EECS 1012: LAB 10 – Learning Kit; Code Breaker (March 18– 24, 2019)

1. Read the lab instructions in this document and take the pre-lab quiz for Lab #10.

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 server-side code: node.js, express and JSON

3. LAB 10 – Two tasks

1) TASK 1: complete your learning kit with 40 flowcharts and JS codes.
2) TASK 2: server-side of the 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 "**Lab10**" and copy **all** your files.

Task 1) This is another chance to make sure you have a great selection of computational thinking flowcharts and corresponding JS codes. This time, we provide you with a sample solution of Task 1 of Lab9. So, the HTML, CSS, and JS files are already given to you. Moreover, the solution is Dynamic HTML. In particular, the JS file uses DOM and jQuery to manipulate the HTML file at run time. What do you need to do in this task? The following 4 steps:

I. **Read the JS file line by line, perhaps with help of some mentors. That's a great source to extend your JS, jQuery and DOM manipulation even further.**

II. The sample solution has only images for its buttons 19 and 20, i.e. `Prime` and `Sum` problems. You are required to add 18 more pair of images for flowcharts and JS codes related to buttons 1 to 18. Save these images in the `images` folder with name `fc_<x>.jpg` and `js_<x>.jpg` where you replace the `<x>` with the problem number. We already provided you with `fc_19.jpg`, `js_19.jpg`, `fc_20.jpg`, and `js_20.jpg`. Also, store the JS code of each problem as a function near end of `learningKit.js` file, where you see the functions for current problems 19 and 20. Note that you do not have to stick to current problems described for the buttons. Instead, you can replace those problems with problems of your choice, if you wish. In that case, you would need to update the `buttonCaption` array and the `descriptionOfProblem` array accordingly.

III. Modify the JS code to increase the number of buttons to 40. That means you would need to modify the `buttonCaption` array and the `descriptionOfProblem` array as well as the for-loop iteration.

IV. You would need to add to the `images` folder the images for the flowchart and JS codes of the 20 buttons that you just added, with the naming format specified in Step II above and also copy each JS code as a function towards end of `learningKit.js`.

Note: This task will take an enormous amount of time if you have not prepared them before going to the lab. So, make sure you do have images for most—preferably all—of these 40 problems (both for their flowcharts and for corresponding JS codes).

Also, note that these 40 problems do not need to be brand new problems. Instead feel free to recycle many problems that you saw in the course.

Keep in mind to meet 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*.

Refer to Lab9 for examples of algorithms.

Show your complete task to your TA. Your TA may randomly pick one or two of your problems and ask you to explain it or modify it a little bit.

Task 2) In Task 2 of Lab 9, you developed the gameboard of the **client-side** of the codebreaker game. In this task, you complete the **server-side**. What do you need to do in this task? The following 4 steps:

i) first, review JSON data structure
ii) then, go over details of the requests that are sent from the client-side: one is in `initGameBoard()` and the other is in `processAttempt()` function.
iii) Also, make sure to understand the details of how a response received from the server is processed in the client-side.
iv) Then, copy `code_breaker_serverV0.js` and follow the comments to complete the code. In particular, there are ten `/*T0DO…*/` comments that you should replace with your code.
v) Finally, follow the instructions in readme.txt to run the server and play the game!

Have fun!

Show your code-breaker code to your TA.