

IS216 Large Lab Test

[25 marks]

General Instructions:

- This is a time-bound (2 hours), open-book, open-Internet, and **individual** test.
- You must test your web pages using **Google Chrome Web Browser** Version 118.0.x or later. Your graders will be using only **Google Chrome Web Browser** (Version 118.0.x or later) to test your web pages.
- **No questions will be entertained** by the IS216 teaching team (faculty/instructor/invigilators) during the test period. If necessary, make your own assumptions and proceed to complete test questions.
- You must **use only standard HTML5, CSS, Bootstrap (Version 5.3), JavaScript, Axios, and Vue.js (Version 3)** in your solutions unless the question specifies otherwise. Do not use any other third-party libraries (e.g. Angular, React, or others).
- Use meaningful names for HTML class/id and JavaScript variables and functions/methods. You must indent your code (HTML/CSS/JavaScript) properly. Failure to do so will attract a penalty of up to **20%** of your score for the corresponding question.
- You **MUST** include your name as author in the comments of all your submitted source files. Failure to do so will attract a penalty of up to **20%** of your score for the corresponding question. For example, if your registered name is "KIM Jong Un" and email ID is kim.jongun.2022, include the following comment at the beginning of each source file you write.

HTML files	CSS, JavaScript files
<pre><!-- Name: KIM Jong Un Email: kim.jongun.2022 --></pre>	<pre>/* Name: KIM Jong Un Email: kim.jongun.2022 */</pre>

- You may wish to comment out the parts in your code which cause errors. Commented code will not be marked.

Academic Integrity

- All student submissions will be thoroughly checked by an SMU-approved source code plagiarism checker software program AND an additional external software program. The source code checking will be conducted across all submissions (from 10 sections of IS216).
- Suspected plagiarism cases will be reported immediately to the IS216 faculty in charge and SCIS Dean's Office for further investigation.
- Students in the suspected cases will be informed accordingly by their section faculty, and the incident will be escalated to the SMU University Council of Student Conduct.
- More information about the SMU Student Code of Conduct can be found at this link: <https://smu.sharepoint.com/sites/ucsc/Documents/Code%20of%20Academic%20Integrity.pdf>

Submission Instructions

- **Due Date**

- **22 November 2023 (Wednesday), 4:00 PM Singapore Time**
- Late submission policy is as follows:

Submit within 5 minutes of set deadline	10% penalty of your entire test's score
Beyond 5 minutes	0 mark

- **Zip** up all files in **Q1/Q2/Q3/Q4** folders into **<YOUR_SMU_ID>.zip**
 - For example, **kim.jongun.2022.zip**
 - **Verify by unzipping this zip file – check the content inside**
 - **Incorrect submission file name** WILL attract a penalty of up to **20%** of your score for the entire test.
- Only **zip** format is accepted.
 - **.7z, rar** or other **compression formats** are **NOT** accepted.
 - Until the correct **zip** format is submitted again by the student, it will be assumed that the student has NOT made the submission and late submission policy will apply.
- Submit the **zip** file to the following location:
 - **IS216-MERGED** eLearn page: <https://elearn.smu.edu.sg/d2l/home/365191>
 - Go to **Assignments** → **Large Lab Test** → **Submit your ZIP file**
 - **It is your (student's) individual responsibility to ensure that the zip file submission was successful.**
 - **Your section faculty and Teaching Assistants will NOT verify the submission for you.**

Legend



Do NOT edit this given resource file.






Your answer/code goes here into this given resource file.

Q1. CSS + Vanilla JavaScript

[7 Marks]

Given resources in folder Q1 :

-  q1.html
-  q1.css
-  q1.js

Instructions

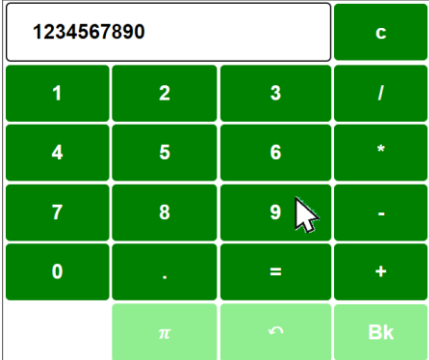
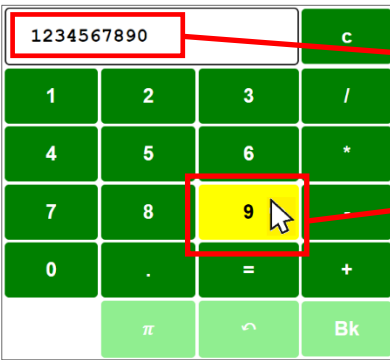
- You are to edit only q1.css and q1.js.
- Your solution will be graded based on the correct output and behavior of your HTML page. No partial marks will be given for incomplete/incorrect solution.

