

#Important reminders

- 1) You **must** attend **your assigned** lab session (we will be marking your submission in the lab).
- 2) Do the mini-quiz prelab quiz. It will open on Wednesday by 13:00 and will expire on Thursday by 12:59. It's part of your lab grade. Each lab counts to 2% of your overall grade.
- 3) You must arrive on time – anyone later than 15 minutes may not be admitted to the lab.
- 4) Submitting your lab work to Moodle is required. Otherwise, you receive 0 even if you show the results to your TA.

#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` statement in js
 - memory space (aka variable), js operators such as `+` and overloaded operators
 - flowchart symbols as described in the lecture notes
- 3) don't forget to do the mini-quiz by Thursday at 12:59pm.

1. GOALS/OUTCOMES FOR LAB

- To practice **computational thinking** and implement it in js

2. LAB – TASK

- 1) You first and major task in this lab is to draw 8 flowcharts. This task must be done in teams of two (not less not more). While you are done, you should show your flowcharts to the TAs before you go to next part. The TA may ask you to make minor modifications to your flowcharts to demonstrate your computational thinking skills in those contexts.
- 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 first 5 flowcharts to js.
- 3) You will generate at least five `html` and `js` files in this process. You should demo each HTML file to the TA. For that, please, have each `html` file open in a different tab so you can show the progression.
- 4) See next pages for details on how to modify your `html` and `js` files.

3. SUBMISSIONS

1) Manual verification by TA

You will need to have one of the TAs verify your lab before submission. The TA will look at your various files in their progression. The TA may ask you to make minor modifications to the lab to demonstrate your knowledge of the materials.

The TA will mark your name off a list and ask you to sign that you have been verified.

2) Moodle submission

You will see an assignment submission link on Moodle. Create a **folder** named “**Lab4**” and copy all of your lab materials inside (`img_{01,02,03,04,05,06,07,08}.jpg`, `ct_Ex{1,2,3,4,5}.html` and `ct_Ex{1,2,3,4,5}.js`). This folder should be compressed (or `tar.gz` on the VirtualBox machines) and the compressed file submitted.

Part 1: This exercise must be done in teams of two (not less not more). If you have done it at home, you must discuss it with a peer from your lab before you show your final solution to your TA.

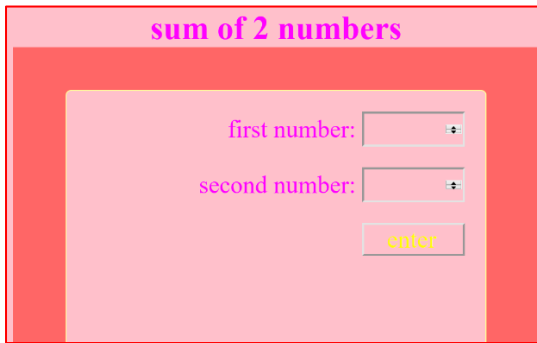
Using a computer program (or a pen and pencil), draw the following flowcharts and write your name on each. By end of this lab, you should take a screenshot (or a picture) from each flowchart and both you and your teammate should submit them to moodle as `img_{01,02,03,04,05,06,07,08}.jpg` files, where `img_x` is the flowchart of 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 slides, for all 8 following exercises.

- Ex 1) draw a flowchart for a computer program to receive two numbers and output their sum.
- Ex 2) draw a flowchart for a computer program to receive three numbers and store them in memory spaces called `a`, `b`, and `c` as three sides of a triangle and, by using Heron's formula, calculate and output the area of the triangle. You may need to refresh your memory or learn about Heron's formula by visiting <https://www.mathopenref.com/heronsformula.html>
- Ex 3) draw a flowchart for a computer program to receive three numerical coefficients of a quadratic equation (store them in memory spaces called `a`, `b`, and `c`) and calculate and output its roots. (assume coefficients are good enough such that a solution in real number exists. Don't worry about cases that a solution does not exist). If you need to refresh your memory on this topic, this might be a good source: <https://www.mathsisfun.com/algebra/quadratic-equation.html>
- 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. This page might be a good reference: <https://www.math10.com/en/algebra/quadratic-equation.html>
- Ex 5) draw a flowchart to receive a number and map it to a letter grade based on York standard. You may need to look at this reference: <http://calendars.registrar.yorku.ca/2012-2013/academic/grades/index.htm> Assume if the grade is 48 or 49, it's mapped to E.
- Ex 6) draw a flowchart to continue receiving numbers and output if they are positive or negative until a zero is entered. When a zero is entered, the program should stop (not receiving any more input).
- Ex 7) by modifying your flowchart above, draw a flowchart to continue receiving numbers and output if they are positive or negative until a zero is entered. When a zero is entered, the program should output how many positive and how many negative numbers have been entered, then it stops.
- Ex 8) draw a flowchart to continue receiving numbers and output if they are divisible by 6 or not until a zero is entered. When a zero is entered, the program should output how many of the entered numbers were divisible by 6, then it stops. **IMPORTANT RESTRICTION:** you are not allowed to divide the number by 6; therefore, you are not allowed to use the modulus operator (%) over 6 to verify the remainder when the number is divided by 6. You can do any other math trick you wish.

