

EECS 1012: LAB 04 –HTML+JS+ComputationalThinking (June 3-10, 2021)

A. REMINDERS

- The pre-lab mini quiz is considered part of this lab.
- You are required to complete the pre-lab mini quiz 4 posted on eClass no later than the beginning of your lab time, i.e., 10am this Friday if you are in the morning sessions (lab03, 04), and 1:30pm if you are in the afternoon sessions (lab01, 02, 05)
- You are welcome to attend the lab session, if you stuck on any of the steps below. TAs and instructor will be available via Zoom to help you. If you need help, you are expected to come to your own lab session. Please check the updated lab times and Zoom links on eClass (may change slightly for different labs).
- Please be patient while you are waiting to be admitted by the zoom host. This is one-to-one session and the TAs admit on first come first served basis.
- Instructor will be around in different lab sessions to help. If you need to talk to the instructor, let the TA know and they would notify the instructor.
- You can submit your lab work anytime before the specified deadline.

B. IMPORTANT PRE-LAB WORKS YOU NEED TO DO BEFORE GOING TO THE LAB

- 1) Download this lab files and read them carefully to the end.
- 2) You should have a good understanding of
 - Events (such as `onclick`) and event handlers
 - `document.getElementById().innerHTML`
 - `js` functions such as `parseInt()`, `parseFloat()`, `toFixed()`
 - the `if` statement and `switch case` statement in JavaScript
 - memory space (aka variable), `js` operators such as `“+”` and the concept of overloading
 - flowchart symbols as described in the lecture notes
- 3) Practice drawing flowchart symbols in `draw.io` or MS PowerPoint or Word. If you plan to draw your flowcharts on paper, please make sure you have pencils, erasers, and perhaps a ruler.

C. GOALS/OUTCOMES FOR LAB

- To practice **computational thinking** by first drawing flowcharts for basic computation problems, followed by implementation in JavaScript.

D. TASKS

- 1) Your first and major task in this lab is to draw 7 flowcharts.
 - Note you should NOT open VS-Code or browsers before finishing Task 1. You can draw your flowcharts on paper or draw them in `draw.io` or in MS PowerPoint or Word.
 - You are encouraged to discuss the flowcharts with peers, TAs and instructor. **But you cannot copy other's solution.**
- 2) Then, you are provided with **ct.html** document and supporting files such as **ct.css** and **ct.js**. Your task is to translate your flowcharts to `js` code.
 - You will generate at several **html** and **js** files in this process.
- 3) See next pages for details on how to modify your **html** and **js** files.

E. SUBMISSIONS

eClass submission. More information can be found at the end of this document.

F. COMPUTATIONAL THINKING

Part 1: For this flowchart part you are encouraged to discuss with other people (peers, TAs, instructor), even to share your partial solutions. But you should not copy other people's solution.

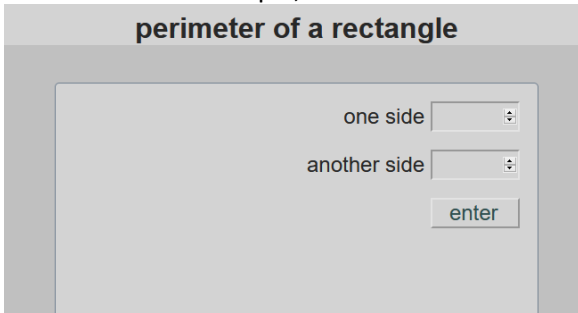
Using a computer program (or on paper), draw the following flowcharts and write your name on each. By end of this lab, you should take a screenshot (or a picture) from each flowchart and you should submit them to eClass as **img_{01,02,03,04,05,06A,06B}.jpg** files, where **img_x** is the flowchart for exercise **x** below. Make sure the size of each image is less than 500KB, e.g. by reducing the resolution of your camera.