View q1.html in a browser and study the 3 given files to understand how the calculator works. All the buttons work except those in the last row (π , Bk and \curvearrowright).

Part A: (2 marks)

Change q1.css so that:

- The display area uses courier font. (0.5 mark)
- When the mouse hovers over any button, it turns from green/lightgreen-with-white-text to yellow-with-black-text. (1.5 marks)

Original q1.css	After q1.css is corrected
	 <p>Font = courier</p> <p>Button becomes yellow-with-black-text on mouse over</p>

Part B: (5 marks)

Edit `q1.js` so that the 3 buttons on the last row (π , Bk and \curvearrowright) behave as follows:

- π (Pi) appends "3.14" into the display area. (1 mark)
- Bk (Backspace) clears the last character in the display area and does nothing if the display area is blank. (2 marks)
- \curvearrowright (Undo) puts back the last expression in the display area just before the last time the = button was clicked. \curvearrowright does not do anything if the = button has never been clicked. (2 marks)

These screenshots show how these 3 buttons should work. Pressing the following buttons at the left column in order will result in the display at the right column:

How π (Pi) works	
7, *, 7, *, π	7*7*3.14
1, π , 6	13.146


How Bk (Backspace) works	
1, 2, 3, 4, 5, Bk	1234 ("5" got deleted)
π , Bk	3.1 ("4" got deleted)
Bk	 (Nothing happens)
10, +, 20, =, Bk	3 ("0" got deleted)


How \curvearrowright (Undo) works	
1, 2, 3, 4, \curvearrowright	1234 (Nothing happens since = was never clicked)
1, 2, +, 3, =	15 (Display answer)
1, 2, +, 3, =, \curvearrowright	12+3 (Reverts to display before = was last clicked)
1, 2, +, 3, =, *, 6, \curvearrowright	12+3 (Reverts to display before = was last clicked)
1, 2, +, 3, =, *, 6, =, \curvearrowright	15*6 (Reverts to display before = was last clicked)
1, +, 2, =, 9, C, 5, \curvearrowright	1+2 (Pressing the C button doesn't affect how undo works)

Q2. Basic Vue.js

[5 Marks]

Given resources in folder Q2 :

 q2.html

 q2.js

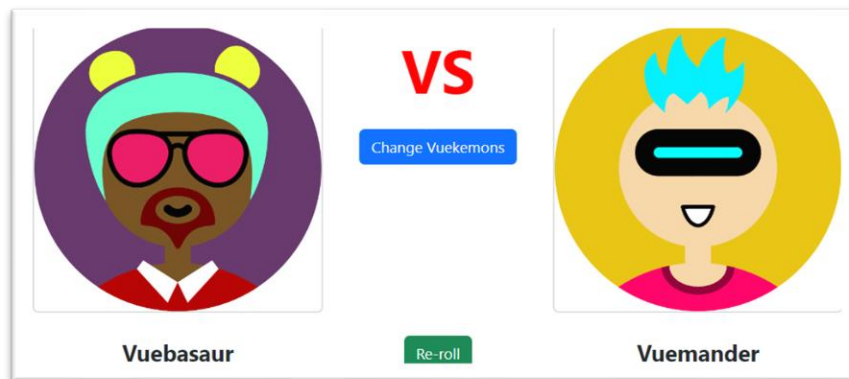
Instructions

1. You are to use Vue.js to answer the parts below.
2. Your solution will be graded based on the correct output and behavior of your HTML page. No partial marks will be given for incomplete/incorrect solution.
3. Open q2.html in a browser and examine how it looks.
4. Study q2.html and q2.js carefully before attempting the parts below.