Show all your flowcharts to your TA before going to Part 2. The TAs may ask any of the teammates questions about the flowcharts, they may also ask you to modify your flowcharts slightly.

Part 2: you are given **ct.html**, **ct.css**, **ct.js** files. By reading these files carefully, you can enhance your learning.



sum of 2 numbers

first number:

second number:

enter

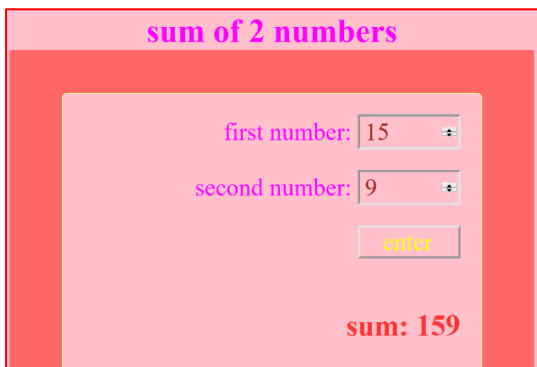
Exercise 1. Copy **ct.html** to a new file named **ct_Ex1.html**. Copy **ct.js** to a new file named **ct_Ex1.js**.

Launch **ct_Ex1.html** with your browser and enter two numbers and click on the “enter” button, nothing happens. Let’s fix it:

Make three changes to **ct_Ex1.html**, as follows:

- 1) Connect it to **ct_Ex1.js** by adding a link in the head element.
- 2) Add an event to the button such that when it’s clicked you handle that event by the `add()` function in your `js` file.
- 3) Add your name to the list of authors of this page.

Launch **ct_Ex1.html** with your browser and enter 15 and 9. You should see the following result:



sum of 2 numbers

first number: 15

second number: 9

enter

sum: 159

That is because the variables `w` and `h` in your `js` function 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 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 and another meaning is to add two numbers. Because `w` and `h` are currently strings, “+” concatenates them, and the result is 159. `parseInt()` is a function that receives a string as its argument and returns its equivalent as an integer number. In other words, it can receive “15” and return 15. It can also receive “9” and return 9.

In summary, “15”+“9” results in “159”. But, 15+9 results in 24. So, open your **ct_Ex1.js** file and by using `parseInt`, modify the code such that sum of the two inputs are calculated, not their concatenations.

sum of 2 numbers

first number: 15

second number: 9

enter

sum: 24

Before moving on to Exercise 2, compare the code in **ct_Ex1.js** with your **img_01.jpg**, the flowchart you drew in Exercise 1 of Part 1. Are they conceptually the same? If the answer is “no”, something is 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 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. The designer just states to input 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 everything else in the design. One advantage of drawing 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 attack 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 when, for instance, you enter 10, 11, and 4 for the sides of the triangle.

area of a triangle when you enter three good sides

a: 10

b: 11

c: 4

enter

area: 19.96

In particular, you need to make about 6 changes in your **ct_Ex2.html** as follows:

- Make sure you link your html file to your new js file
- Change the header to be “area of a triangle when you enter three good sides”
- 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 side of a triangle cannot be negative numbers change the min and max to 1 and 100
- When the “enter” button is clicked, the event handler `area()` should be triggered.

