

IS216 Large Lab Test

[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 103.0.x or later). Your graders will be using only Google **Chrome** Web Browser (Version 103.0.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 5.2), 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 "TAN So Tong" and email ID is tan.sotong.2021, include the following comment at the beginning of each source file you write.

HTML files	CSS, JavaScript files
<pre><!-- Name: TAN So Tong Email: tan.sotong.2021 --></pre>	<pre>/* Name: TAN So Tong Email: tan.sotong.2021 */</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 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
 - 04 November 2022 (Friday) 7:00 PM Singapore Time OR a later time announced by the invigilator.
 - 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, tan.sotong.2021.zip
 - Verify by unzipping this zip file – **CHECK THE CONTENT INSIDE!!!**
 - **Incorrect submission file name** WILL attract up to 20% **penalty** of your entire test's score.
- **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/333446>
IS216-Web Application Development II-Merged Section - 2021-221IS216_MERGED_SECTION
 - 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. Submission of zip files containing incorrect/corrupted files will incur penalty.**
 - Your section faculty and Teaching Assistants will NOT verify the submission for you.

Contents

Q1. CSS	[10 marks]	4
Q2. Wordle Game	[10 marks]	5
2A. Implement function <code>display_character (parentElement, character)</code>	[3 marks]	6
2B. Implement function <code>display_game (tableElement, game)</code>	[5 marks]	7
2C. Implement function <code>check_guess (answer, guess)</code>	[2 marks]	8
Q3. Smart Home	[10 marks]	10
3A: Implement function <code>door1Func ()</code>	[3 marks]	11
3B: Implement function <code>door2Func ()</code>	[3 marks]	13
3C: Implement function <code>couchFunc ()</code>	[4 marks]	14
Q4. Sales	[10 marks]	16
4A: Compute Total Sales	[2 marks]	18
4B: Compute Yearly Team Sales	[3 marks]	18
4C: Implement Vue Component	[5 marks]	19

Legend



Do not edit this given resource






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

Q1. CSS

[10 marks]

Given resources in folder Q1

-  index.html
-  bg.jpg
-  style.css

This question tests your ability to **use vanilla CSS only** to decorate a web page **index.html** using an external CSS file **style.css**. You are to **update the given style.css only** according to the instructions given. **Do not edit the other files.**

- All HTML **span** elements have
 - Blue, double border of 10px thickness,
 - Distance between its content and top/bottom borders is 20px, and distance between its contents and left/right borders is 30px. See figure 1.2 below.
- All HTML elements with **class 'easy'** have
 - 18px Courier text, and
 - Text color is 255 red, 165 green, and 0 blue.
- The HTML element with **id 'simple'** has
 - The image 'bg.jpg' as the background,
 - Text is in bold.

Figure 1.1. When done correctly, **index.html** displays as follows:

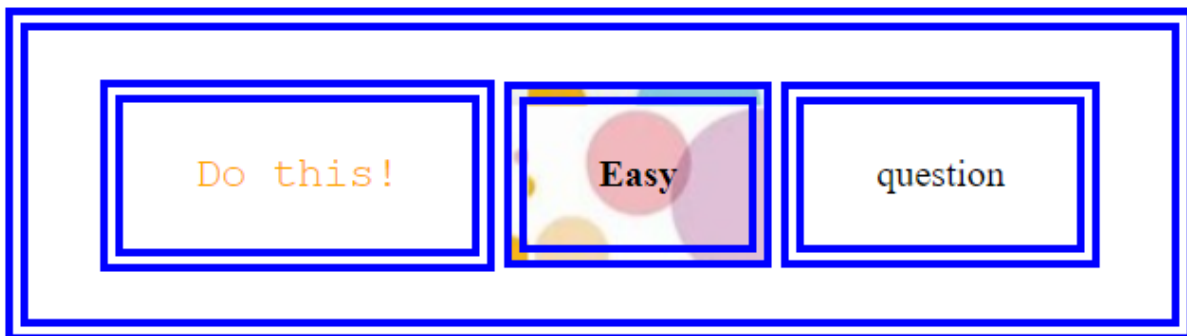
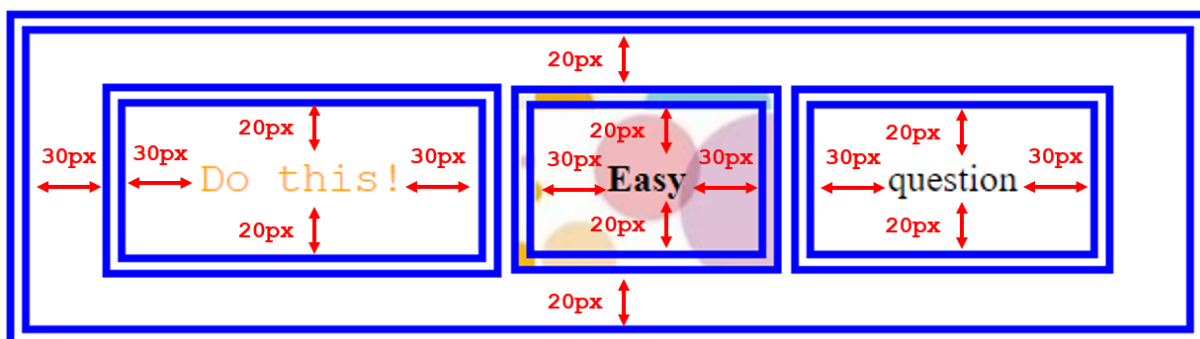