Part A (2 marks):

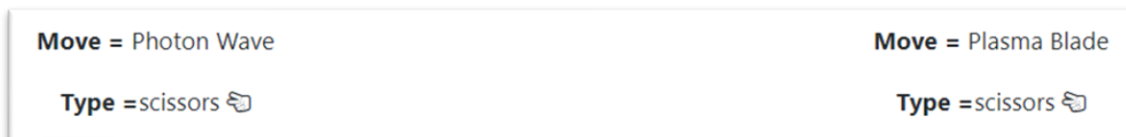
1. Using Vue.js, add code to q2.html to show the following output:
 - Image of vuekemon (use relative path given in object)
 - Name of vuekemon below image

Expected output:



2. There are errors in q2.html, at span tags with class "moveInfo". Modify the code to show the following output:
 - The vuekemon's currentMove's label
 - The vuekemon's currentMove's type

Expected output:



Note:

- The data used for Part A is vuekemonA and vuekemonB, that are both initialized to randomly reference 2 different objects out of the 3 objects in the vuekemons array variable (found in q2.html). Hence, for Part A, you **DO NOT** need to change q2.js.
- Refreshing the page or clicking on the "Change Vuekemons" button are 2 ways to test out Part A. No action is needed on your part for this.


Part B (2 marks):

1. Complete `determineWinner()` in `q2.js` to determine the winner of each round using the usual "Rock-Paper-Scissors" rule. `determineWinner()` should update the variable `roundWinner` found under the Vue instance `data` object with the correct character found in the table below i.e. "A" if `vuekemonA` wins, "B" if `vuekemonB` wins or "" (empty string) if it is a tie.

this.roundWinner			
vuekemonA. currentMove.type \ vuekemonB. currentMove.type	rock	paper	scissors
rock	" "	"B"	"A"
paper	"A"	" "	"B"
scissors	"B"	"A"	" "

2. In `q2.html`, conditionally display the `h3` tag with class "winner" based on the return value of the `winnerMsg(player)` method.

Expected output:



Vuekachu

Current Move

Move = Photon Wave

Type = scissors 🗡️

VS


Change Vuekemons

Re-roll

1 - 0

Round 1

% win for A - % win for B



Vuemander

Current Move

Move = Heat Emperor

Type = paper 🖐️

Round Winner!

To code: Conditionally display h3 tag

Part C (1 mark): Complete the code in `q2.js` for the computed properties `percentWinA` and `percentWinB`, such that the percentage wins of both players are correctly calculated and formatted.

Note:

- In the HTML, the two computed properties `percentWinA` and `percentWinB` (along with display for scores and round number) are already coded in and displayed under the "Re-roll" button. You **DO NOT** have to make any changes to the HTML.
- A player's percentage win should be calculated as the **number of times that player won divided by the total number of rounds played**.
- It should be displayed on the browser to **1 decimal place** e.g. 100.0%, 33.3%
- An example of a successful output is given below:

(1) First roll (automatically performed on initialization).

Result: vuekemonA wins. % win of A is 100.0% of 1 round played.

Vuemander <i>Current Move</i> Move = Heat Emperor Type = paper 📄	<div>Re-roll</div> 1 - 0 Round 1 100.0% - 0.0%	Vuebasaur <i>Current Move</i> Move = Spore Bomb Type = rock 🪨
<div>Round Winner!</div>		

(2) Second roll (triggered using the "Re-roll" button).

Result: vuekemonB wins. % win of both A and B are 50.0% of 2 rounds played.

Vuemander <i>Current Move</i> Move = Heat Emperor Type = paper 📄	<div>Re-roll</div> 1 - 1 Round 2 50.0% - 50.0%	Vuebasaur <i>Current Move</i> Move = Thorn Whip Type = scissors ✂️
<div>Round Winner!</div>		

(3) Third roll (triggered using the "Re-roll" button).




Result: No player wins. % win of both A and B are both 33.3% of 3 rounds played.