IMPORTANT: You are required to use the symbols introduced in the lecture which are inspired from this book ("Computer Science: a first course" by Forsythe, Keenan, Organick, Stenberg)

IMPORTANT: You are required to provide preconditions and postconditions for each solution you provide.

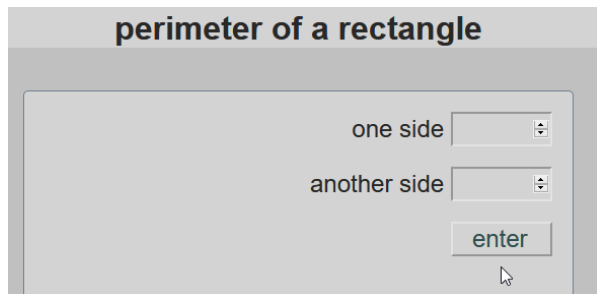
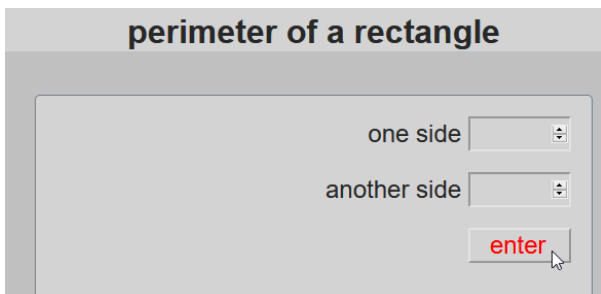
- Ex 0) as a starter, read the provided flowchart for a computer program to receive two numbers and output the larger one. Assume the two number are not same.
- Ex1) draw a flowchart for a computer program to receive two numbers as sides of a rectangle and output the rectangle's perimeter, in the form of "Perimeter: x" where x is the perimeter. Save your work as *img_01.jpg*
- Ex2) draw a flowchart for a computer program to receive three numbers and store them in memory spaces called *a*, *b*, and *c* as three semi-axes of an ellipsoid, and calculate and output the volume of this ellipsoid, in the form of "volume: x" where x is the volume. Figure out the formular from the internet. Save your work as *img_02.jpg*
- Ex 3) draw a flowchart for a computer program to receive three floating point numbers and store them in memory spaces called *a*, *b*, and *c*, and return the largest number of the three, in the form of "largest: x" where x is the largest value. Assume the three number are all different. Hint: you can compare pair of the values and there are several cases to consider, but a simpler way to compare is, as you see in the SMQ1, check if *a* is larger than *b* AND also larger than *c*. If yes you got the answer. If not (else), check if *b* is larger than *a* AND also larger than *c*, if yes you got answer, if not, you also got the answer. *If else if else* should be enough. Save your work as *img_03.jpg*
- Ex 4) draw a flowchart to receive three numerical coefficients of a quadratic equation and determines if it has two distinct real roots, one root, or no roots in real numbers, output "it has 2 distinct roots", "its roots are identical", "it has no roots in real numbers" respectively. This page might be a good reference: <https://www.math10.com/en/algebra/quadratic-equation.html> Save your work as *img_04.jpg*
- Ex 5) assume there is a webpage containing an HTML input of type text and a button. When the button is clicked, a function named *divisibleBy6* is called. Draw a flowchart for the function, that receive a number, and first check whether the input is divisible by 6 or not, outputting "divisible by 6" or "not divisible by 6", respectively. **Then**, further check if the input is zero, and if it is zero, also outputs "program ended". Hint: you need two *if* clauses, one after another. First check if the input is divisible by 6, and output accordingly, and then check if the input is 0 or not. Save your work as *img_05.jpg*.
IMPORTANT RESTRICTION: you are not allowed to use the modulus operator (%) over 6 to verify the remainder whether the number is divided by 6. You can do any other math trick you wish.
- Ex 6A) draw a flowchart to receive a number from 0~100, and map it to a letter grade based on York standard. Use *if .. else if .. else if ...* You may need to look at this reference: <http://calendars.registrar.yorku.ca/2012-2013/academic/grades/index.htm> Assume no letter grade E, i.e., if the grade is below 50, it's mapped to F. Save your work as *img_06A.jpg*.
- Ex 6B) draw a flowchart to receive a number from 0~100, and map it to a letter grade based on York standard. Use the "many case" style shown in class (which corresponds to *switch cases* struct in JS programming language). Again assume no letter grade E. Save your work as *img_06B.jpg*.

