IS216 Sample Large Lab Test (Set C) [40 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 94.0.x or later). Your graders will be using only **Google Chrome Web Browser** (Version 94.0.x or later) to test your web pages.
- No questions will be entertained by the IS216 teaching team (faculty/instructor/Teaching Assistants) 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 5.1), 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. 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.2020, include the following comment at the beginning of each source file you write.

HTML files	CSS, JavaScript files

`<!--`
`Name: KIM Jong Un`
`Email: kim.jongun.2020`
`-->`
	`/*`
	`Name: KIM Jong Un`
	`Email: kim.jongun.2020`
	`*/`

- 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 11 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 **HERE** (or at https://smu.sg/2020-is216-smu-code).

Submission Instructions

- **Due Date**
  - **XX October 2021 (Friday) XX:XX PM Singapore Time**
  - Late submission policy is as follows:

Submit **within 5 minutes** of set deadline	10% penalty (of your score for the entire test)
Submit **within 10 minutes** of set deadline	25% penalty (of your score for the entire test)
Submit **within 15 minutes** of set deadline	50% penalty (of your score for the entire test)
Beyond 15 minutes, 0 mark (submission will NOT be accepted)	

- **Zip** up all files in **Q1/Q2 folders** into **<YOUR_SMU_ID>.zip**
  - For example, **kim.jongun.2020.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/303340

    IS216-Web Application Development II-Merged Section - 2021-221IS216_MERGED_SECTION
  - Go to **Assignments → XYZ →** *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.

IMPORTANT

Inside **resources** folder, you will find **Bootstrap** files. **Please do NOT edit these files.**

- ➔ `Bootstrap\`
  - 🚫`bootstrap.bundle.min.js`
  - 🚫`bootstrap.min.css`

Q1. [Vue.js] Big Bang [15 marks]

Given resources:

- ✏️ `q1.html`
- ✏️ `q1.js`
- 🚫 `bb_gd.jpg, bb_taeyang.jpg, bb_top.jpg`

Part A

Edit **`q1.html`** to have a webpage that does the following:

1. Contains radio buttons for the selection of **Celebrity** (there are a total of THREE (3) celebrities).
    a. You cannot hard-code their info. You must retrieve it from the Vue instance in **`q1.js`**.
2. Contains checkboxes for the selection of **Activities**.
    a. You cannot hard-code their info. You must retrieve it from the Vue instance in **`q1.js`**.
    b. All prices must be displayed in 2 decimal places.
3. The "Bill Section" (below <hr>) should only be **shown** if the **total_bill** computed property (in the Vue instance) is a non-null value.
    a. Do not modify **total_bill** computed property implementation in this **Part A**.
    b. Use the given implementation as is. HINT: Observe what the computed property returns and think about how you can leverage it from `q1.html`.

Your **`q1.html`** must display as below:

Part B

Edit **`q1.html`** and **`q1.js`** so that **`q1.html`** displays the Bill Table in an HTML table that looks like below.

The "Bill Table" comes from **total_bill** computed property in **`q1.js`**.

1. By default, it returns **null**.
2. Modify the implementation of **total_bill** computed property such that it returns an **HTML table**.
3. Subsequently, modify **`q1.html`** such that the **HTML table** is correctly displayed.
4. **NOTE**:
    a. The "Bill Table" is displayed only if a celebrity is selected AND at least 1 activity is selected.
    b. All prices and the total bill amount must be displayed in 2 decimal places.

Your Bill

Your Fan Request	
	Taeyang
Hug	$30.25
Total	$30.25

Sample Output (q1.html)

Step 1: Select Celebrity		
		
◉ G-Dragon	○ Taeyang	○ TOP
Step 2: Choose Activities		
☐ Chat ($10.50)	☑ Hug ($30.25)	☑ Kiss ($60.99)

Your Bill

Your Fan Request	
	G-Dragon
Hug	$30.25
Kiss	$60.99
Total	$91.24

Q2. [JavaScript] Hiking [25 marks]

Given resources:

- `q2.html`
- `q2.js`
- `q2-D.html`

Ariana keeps track of her hikes by recording down how many steps she takes and whether each step was **up** (U) or **down** (D). A hike starts at **sea level** and ends at **sea level**. Each step **up** or **down** represents ONE (1) unit change in altitude. We define the following terms:

- A **mountain** is a sequence of consecutive steps above sea level, starting with a step up from sea level and ending with a step down to sea level.
- A **valley** is a sequence of consecutive steps below sea level, starting with a step down from sea level and ending with a step up to sea level.