Vuemander <i>Current Move</i> Move = Heat Emperor Type = paper 📄	<div>Re-roll</div> 1 - 1 Round 3 33.3% - 33.3%	Vuebasaur <i>Current Move</i> Move = Mud Ball Type = paper 📄
--	--	--

Q3. Bootstrap, JavaScript and AXIOS

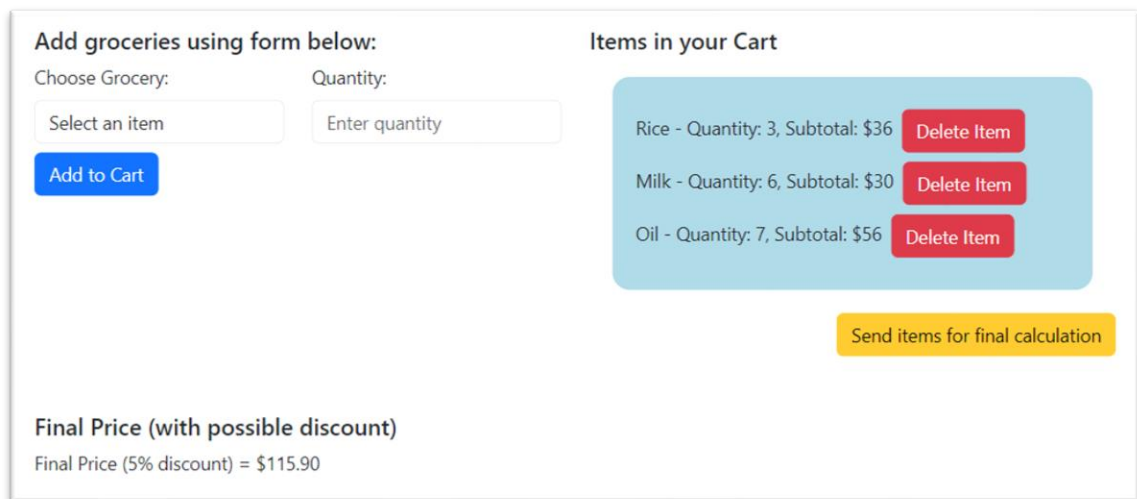
[7 Marks]

Given resources in folder Q3 :

-  q3.php
-  q3.html
-  q3.js

Instructions

1. Your solution will be graded based on the correct output and behavior of your HTML page. No partial marks will be given for incomplete/incorrect solution.
2. The following screenshot is an example of what the **final working app** should look like at ≥ 992 px.



The screenshot displays a web application interface for a grocery cart. It is divided into two main sections: 'Add groceries using form below:' and 'Items in your Cart'.

Add groceries using form below:

- Choose Grocery:** A dropdown menu with 'Select an item'.
- Quantity:** A text input field with 'Enter quantity'.
- Add to Cart:** A blue button.

Items in your Cart:

- Rice - Quantity: 3, Subtotal: \$36 **Delete Item**
- Milk - Quantity: 6, Subtotal: \$30 **Delete Item**
- Oil - Quantity: 7, Subtotal: \$56 **Delete Item**

Send items for final calculation: A yellow button.

Final Price (with possible discount)

Final Price (5% discount) = \$115.90

3. Part A is independent of Parts B, C and D. i.e. even if the Bootstrap question part A is not done, you can still do Parts B, C and D, and vice versa.

Part A (2 marks):

- When the browser width is **≥ 992 px (lg breakpoint)**, the form should appear as shown below:

Add groceries using form below:

Choose Grocery:

Quantity:

Items in your Cart

Final Price (with possible discount)

- "Choose Grocery" and "Quantity" fields should occupy **3 columns** of Bootstrap 12-columns grid, each.
 - "Items in your Cart" div should occupy **6-columns** of Bootstrap 12-columns grid.
 - The "Send items for final calculation" button should be justified to the right.
- When the browser width is **< 992 px (lg breakpoint)**, the form should appear as shown below:

Add groceries using form below:

Choose Grocery:

Quantity:

Items in your Cart

Final Price (with possible discount)