Part 2: the main task of part2 is to implement the flowcharts given in part1 using JS. You are given **ct.html**, **ct.css**, **ct.js** files. Reading these files carefully in order to enhance your learning before doing the following exercises. For example, in JS we use the `.value` property (not `.innerHTML`) gets the input text in textbox.



Exercise 0. Copy **ct.html** to **ct_Ex0.html**. Copy **ct.js** to a new file named **ct_Ex0.js**.

- Add your name to the list of authors of this page.
- Link the JS file to the html file.
- Improve the webpage in such a way that when mouse over the button, the text on the button becomes red. When the mouse left the button, the button text resumes the original color. (Hint: consider the *mouseover* and *mouseout* events).



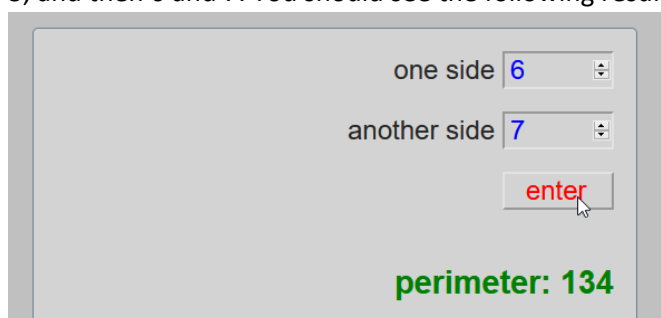
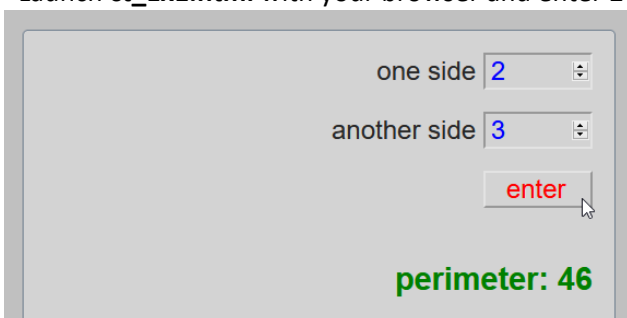
Exercise 1. Copy **ct_Ex0.html** to a new file named **ct_Ex1.html**. Copy **ct_Ex0.js** to a new file named **ct_Ex1.js**.

Launch **ct_Ex1.html** with your browser and enter two numbers and click on the “enter” button, nothing happens. In this exercise, you would modify the code such that it runs properly:

Make following changes to **ct_Ex1.html**:

- Connect it to **ct_Ex1.js** by adding a link in the head element.
- Add an event to the button such that when it’s clicked you handle that event by the `perimeter()` function in your **js** file.

Launch **ct_Ex1.html** with your browser and enter 2 and 3, and then 6 and 7. You should see the following result:



The expect perimeter should be 10 and 26 but we got some other values. The reason is that, as demonstrated in class, the variables `w` and `h` in your `js` function – retrieved from the `.value` of the DOM that corresponds to textbox in HTML – are from data type “string”. The data type variables in `js` are determined by the data type of the expression that is on the right side of the assignment. In this case, because the data type of property value of the `html` object that the `getElementById` returns is “string”, therefore the data type of `w` (and `h`) is “string” too.