For instance, given a hike path '**DDUUUUDD**':

- She first enters a valley TWO (2) units deep (down).
- After that, she climbs out TWO (2) units high and then continues onto a mountain TWO (2) units high (up). She returns to sea level and ends her hike.

Part A

In **q2.js**, complete the implementation of the function `get_level()`. It takes the hiker's hike **path** (e.g. **DDUUUUDD**) as a String from the **textarea** (in **q2.html**), and it returns the level at which the hiker is situated (at the end of the hike).

- If the level at which the hiker is situated is ZERO (0), it means the **path** is a **VALID path** (since we define a valid path to be the one where the hiker ends the hike at **sea level**).
- If the level at which the hiker is situated is a NEGATIVE NUMBER, it means the **path** is an **INVALID path**. It means that the hiker is still in a **valley**. The hike did NOT end yet and we declare this an INVALID path.
- If the level at which the hiker is situated is a POSITIVE NUMBER, it means the **path** is an **INVALID path**. It means that the hiker is still on a **mountain**. The hike did NOT end yet and we declare this an INVALID path.

Given a path **DDUUUUDD**, the function must return ZERO (0).
Given a path **DDDDUUU**, the function must return NEGATIVE ONE (-1).
Given a path **UUUUDD**, the function must return POSITIVE TWO (2).

Part B

In **q2.js**, complete the implementation of the function `count_valleys()`. It takes the hiker's hike **path** (e.g. **DDUUUUDD**) as a String from the **textarea** (in **q2.html**), and it returns the **total number of valleys** she walked through.

Given a path **DDUUU**, the function must return ONE (1).
Given a path **DDDDUUU**, the function must return ZERO (0).
Given a path **DDDUUUDDUU**, the function must return TWO (2).

Part C

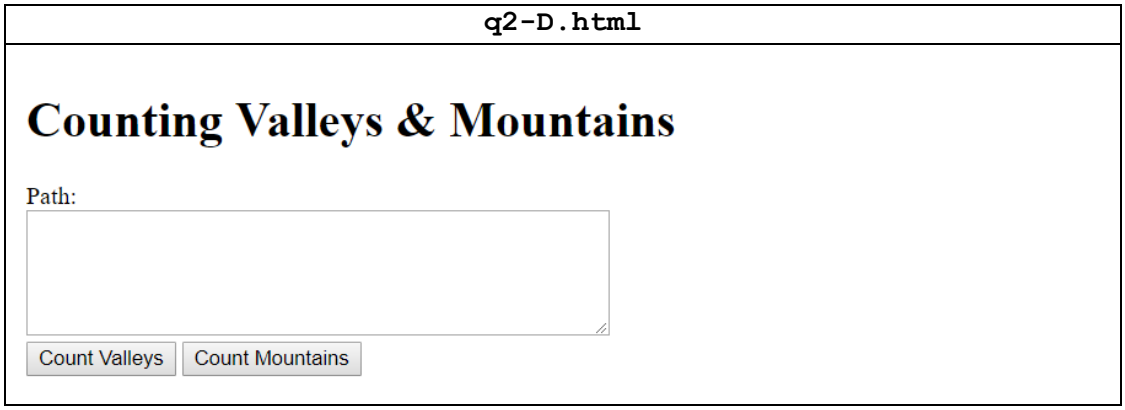
In **q2.js**, complete the implementation of the function `count_mountains()`. It takes the hiker's hike **path** (e.g. **DDUUUUDD**) as a String from the **textarea** (in **q2.html**), and it returns the **total number of mountains** she walked through.

Given a path **UUUDDD**, the function must return ONE (1).
Given a path **UUUUDDD**, the function must return ZERO (0).
Given a path **UUUDDDUUDD**, the function must return TWO (2).

