EECS 1012: LAB #9 – Learning Kit; Code Breaker (March 11– 17, 2019)

1. Read the lab instructions in this document and take the pre-lab quiz for Lab #9.

NOTE: Try to complete the tasks given in this write up before coming to your lab session, the lab must be completed and verified by the TA before the end of the lab session.

2. GOALS & OUTCOMES FOR THIS LAB

- To practice more computational thinking algorithms
- To practice JQuery
- To Practice working with DOM

3. LAB 9 – Two tasks

1) TASK 1: Complete your learning kit with 30 flowcharts and JS codes, and also enhance it by using JQuery and DOM.
2) TASK 2: Client side of Code Breaker

4. SUBMISSIONS

1) [Manual verification by a TA]
As with previous labs, when you have completed all tasks, ask the TA to come and verify your code and output. You must sign the verification sheet to get marks. If you attended the entire lab and still cannot complete the lab before the end of the session, you will receive 50% credit for the lab.

2) Moodle submission
Create a **folder** named "**Lab9**" and copy **all** of your HTML and JS files.

Task 1) In the learning kit that you developed in Lab3, you considered the flowcharts and JavaScript codes for 10 problems. In this lab, you should increase the number of problems to 30 with the following requirements:

A) at least 5 problems should have a *loop* without nesting.
B) at least 3 problems should have a *nested loop*.
C) at least 2 problems should have a nested loop with depth of minimum two.
D) at least 3 problems should be related to *arrays*.
E) at least 5 problems should call *functions*.

Examples of algorithms requiring one loop without nesting:

1) Sum of numbers 10 to 30
2) Factorial
3) Fibonacci
4) a*b, without using multiplication (assume *a* and/or *b* are whole numbers)
5) a modulo b, without using modulo operation (use subtraction instead)
6) converting a number from base 10 to base 2.

Examples of algorithms requiring loops with one level of nesting:

1) draw triangle (Exercise 15 in Lab5)
2) output the multiplication table of size *n*
3) output a^b, without using power or multiplication
4) factorial, without using multiplication

Examples of algorithms requiring loops with two levels of nesting:

1) multiply two matrices
2) sum of numbers in a 3D array

Examples of algorithms using arrays:

1) examples on Slides 5-39 to 5-44
2) Simple Lottery: define an array of 3 items, store a random number between 0 to 9 to each item; ask a user to guess 3 numbers between 0 to 9 (the order is important); if all 3 guesses are good, bingo!

Examples of algorithms using functions:

1) An example on Slide 5-49
2) Output factorials of 1 to 10
3) Sum of all prime numbers less than 100
4) Output all numbers between 1 and 1000 that all palindrome

Bonus Option 1: You obtain 5% bonus points on this lab, if you increase the number of problems to 40. It's at discretion of TAs to judge if your 40 problems are distinct or at least diverse enough or not.

Bonus option 2: You obtain another 5% bonus points if you create the buttons of `myLearningKit.html` file (from Lab3) by `JavaScript` using JQuery and DOM.

In particular, in `myLearningKit.html`, delete all buttons from the `nav` tag:

```
    <nav>
      <!-- In Ex2, add an attribute onclick to button-->
      <button>Problem01</button>
      <!-- In Ex4, add an attribute onclick to button-->
      <button>Problem02</button>
      <button>Problem03</button>
      <button>Problem04</button>
      <button>Problem05</button>
      <button>Problem06</button>
      <button>Problem07</button>
      <button>Problem08</button>
      <button>Problem09</button>
      <button>Problem10</button>
    </nav>
  </header>
```

delete these

And add the following code fragment to your JS file:

```
for (var i = 1; i<=30; i++){

    var newBtn = document.createElement("button");
    $(newBtn).attr("id", "btn" + i);
    $(newBtn).html("Problem " + i);

    $("nav").append(newBtn);
}
```

Note: we assign an id to each button.

We leave it for you to figure out how and where exactly this code can be added. Also, you need to figure out how to add handlers for "on click" events to these buttons.

To enhance your code a bit more, if you like to have more meaningful captions for each button, you may want to create an array like the example below.

```
var caption=["Odd or Even", " Sum of Numbers", "Triangle Area",
             "Sum of 10 to 30", "Decimal to Binary", "Quadratic Equation",

for (var i = 1; i<=30; i++){

    var newBtn = document.createElement("button");
    $(newBtn).attr("id", "btn" + i);
    $(newBtn).html(caption[i-1]);

    $("nav").append(newBtn);
}
```

Show your complete lab to your TA.

Task 2) Use html, CSS, JS, (and your creativity)) to design the client side of the code-breaker game—of your taste. If it's your own design, you gain 10% bonus points. We provide a sample starting code by Wednesday for those who want to use our design. You would still need to complete that code. In particular,

1) Open `code_breakerV0.html` and save it as `code_breaker.html`. You should read the comments in your `code_breaker.html`, and make changes such that your html file becomes similar to `codebreakerV1.html` eventually.
2) Then, open `code_breakerV0.js` and save it as `code_breaker.js`. In the `createGameBoard` function of `code_breaker.js`, you should read the comments and write the code for all 15 lines specified by "`//...`". These lines are related to html tags that you deleted from your html file, and you are creating them in your js file.

If you make the above changes properly, your code_breaker should work fine. Notice that the url provided in the js file expires every 8 hours, so you need to update it the latest url provided in moodle.

Show your code-breaker code to your TA.

Note that this project is a great source of learning. Hence, read all comments carefully and make sure you have a clear understanding of the code.

Next week, we will complete this project by adding the server component to it.