Also, the “+” operator is overloaded in `js` (and in many other programming languages). That means “+” has more than one meaning in `js`. As far as our concern is, one meaning is to concatenate two strings (you will use this in output part) and another meaning is to add two numbers. `w` and `h` are currently strings “2” and “3”, therefore “+” concatenates them, and the result is string “23”, which is then converted to a number 23 to be multiplied by 2 (as

* makes no sense when applied to strings, but can work if that string is interpreted as a number), giving 46. For the same reason, another input 6 and 7 and you will get 134.

We need to convert the string to numbers. As demonstrated in class, `parseInt()` is a JS function that receives a string as its argument and returns its equivalent integer number. In other words, it can receive “2” and return 2. It can also receive “3” and return 3.

In summary, “2”+“3” results in “23”. But, 2+3 results in 5. So, open your `ct_Ex1.js` file and modify the code such that sum of the two inputs are calculated, not their concatenations. You should get the following results.

The image shows two side-by-side screenshots of a web form. Each form has two input fields: 'one side' and 'another side', each with a small up/down arrow. Below the inputs is a red 'enter' button. At the bottom of each form, the result is displayed in green text. In the left form, 'one side' is 2 and 'another side' is 3, resulting in 'perimeter: 10'. In the right form, 'one side' is 6 and 'another side' is 7, resulting in 'perimeter: 26'.

Before going to Exercise 2, compare the code in `ct_Ex1.js` with your `img_01.jpg`, i.e., the flowchart you drew in Exercise 1 of Part 1. Are they conceptually the same? If the answer is “no”, something may be wrong. Either modify your flowchart to be a good match to this code or provide a js code which is equivalent to your flowchart. However, note that in the flowchart, we do not go to details of languages.

For instance, we assume “+” means *addition* as it normally does in real world; but, if in a programming language “+” has been overloaded, that’s the responsibility of the programmer (not the designer of the flowchart) to use it properly. As another example, the designer—in their flowchart—does not get involve in how w and h should be inputted, and where the result should be outputted. The designer just states to input w and h and output the result based on w and h. Its the task of the programmer to translate the flowchart to an appropriate statement in the target programming language, here js; elsewhere, java, python, c, etc. This is true for many other components in the design. One advantage of drawing a flowchart first is that you focus on the design, the process of computational thinking, instead of getting distracted by errands of the programming language, such as how to input, how to convert from string to number, what the right syntax is, etc. This becomes very critical when you want to tackle bigger projects.

Exercise 2. Copy `ct_Ex1.html` and `ct_Ex1.js` to new files named `ct_Ex2.html` and `ct_Ex2.js`.

In this exercise, you should translate your flowchart of Exercise 2 of Part 1 to js. You should make some changes in both html and js files, such that you get the following results.

The image shows two side-by-side screenshots of a web form titled 'volume of an ellipsoid'. Each form has three input fields: 'a', 'b', and 'c', each with a small up/down arrow. Below the inputs is a red 'enter' button. At the bottom of each form, the result is displayed in green text. In the left form, 'a' is 1, 'b' is 2, and 'c' is 3, resulting in 'volume: 25.13'. In the right form, 'a' is 3, 'b' is 4, and 'c' is 6, resulting in 'volume: 301.59'.

In particular, you need to make some changes in your `ct_Ex2.html`, including (but may not limit to):

- link your html file to your new js file
- Change the header text accordingly
- Add a new input (because this program receives 3 inputs) with id “num3”
- Correct the labels of the inputs to a, b, and c

- As semi-axes of an ellipsoid cannot be negative numbers change the min and max of textbox to 1 and 100
- When the “enter” button is clicked, the event handler volumn() should be triggered.

Also, you need to make changes in your **ct_Ex2.js**, including (but may not limit to):

- Make sure name of the function is volumn()
- Make sure pre/post conditions are updated to reflect how this function works
- Calculate volume with what you have in your flowchart for the formula. Use **Math.PI** for π value (we will talk about Math in JS later, but you search something like ‘PI in JS’ to get more info)

Note that you may get 25.132741228718345 for input 1 2 3. But we want the answer to be fixed to 2 digits after the decimal point. This should be addressed in the implementation, not necessarily the design (i.e., flowchart). Explore the toFixed() function in JS.