Part D

When page **`q2-D.html`** loads for the first time, it looks like the following:

`q2-D.html`

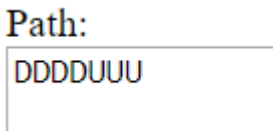
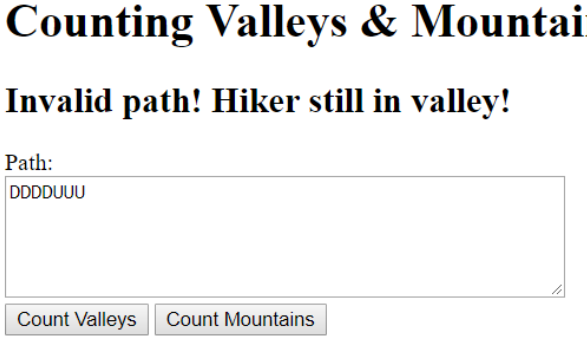
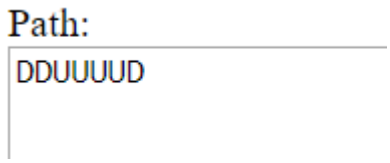
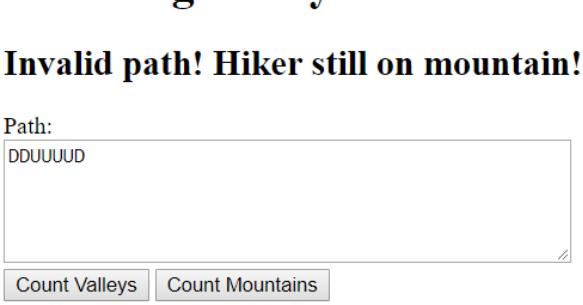


The page has:

- A textarea (empty)
- A button which displays “Count Valleys”
- A button which displays “Count Mountains”

Edit **`q2.js`** so that it performs the following:

Sample Output

`q2-D.html` **(before button click)**	**`q2-D.html`** **(after button click)**
 **Click on either of the two SUBMIT buttons**	
 **Click on either of the two SUBMIT buttons**	

Path: DDDUUU Click on “Count Valleys” SUBMIT button	**Counting Valleys & Mountains** Path: DDDUUU Count Valleys Count Mountains **Number of Valleys: 1**
Path: DDDUUU Click on “Count Mountains” SUBMIT button	**Counting Valleys & Mountains** Path: DDDUUU Count Valleys Count Mountains **Number of Mountains: 0**
Path: UUUDDDUUDD Click on “Count Mountains” SUBMIT button	**Counting Valleys & Mountains** Path: UUUDDDUUDD Count Valleys Count Mountains **Number of Mountains: 2**
Path: DDUUDDUUDDDUUU Click on “Count Valleys” SUBMIT button	**Counting Valleys & Mountains** Path: DDUUDDUUDDDUUU Count Valleys Count Mountains **Number of Valleys: 3**

Part E

This time, a hiker's **path** looks like the following (for example):

SDDUU

- The first character 'S' indicates the **start** of a hike, which occurs at **sea level**.
- Next, she enters a valley TWO (2) units deep (down).
- After that, she climbs out TWO (2) units high and reaches **sea level**.
- And, she is done with the hike.

Edit **`q2.js`** such that the function `print_path()` outputs the hike path. The hike path can be retrieved from the **textarea** (in **`q2-D.html`**).

Upon clicking **"Show Hiking Path"** button, **`q2-D.html`** must correctly display the hike path's **altitude change** as the hike progresses (Left to Right).

***IMPORTANT: You MAY ASSUME** that the user input **path** will always be a **VALID HIKE**. Please test your code against the below listed test cases ONLY.*

q2-D.html (before button click)	**q2-D.html (after button click)**
Path: SDDDUUU Show Hiking Path	S U D U D U D
Path: SDDUUDDDUUU Show Hiking Path	S U U D U D U D D U D
Path: SDDUUUUDD Show Hiking Path	U U D S U D D U D