Figure 1.2. Illustration of the gaps between a span's border and its contents (point 1b above):



Q2. Wordle Game

[10 marks]

Given resources in folder Q2

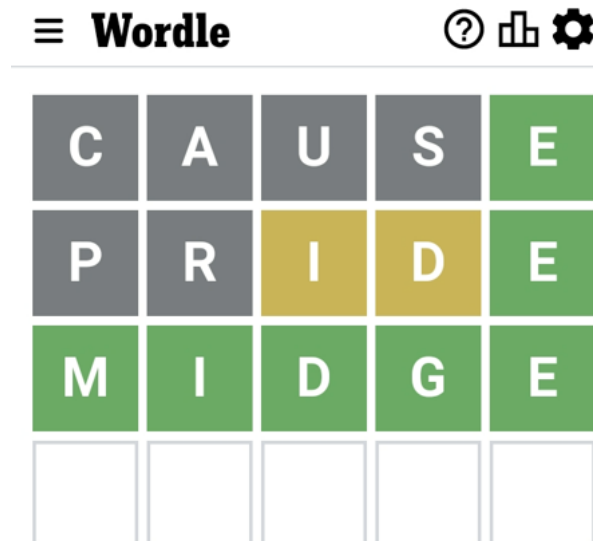
-  test.html
-  app.js

Wordle is a popular web-based game.

Everyday, a five-letter word is chosen which players aim to guess within six tries. After every guess, each letter is marked as either green, yellow or gray: green indicates that letter is correct and in the correct position, yellow indicates that letter is correct but not in the right position, while gray indicates that letter is incorrect regardless of whether it's in the correct position or not.

Multiple instances of the same letter in a guess, such as the "o"s in "robot", will be colored green or yellow only if the letter also appears multiple times in the answer; otherwise, excess repeating letters will be colored gray.

Reference: <https://en.wikipedia.org/wiki/Wordle>



For the screenshot above,

1. CAUSE is the first guess.
 - a. Only E is correct and in the correct position, thus, it is in green.
 - b. The rest are wrong, thus, gray.
2. PRIDE is the 2nd guess.
 - a. E is correct and in the correct position.
 - b. I and D are correct but in the wrong position, thus, yellow.
3. MIDGE is the 3rd guess. It is the secret word, thus, all letters are green.

You will create a simplified version of Wordle by implementing the functions listed below **using vanilla JavaScript only**. The data structure for this application has been designed as follows:

1. `character` is an object that depicts the correctness of a letter in a guess. It has the following properties:
 - a. `status` whose possible values are "correct", "partial", "wrong"
 - b. `letter` is a string

Examples:

- `{status: "wrong", letter: 'R'}`
- `{status: "partial", letter: 'D'}`
- `{status: "correct", letter: 'E'}`

2. `word` is an array of `character` objects.

- a. Example:

```
[ {status: "wrong", letter: 'P'},
  {status: "wrong", letter: 'R'},
  {status: "partial", letter: 'I'},
  {status: "partial", letter: 'D'},
  {status: "correct", letter: 'E'} ]
```
- b. **Do NOT assume or hardcode the number of elements is 5** in the array. This allows us to expand the game to words with more letters.

2A. Implement function **display_character(parentElement, character)** [3 marks]

Implement function `display_character(parentElement, character)` in `app.js`.

1. It takes in the following input parameters:
 - a. `parentElement` is an HTML element
 - b. `character` is a character object.
2. Set `parentElement`'s text to `character`'s property "letter".
3. If `character`'s property "status" is "correct" (case sensitive), set `parentElement`'s background colour to green. For values "partial" and "wrong", set to yellow and grey respectively.