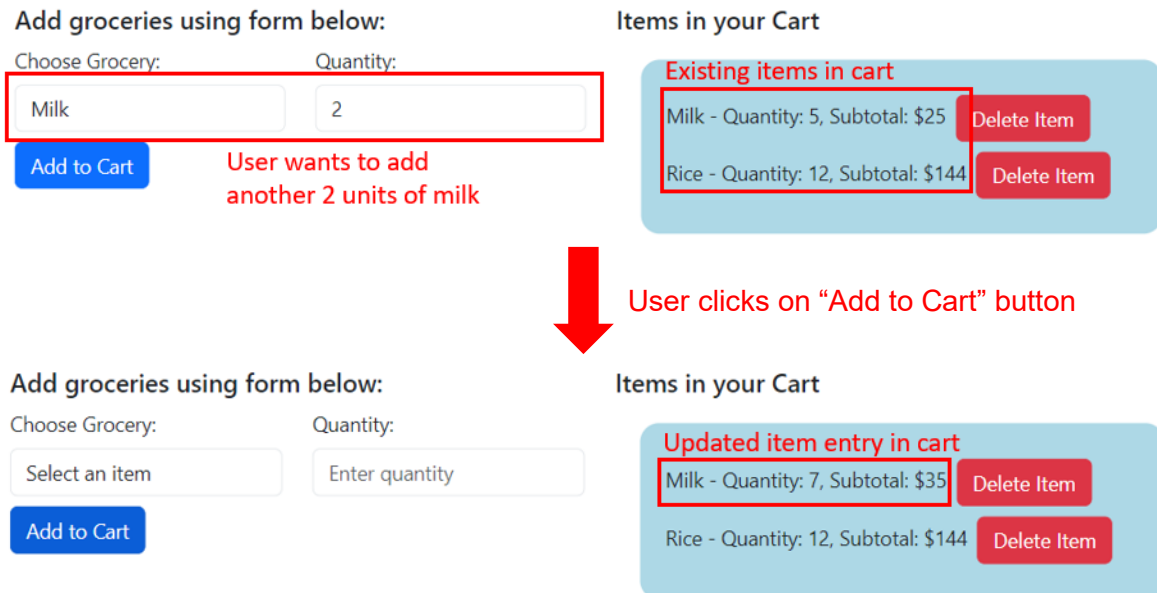
- "Choose Grocery", "Quantity" and "Items in your Cart" fields should occupy the full width of the screen.
- The "Send items for final calculation" button should be centralized.

Note:

- You are only required to **add Bootstrap classes** in `q3.html` to achieve the desired page layout.
- You are **NOT** to use JavaScript to achieve the web page layout required.
- You are not required to add/edit other styles, such as margin, padding, font style, font size, etc., not mentioned in the requirements.

Part B (2 marks): When the user selects an item from the dropdown list of "Choose Grocery", as well as input a non-zero quantity, a div should be added under the div of id="cart", with the following specifications:

- The added div should have an id of "rice", "milk" or "oil", depending on which item is added.
- The text to display in the added div should look like these examples: e.g.
 - Milk - Quantity: 5, Subtotal: \$25
 - Rice - Quantity: 12, Subtotal: \$144
- After adding an item e.g. 5 units of milk added, if the user adds the same item again, the corresponding div should be updated with the new quantity and subtotal, instead of having another div added i.e.



Note:

- You are only supposed to **add vanilla JavaScript** in the `addToCart()` function in `q3.js`.
- You are NOT to modify the HTML portion in anyway, nor are you required to.
- Use the constants `riceBasePrice`, `milkBasePrice` and `oilBasePrice` defined at the top of the `q3.js` file. Otherwise, the base price of each item is coded into the HTML of the dropdown list also e.g. `<option value="Rice" data-price="12">Rice</option>`

Part C (1 mark): Add a "Delete Item" button in the div element created for every new item added to the cart. When the user clicks on the "Delete Item" button, it should delete the item's div element from the "cart" div.

Example hierarchy (with incomplete HTML, showing only hierarchical relationship):

- `<div id="cart">`
 - `<div id="milk">`
 - Text e.g. Milk - Quantity: 7, Subtotal: \$35
 - `<button>Delete Item</button>`

For styling, use the Bootstrap classes `"btn btn-danger ms-2"` for your "Delete Item" button.

Note:

- You are only supposed to **add vanilla JavaScript** in the `addToCart()` function in `q3.js`.
- You are NOT to modify the HTML portion in any way, nor are you required to.