Before moving on to Exercise 3, compare your code in **ct_Ex2.js** with your **img_02.jpg**, your flowchart for this problem. Make sure they are match.

Exercise 3. Copy **ct_Ex2.html** and **ct_Ex2.js** to new files named **ct_Ex3.html** and **ct_Ex3.js**.

In this exercise, you should translate your flowchart of Exercise 3 of Part 1 to js. You should make some changes in both html and js files, such that you get the following results.

The image shows two identical web forms side-by-side, both titled "largest of three". Each form has three input fields labeled 'a', 'b', and 'c', and a button labeled 'enter'. Below the inputs, the result is displayed in green text.

Input a	Input b	Input c	Output
4.5	5.6	43.2	largest : 43.2
-4.5	-5.6	-63.2	largest: -4.5

In particular, you need to make changes in your **ct_Ex3.html**, including (but may not limited to):

- Change the header text accordingly
- Change back min and max of the textbox to -32768 and 32767, respectively
- When the “enter” button is clicked, the event handler largestOf3() should be triggered.

Also, you need to make changes in your **ct_Ex3.js** accordingly. Make sure the pre/post conditions are updated to reflect how this function works. Also since the inputs can be floating point number, you should parse the input from string to floating point number, otherwise you will get wrong results for some inputs (such as the three negative inputs on the right image above).

Before moving on to Exercise 4, compare your code in **ct_Ex3.js** with your **img_03.jpg**, your flowchart for this problem. Make sure they are match.

Exercise 4. Copy **ct_Ex3.html** and **ct_Ex3.js** to new files named **ct_Ex4.html** and **ct_Ex4.js**.

In this exercise, you should translate your flowchart of Exercise 4 of Part 1 to js. You should make some changes in both html and js files, such that you get one of the following results depending on the inputs.

The event handler isequation(). Make sure sure pre/post conditions are updated to reflect how this function works.

Before moving on to Exercise 5, compare your code in **ct_Ex4.js** with your **img_04.jpg**, your flowchart for this problem. Make sure they are match.

about roots of a quadratic equation	about roots of a quadratic equation	about roots of a quadratic equation
a <input type="text" value="1"/> b <input type="text" value="3"/> c <input type="text" value="2"/> <input type="button" value="enter"/>	a <input type="text" value="2"/> b <input type="text" value="2"/> c <input type="text" value="2"/> <input type="button" value="enter"/>	a <input type="text" value="2"/> b <input type="text" value="4"/> c <input type="text" value="2"/> <input type="button" value="enter"/>
it has 2 distinct roots	it has no roots in real numbers	its roots are identical

Exercise5. Copy `ct_Ex4.html` and `ct_Ex4.js` to new files named `ct_Ex5.html` and `ct_Ex5.js`.

In this exercise, you should translate your flowchart of Exercise 5 of Part 1 to js.

Make necessary changes to the html file, including comment out the textbox for b and c as we only take one input here. Also change the header text accordingly. The event handler is `divisibleBy6()`.

In the JS, following the flowchart, the function should have two *if* clauses, one after another -- first check if the input is divisible by 6, and output accordingly, and then further check if the input is 0 or not. If it is not 0, nothing happens and the function ends. If the input is 0, also output "program ended", and disable the button (so that nothing happens when clicking, thus the function cannot be called anymore) and function ends. Hint: the output text for input 0 can be a concatenation of the first checking result (0 is divisible by 6 not) and "
 program ended". Don't forget to update the pre and post condition for the function.

divisible by 6	divisible by 6	divisible by 6	divisible by 6
a <input type="text" value="23"/> <input type="button" value="enter"/>	a <input type="text" value="24"/> <input type="button" value="enter"/>	a <input type="text" value="48"/> <input type="button" value="enter"/>	a <input type="text" value="0"/> <input type="button" value="enter"/>
not divisible by 6	divisible by 6	divisible by 6	divisible by 6 program ended

button is disabled now

Exercise6. Copy `ct_Ex5.html` and `ct_Ex5.js` to new files named `ct_Ex6.html` and `ct_Ex6.js`. In this exercise, you should translate your flowchart of Exercise 6 (both part A and B) of Part 1 to js.