The file `test.html` contains code to check your function `display_character()`. Read through the file to understand how your function `display_character()` is expected to be used.

If done correctly, the following output is expected in `test.html`:

2A: display_character()

Correct letter, correct spot

Background color is green.

A

Correct letter, wrong spot

Background color is yellow.

A

Wrong letter

Background color is grey.

A

2B. Implement function `display_game(tableElement, game)` [5 marks]

Implement function `display_game(tableElement, game)` in `app.js`.

1. It takes in the following input parameters:
 - a. `tableElement` is an HTML table element
 - b. `game` is an array of word arrays; i.e., 2D array of character objects.
2. For each word in the input parameter `game`,
 - a. Create a new `tr` element and append it to the input parameter `tableElement`.
 - b. For each character in `word`,
 - i. Create a `td` in the `tr` element
 - ii. Reuse the `display_character()` to display each character in the `td`.
3. **Do NOT use `innerHTML`** for this function.

The file `test.html` also contains code to check your function `display_game()`. Read through the file to understand how your function `display_game()` is expected to be used.

If done correctly, the following output is expected in `test.html`:

2B: display_game()

C	A	U	S	E
P	R	I	D	E
M	I	D	G	E

2C. Implement function `check_guess(answer, guess)` [2 marks]

Implement function `check_guess(answer, guess)` in `app.js`.

1. It takes in the following input parameters:
 - a. `answer` is a String representing the secret answer in UPPERCASE.
 - b. `guess` is a String representing a user's guess in UPPERCASE.
2. Returns a new `word` array that contains the following `character` objects representing the results of checking each alphabet in `guess` against `answer`.
Each `character` object in the `word` array:
 - a. Property `letter` to a corresponding positioned alphabet in `guess`.
 - b. Property `status` to "correct" if the alphabet is a correct letter in the correct position in `answer`. If correct letter but wrong position, `status` is "partial". Otherwise, `status` is "wrong".
 - i. Multiple instances of the same alphabet in a `guess` will have `status` "correct" or "partial" only if the alphabet also appears multiple times in the `answer`; otherwise, excess repeating alphabets will have `status` "wrong".
 - ii. E.g. the answer is "AGREE" and E appears twice. The user's guess is "EERIE" and has 3 Es.
 1. The last E of the guess "EERIE" is a correct letter in the correct 5th position – last E of answer "AGRE**E**", thus, it has `status` "correct",
 2. The first E of "EERIE" (1st position) has `status` "partial" because it is a correct but in the wrong position – the other E is in 4th position of "AGRE**E**". "Partial" status will only be assigned to the first occurrence of the instance that is not "correct".
 3. As all the Es in the answer have been accounted for, the E in the 2nd position of "E**E**RIE" has `status` "wrong".
 - iii. Refer to the sample output below for file `test.html`.
3. Return the `word` array.

The file `test.html` also contains code to check your function `check_guess()`. If done correctly, the following output is expected in `test.html`:

2C: check_guess()

Answer is "MIDGE"

User's guess is "PRIDE".

```
[
  {
    "letter": "P",
    "status": "wrong"
  },
  {
    "letter": "R",
    "status": "wrong"
  },
  {
    "letter": "I",
    "status": "partial"
  },
  {
    "letter": "D",
    "status": "partial"
  },
  {
    "letter": "E",
    "status": "correct"
  }
]
```

... continue on the right...

... continue from the left...

Answer is "AGREE"

User's guess is "EERIE".

```
[
  {
    "letter": "E",
    "status": "partial"
  },
  {
    "letter": "E",
    "status": "wrong"
  },
  {
    "letter": "R",
    "status": "correct"
  },
  {
    "letter": "I",
    "status": "wrong"
  },
  {
    "letter": "E",
    "status": "correct"
  }
]
```

In the interest of time, we will not implement the full game in this lab test. You may do it yourself after the lab test for extra practice.

Q3. Smart Home

[10 marks]

