* to represent a checker board. Don't initialize it.
   The checkers array has been initailzed with 64 squares.
2. Write a function to display the board (it will return a string with the HTML that represents the board.)
   The board:

   BRBRBRBR
   RBRBRBRB
   BRBRBRBR
   RBRBRBRB
   BRBRBRBR
   RBRBRBRB
   BRBRBRBR
   RBRBRBRB

3. Write a function named *makeMove* to place a Red or Black checker on the board. Your function will take these parameters: row, column, letter.
   We'll use lower case letters to represent the checkers and just put 4 checkers on the board.
   Checker board:
   bRbRBRBR
   RBRBRBRB
   BRBRBRBR
   RBRBRBRB
   BRBRBRBR
   RBRBRBRB
   BRBRBRBR
   rBrBRBRB

Part 2: Web Apps

You will create two web apps. The HTML page for each of these has already been written for you. You will just write the JavaScript file.

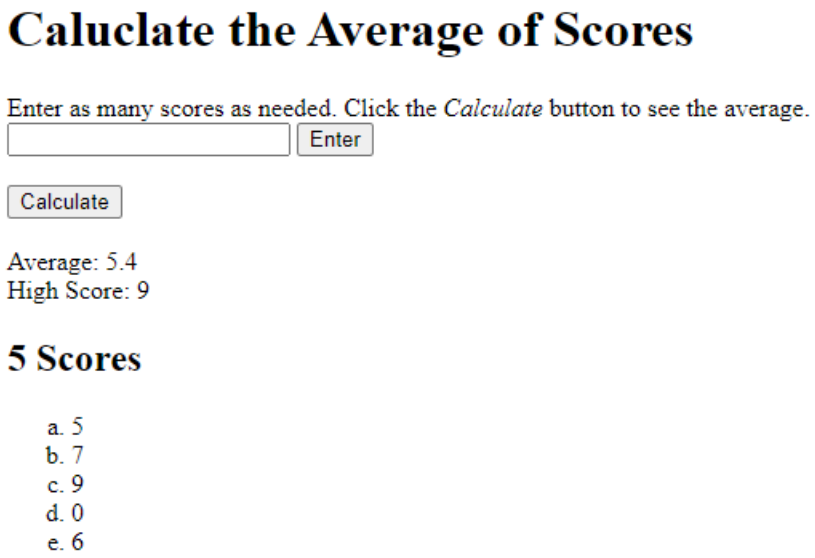
Web App I for Group A – Average Score Calculator

You have been provided with a web page for entering scores, calculating an average, and finding the highest score. You will need to create a file named *AverageOfScores.js* in which you will:

1. Declare an empty array named *scores*.
2. Define a function named *calcAverage* that has no parameters and that returns the average of the values in the scores array.
3. Define a function named *findHighest* that has no parameters and returns the highest value in the scores array.

Use loops to calculate the average and to find the highest score.

Here is a screenshot of the finished web app:



Lab 5 Instructions, Selection, Group B

CS 133JS, Beginning Programming: JavaScript

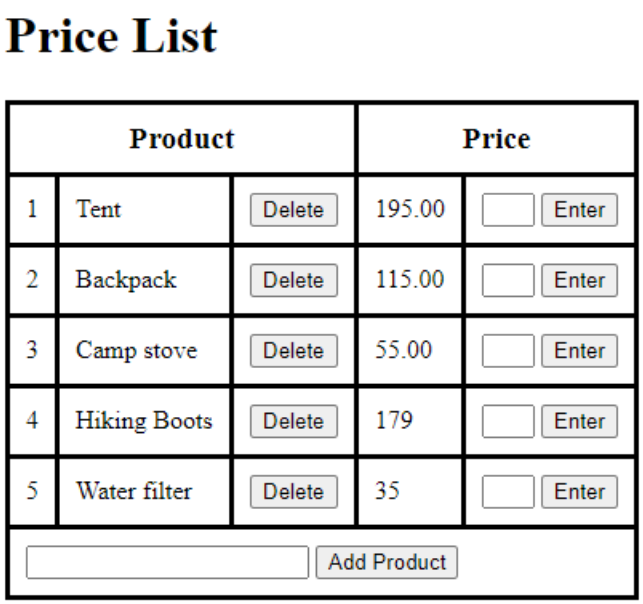
Web App II for Group A – Price List

This web app displays a list of products and prices. A user can add items to the list and enter prices.

Implementation:

1. Declare two one-dimensional arrays:
    a. *products*
    b. *prices*
2. Write three functions:
    a. *addProduct*
        - Has one parameter: a product name.
        - Returns true if the product was found.
    b. *removeProduct*
        - Has one parameter: a product name.
        - Returns true if the product name was found.
        - Use the *indexOf* method.
    c. *changePrice*
        - Has two parameters: the array index for the product, and the price.
        - Returns true if the index is valid.
        - Use the *split* method.

This is a screenshot of a completed web app:



Submitting your lab work on Moodle

Beta Version

Post the following in the *Lab Beta forum*:

1. The web pages you created for part 2.
   (Zip the files for you web pages and attach them to the post.)
2. A code review of your lab partner's web page for part 2.
   (Review the part 2 web apps for one of your lab partners using the Code Review Form provided.)

Code Review

1. Submit a copy of the code review above to the *Lab Code Review assignment*.

Production Version

You may revise your beta version before submitting the production version. On the code review form you received from your lab partner, complete the "Production" column to show what you did or did not revise.

Upload the following to the *Lab Production Version* assignment:

1. A zip file containing the two files (.html and .js) for part 1.
2. A zip file containing the four files for part 2.
3. The code review from your lab partner with the "Prod" column filled in by you.