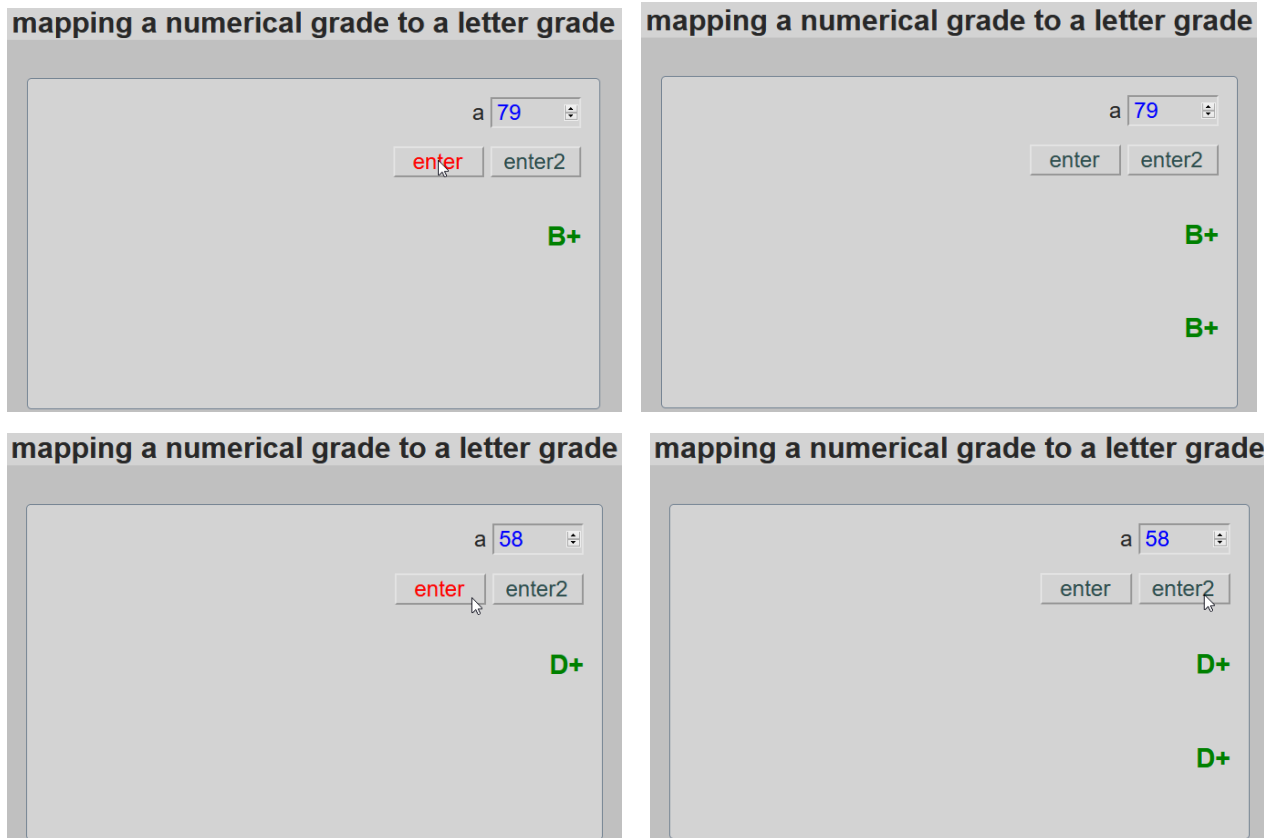
Make the following changes to the HTML file:

- Change the min and max of the textbox to 0 and 100 respectively.
- Add a new button with text "enter2", beside the original button.
- Below the original output paragraph, add another output paragraph, with id "output2".
- When the first enter button is clicked, trigger the event handler `mappingA()`, which uses `if else` to map the grade (corresponds to flowchart for part 6A). When the newly added `enter2` button is clicked, trigger the event handler `mappingB()`, which do the mapping using `switch case` – no `if else` should be used.