Also, you need to make about 4 changes in your **ct_Ex2.js** as follows:

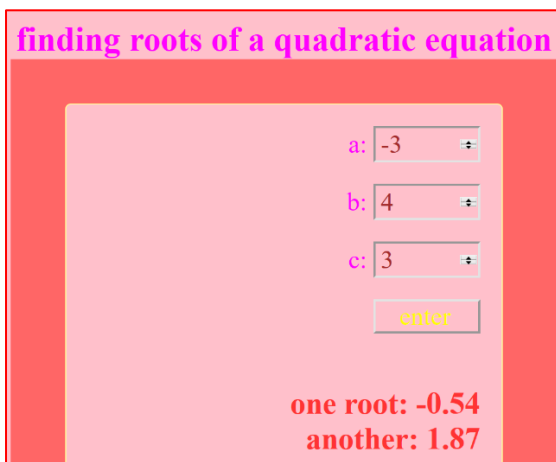
- Make sure name of the function is `area()`
- Add a line to capture the value of `c`
- Replace the line `s=w+h;` with what you have in your flowchart for Heron's formula
- Correct the output to have a more relevant label and correct value as the picture above shows

Note that the answer is fixed to 2 digits after the decimal point. This should be addressed in the implementation, not necessarily the design.

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 when, for instance, you enter -3, 4, and 3 as the coefficients of the quadratic equation.



finding roots of a quadratic equation

a:

b:

c:

one root: -0.54
another: 1.87

In particular, you need to make about 4 changes in your **ct_Ex3.html** as follows:

- Make sure you link your html file to your new js file
- Change the header to be "finding roots of a quadratic equation"
- Change back min and max to -32768 and 32767, respectively
- When the "enter" button is clicked, the event handler `equation()` should be triggered.

Also, you need to make about 3 changes in your **ct_Ex3.js** as follows:

- Make sure name of the function is `equation()`
- Replace the lines you had for heron's formula; with what you have in your **img_03.jpg**
- Correct the output to have a more relevant label and correct value as the picture above shows

Again, note that the answer is fixed to 2 digits after the decimal point. Also note that roots are shown in different lines.

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.

about roots of a quadratic equation

a:

b:

c:

it has 2 distict roots

about roots of a quadratic equation

a:

b:

c:

it has no roots in real numbers

about roots of a quadratic equation

a:

b:

c:

its roots are identical

In particular, you need to make 2 changes in your **ct_Ex4.html** as follows:

- Make sure you link your html file to your new js file
- Change the header to be “about roots of a quadratic equation”

Also, you need to make 2 changes in your **ct_Ex4.js** as follows:

- Replace the lines you had for calculating the roots with what you have in your **img_04.jpg** to determine if the equation has one, two, or no roots in real numbers
- Correct the output to reflect the message

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.

Exercise 5. 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.

In particular, you need to make 3 changes in your **ct_Ex5.html** as follows:

- Make sure you link your html file to your new js file
- Change the header to be “mapping a numerical grade to a letter grade”
- When the “enter” button is clicked, the event handler `mapping()` should be triggered.

Also, you need to make 2 changes in your **ct_Ex5.js** as follows:

- Make sure name of the function is `mapping()`
- Remove codes from line 9 to the comment above the switch statement
- Uncomment the block that the switch statement is in

Now, save the changes and launch **ct_Ex5.html** with your browser. If you enter any number greater than 79, it is correctly mapped to an A or A+; Also, if you enter any number less than 48, it is correctly mapped to F. But, if you enter, any number between 48 and 79, it's not mapped correctly. Your task is to fix this issue.

mapping a numerical grade to a letter grade

input:

A

In particular, you need to make one big change in your **ct_Ex5.js** as follows: add seven more cases to the **switch** statement to map other grades to B+, B, C+, C, D+, D, and E correctly.

Now, compare your code in **ct_Ex5.js** with your **img_05.jpg**, your flowchart for this problem. Make sure they are match. That means if in your flowchart, you used several **if** statements instead of a **switch**, you should change either your **js** code or your flowchart.

Exercise 6 (further practice, for bonus):

You may want to continue this project and add at least two more flowcharts (from exercises 6-8 of part 1) and corresponding **js** implementations for up to 10% bonus point.

Hint. Even though in the flowcharts of exercise 6-8, you probably have **while** loops, you do not need to use the **while** loop of **js** because your code is event-driven: every time the button is clicked, next iteration of the loop is conducted.