Part D (2 marks): Complete the `showFinalDetails()` function in `q3.js` to accomplish the following:

- i) Send the items' quantity and subtotals in the cart to `q3.php` in a `"data"` object, through a **POST** request.
- ii) Display the final total of the items in the cart in the div with `id="finalPrice"`
 - a. with the text: `"Final Price = $..."` **if the sum of the subtotals < \$100.**
 - b. with the text: `"Final Price (5% discount) = $..."` **if the sum of the subtotals >= \$100.**

Examples of the final output are shown below:

- **If sum of the subtotals < \$100:**

Add groceries using form below:

Choose Grocery:

Quantity:

Items in your Cart

Rice - Quantity: 4, Subtotal: \$48

Final Price (with possible discount)

Final Price = \$48

- **If sum of the subtotals >= \$100:**

Add groceries using form below:

Choose Grocery:

Quantity:

Items in your Cart

Rice - Quantity: 4, Subtotal: \$48

Oil - Quantity: 12, Subtotal: \$96

Final Price (with possible discount)

Final Price (5% discount) = \$136.80

Note:

a) This is how "q3.php" expects your POST request:

- A "data" object containing an array of objects should be sent, each with **at least one key-value pair of**

```
{ subtotal : Actual item subtotal (dollar sign not required) }
```

e.g.

```
[
  { subtotal : 35}, // for milk item of quantity = 7, subtotal = 35
  { subtotal : 32}, // for oil item of quantity = 4, subtotal = 32
]
```

b) This is how "q3.php" will send back a successful response:

- An object containing the following key-value pairs:
 - `discountReached` : true or false (based on whether the sum of subtotals \geq a threshold of \$100)
 - `finalTotal` : a number containing the sum of subtotals, with a discount of 5% applied if the sum of subtotals \geq a threshold of \$100

c) "showFinalDetails()" function will be called when the "Send items for final calculation" button is clicked on. This feature is already coded for you.



d) You are only supposed to **add vanilla JavaScript** in the `showFinalDetails()` function. You are NOT to modify the HTML or CSS portion in any way, nor are you required to.

e) A different PHP file with different discount threshold and percentage will be used to test your code, so you have to use `q3.php` to answer this question, instead of "hard-coding" your answer without using the PHP script's response.

Q4: Vue Components

[6 Marks]

Given resources in folder Q4 :

 q4.js q4.html

q4.js contains the Vue code used by q4.html. Read through both files quickly to understand them.

Instructions

1. You are to edit only q4.js.
2. No hardcoding. During grading, the array of videos defined in the Vue instance's `data` option will be modified; we expect the output to change correspondingly.
3. Your solution will be graded based on the correct output and behavior of your HTML page. No partial mark will be given for incomplete/incorrect solution.

Part A: (no marks, but will affect parts B & C if not correctly done)

Study the `<vote-video>` element at q4.html and determine which attributes are being used to pass data about each video into the `vote-video` Vue component. Then edit the `vote-video`'s component definition in q4.js to insert the relevant props.

Part B: (4 marks)

For now:

- The `<iframe>` is showing a fixed video since the value for `src` is hardcoded.
- The component shows a fixed index (0) and title ("SMU Software Engineering") since both are hardcoded.
- When the user clicks on the video's title, a new window opens to display a fixed video.

Edit the `<vote-video>` component so that it shows the current video in the `<iframe>` and the current video's index and title. When the user clicks on the title, a new window opens to display the current video.


Part C: (2 marks)

For now, nothing happens when the user clicks on the ❤️ button of each `<vote-video>`. Edit the `vote-video`'s component's definition so that when clicked, the ❤️ button emits an event called "like" with the video's `index` as event payload. This `like` event will trigger the `onLike()` method of the Vue instance to increase the relevant video object's `likes` property.


Here are screenshots of `q4.html` (with videos playing) with the original and correctly-modified `q4.js`.

q4.html with original q4.js			q4.html with correctly- modified q4.js. The number of Likes in the table correspond to the number of times each video's ❤️ is clicked.		
Index	Title	Likes	Index	Title	Likes
0	Prof Kyong dancing	0	0	Prof Kyong dancing	5
1	History of SCIS	0	1	History of SCIS	2
2	SCIS Voix	0	2	SCIS Voix	5


Click on the ❤️ of each video.



❤️ Video Index: 0 [SMU Software Engineering](#)




❤️ Video Index: 0 [SMU Software Engineering](#)




❤️ Video Index: 0 [SMU Software Engineering](#)


Click on the ❤️ of each video.



❤️ Video Index: 0 [Prof Kyong dancing](#)



❤️ Video Index: 1 [History of SCIS](#)



❤️ Video Index: 2 [SCIS Voix](#)

End of Test