In the JS, implement function mappingA(), and mappingB().

- mappingA() maps the grade using a bunch of *if elseif elseif ...else*, and outputs the letter grade in the original output paragraph (whose id is “output”).
- mappingB() maps the grade using *switch case* – no *if else* at all. Partially implemented code is given, uncomment them and complete it by adding more cases. The function then then outputs the result in the new paragraph (whose id is “output 2”)

If implemented correctly, both functions should give the same mapping results.



Again, don't forget to change header text in html and change pre-post conditions of the functions in JS.

Now, compare your mappingA() code in **ct_Ex6.js** with your **img_06A.jpg** and your mappingB() code with **img_06B.jpg** which show the flowcharts for this problem. Make sure they are match. That means that, in both mappingA() code and **img_06A.jpg**, you should use several if statements instead of a switch, and in both mappingB() code and **img_06B.jpg**, you should use switch instead of several if statements.

G. AFTER-LAB WORKS (THIS PART WILL NOT BE GRADED)

In order to review what you have learned in this lab as well as expanding your skills further, we recommend the following questions and extra practices:

- 1) You should revisit the 7 exercises after the lab and learn about the parts of your flowcharts that were not a good match to your js code.
 - a. You may want to do some sort of reverse engineering here: study the js code that you eventually developed for each exercise and draw a flowchart for that. Be careful not to include any JavaScript specific notation in your flowcharts. Flowcharts should be as independent as possible from programming languages. That means your flowchart should be understandable to anyone who knows programming even if he/she does not know any JavaScript.
 - b. Hence, your flowchart should not use functions like parseFloat, getElementById, etc., or keywords like if, else, var, etc. You also should not use "=" for an assignment; instead you should use "←". The "=" is used for comparisons, and for that you should not use JS notation "==" for comparison.

- 2) Once you have your own 7 flowcharts and corresponding js codes polished, take photos (or screenshots) of each and add them to your myLearningKit webpage that you started in lab3, such that when buttons Problem01 to 06 are clicked, your corresponding solutions are shown. For Problem01~05, put your flowchart for 'design' image, and your JS solution for 'JavaScript solution' image. For problem06, use one of the flowchart for 'design' image and put one of the two JS solutions to the 'JavaScript solution' image, and another JS solution to the 'Another solution' image.
- 3) We will provide you with sample solutions a bit later. The sample solution should NOT be used as a means of learning how to tackle those exercises. Instead, it should be only used as a reference to compare your own solution with our solution and learn from differences. In general, you cannot learn much computational thinking skills, if any, by studying a solution without putting your efforts first to come up with a solution (even a non-perfect one). As an analogy, no one can learn how to ride a bicycle practically by watching (even 100s of hours of) how others do it. This is true for many other skills including computational thinking.

Please feel free to discuss any of these questions in the course forum or see the TAs and/or Instructors for help.

H. Submissions.

You should already have a **Lab04** folder that contains the following files:

ct.html, ct.js, ct.css

ct_Ex{0,1,2,3,4,5,6}.html and **ct_Ex{0,1,2,3,4,5,6}.js**

If you haven't done so, copy the 7 images of your flowcharts to this folder as well

img_{01,02,03,04,05,06A,06B}.jpg

Make sure size of each image file is not more than 0.5MB, otherwise you may not be able to upload your work

Your HTML should pass the HTML validator at <https://validator.w3.org>

Compress the Lab04 folder (zip or tar.gz), and then submit the (single) compressed file on eClass.