Assignment 1: Login with Encryption & Testing (8% of final grade)

Format: Submit zipped folder containing all files through Blackboard. BE SURE TO PUT YOUR NAME & SECTION IN THE FILENAME: e.g. *Assign1-YourName-A.zip*

BE SURE TO INCLUDE THE VALID USERNAME & PASSWORD IN YOUR SUBMISSION TEXT

Purpose: To implement JavaScript encryption with CryptoJS, and unit testing of your functions with the Jasmine testing framework.

REQUIREMENTS: Modifying your HTML markup from Lab 2 (login)…

1. In addition to the onsubmit function, you will create two custom functions:
    a) **md5Encrypt** (This function has been created for you. It will encrypt a string with MD5 and return a string.)
    b) **checkLogin** (validate user input)
2. Be sure to add a comment above your checkLogin function to describe what the function does, what arguments it expects, and what value it returns. See the md5Encrypt function for an example.
3. The empty output div should now be used for a successful login message (“Welcome back!“) as well as error messages, and NOT for outputting the user/pass.
4. Incorporate thorough Jasmine testing to test the two **md5Encrypt** and **checkLogin** custom functions (*see specifications below*).
5. **Your functions in your Jasmine src folder must be the functions used by your main logic JS file. Creating a testing function that is different from the function used in your code is considered hacking the testing and you will receive a zero for the assignment (you would be immediately fired from any job where you tried to cheat the testing).**

LOG-IN FUNCTIONAL SPECIFICATIONS

The **md5Encrypt** function should return a string of the hashed value (as 32 hexadecimal characters). *Note: This function has been completed for you.*

The **checkLogin** function should use the **md5Encrypt** function, and return the Boolean true if the username and the password match a known username and matching password.

The **checkLogin** function should return 'Invalid Username or Password.' if the username input does not match a known username; or the password input does not match a known password; or a valid username is input with an invalid password, or an invalid username is input with a valid password.

The **checkLogin** function should return 'No username entered.' if the username is an empty string.

The **checkLogin** function should return 'No password entered.' if the password is an empty string.

If you need some hints, scroll down past the rubric…

Marking Rubric

Criteria	Proficient	Competent	Novice
Web Page Success Message	0.5	0	0
	Page loads without errors, and provides specified message for successful login.	Successful login provides correct message, but there are console errors.	Successful login is not possible, or message does not match specification.
Web Page Error Messages	1	0.5	0
	Web page provides specified messages for invalid input and empty string (username & pass).	Web page provides some, but not all, of the specified messages for invalid input and empty string.	Web page does not provide specified error messages.
checkLogin Function: Commenting	0.5	0	0
	The function is preceded by a comment block that describes what the function does, what arguments it expects, and what value it returns.	The function is preceded by an incomplete comment block.	There is no comment that describes what the function is for.
checkLogin Function: DOM Access	1		0
	This function does not directly access the DOM.		This function accesses the DOM.
checkLogin Function: Return Value for Valid	0.5		0
	The function returns the Boolean true if the username and the password match a known username and matching password.		The function returns a value other than the Boolean true for a matching username and password.
checkLogin Function: Return Values for Invalid	1	0.5	0
	The function returns the specified error messages.	The function returns some of the specified error messages.	The function does not return the specified error messages.

Criteria	Proficient	Competent	Novice
Jasmine: Execution	1		0
	The Jasmine Spec Runner runs without errors.		The Jasmine Spec Runner shows errors.
Jasmine: Test md5Encrypt	1	0.5	0
	Jasmine tests md5Encrypt to return a 32 character HEXIDECIMAL string.	Jasmine tests md5Encrypt to return a 32 character string.	Jasmine does not test md5Encrypt to return a 32 character string.
Jasmine: Test checkLogin Return Value	0.5		0
	Test to pass checks for Boolean true.		Test to pass checks for a different value.
Jasmine: Test checkLogin Error Messages	1	0.5	0
	Test to fail checks for all specified error messages and instances.	Test to fail checks for most specified error messages.	There are no test to fail specs, or it is not checking for specified error messages.
CRITICAL FAIL	**0**	**0**	**-8**
			Function in Jasmine is not identical to function used in app logic.

To Get Started…

- I strongly urge you to complete the assignment in this order:
  1. Create your Jasmine tests (specs) for both custom functions.

2. Create your **checkLogin** function, starting with the comments.
3. Make sure **md5Encrypt** and **checkLogin** pass all of your tests.
4. Create your js file for your HTML file, set up your **onload** and **onsubmit** functions. In your HTML file, include your function files from the Jasmine *src* folder.
5. Complete the rest of the assignment.

- In your *Assign2-YourName* folder copy your lab 2 login HTML file (but not your JS file, as you will need something different) and unpack your Jasmine-standalone zip file. On the JavaScript page inside the onload function, create your onsubmit function.
- Your **onsubmit** function will get the values from the form, then pass them as *parameters* to your **checkLogin** function. Your checkLogin function will call **md5Encrypt** when it compares the user input to your hard-coded encrypted password.
- Remember, if you are using the <form> and not just grabbing the <input/> by id, you need to return false at the end of your onsubmit function so that you can output to the same page. Otherwise, the page will refresh on submit and your form values will be gone.

Your Functions…

- The role of a function, generally speaking, is to perform a task apart from the main flow of logic, then return to the main flow with the result. **Neither of the custom functions should access the DOM**. That should be handled by the main flow. So, the main flow goes to the DOM and gets values; your functions process the values and return values to the main flow; then the main flow outputs to the DOM.
- In **checkLogin**, normally we would go to a database and check for a matching username/password there. For this assignment, check against a hard coded user/pass.

- Leave your output logic until after the functions have run, then output to the html page.

Testing…

- Remember that you are not putting your whole js file into the *src* folder, you are pasting the two functions into their own new js file for testing with Jasmine.
- Don't forget to include CryptoJS in your HTML file and in the specRunner.html file just above the <!-- include source files here... →.
- For testing your encrypt function, use Jasmine's toMatch() matcher with a regular expression (a 32 character hexadecimal string).

Your final submission structure should look like this…

assign-1-MyName.zip

assign1.html
assign1.js
jasmine
lib
jasmine-X.x.x
md5.js
src
checkLogin.js *(this is the new file that you created)*
md5Encrypt.js *(this is the file that I gave you)*
spec
login_spec.js
SpecRunner.html