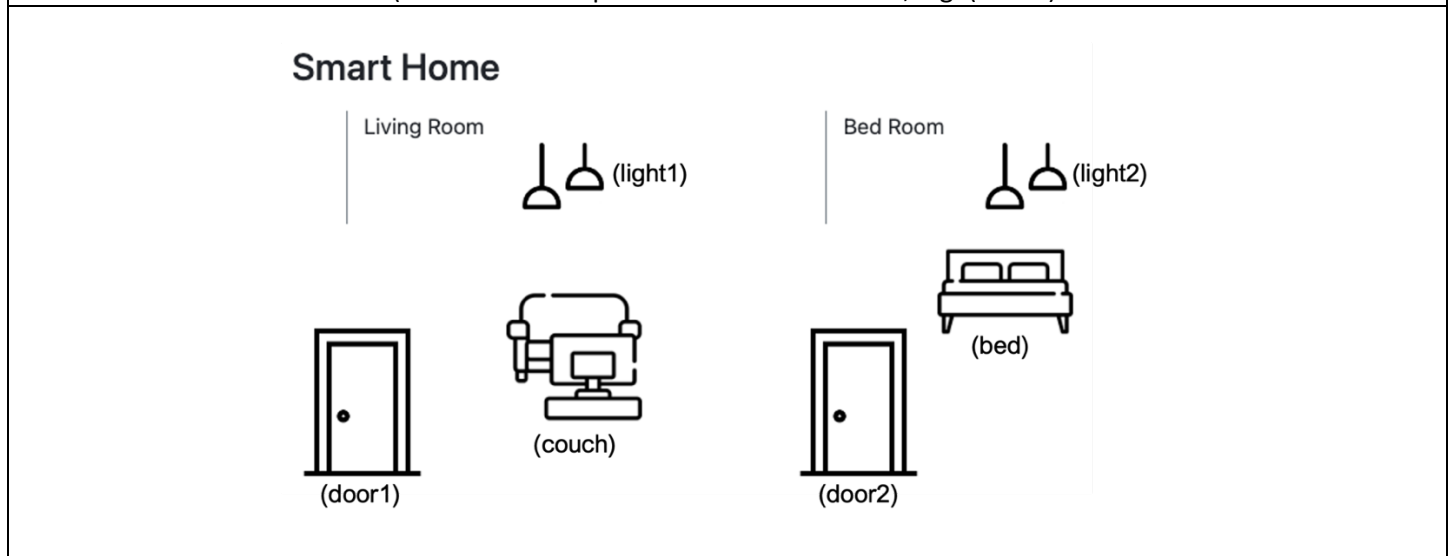
Given resources in folder Q3

-  bootstrap.bundle.min.js
-  bootstrap.min.css
-  image/* [6 image files]
-  app.html
-  app.js

Create a JavaScript program that simulates a *smart home* application, by using **vanilla JavaScript** only.

When `app.html` is run in the browser for the first time, it shows the initial states of the components of a smart home.

(Notice that component names are labeled, e.g. (door1)).



Modify `app.js` to implement the series of scenarios described in the following.



Note:

- Make use of the images given in the 'image' folder.
- Do not need to consider other possible scenarios that are not mentioned.

3A: Implement function `door1Func ()` [3 marks]

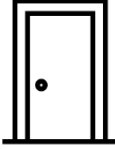


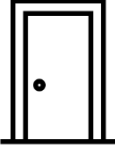


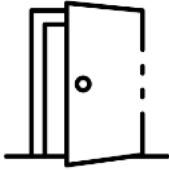


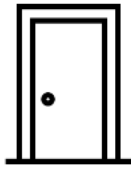


Add an appropriate event listener in an appropriate place and implement the following behavior in `door1Func ()` in `app.js`.

- Whenever *door1* (the door in the living room) is clicked, it toggles. That is, if it was initially closed, it “opens” when clicked, and vice versa.
- Whenever *door1* toggles to “open”, *light1* (the light in the living room) is “on”
- Whenever *door1* toggles to “closed”, the state of *light1* does not change. E.g., if it was initially “on”, it will stay “on”.

- Note:** door “open” means  light “on” means 

You can simulate such behavior by *toggling* the door-close image (`door-close.png`) and the door-open image (`door-open.png`) and by *toggling* the light-off image (`light-off.png`) and the light-on image (`light-on.png`)

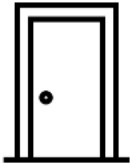
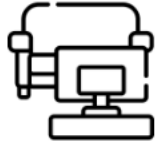
The following illustrates an example run.

app.html	
When <code>app.html</code> is run in the browser for the first time, doors are “closed” and lights are “off”	
<div> <div> <h3>Smart Home</h3> <div> <div>Living Room</div> <div>    </div> </div> </div> <div> <div>Bed Room</div> <div>    </div> </div> </div>	
When <i>door1</i> is clicked, the door “opens” and <i>light1</i> is “on”.	
<div> <div> <h3>Smart Home</h3> <div> <div>Living Room</div> <div>    </div> </div> </div> <div> <div>Bed Room</div> <div>    </div> </div> </div>	

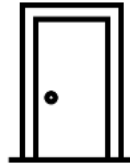
When *door1* is clicked again, *door1* “closes” and *light1* stays “on”

Smart Home

Living Room



Bed Room

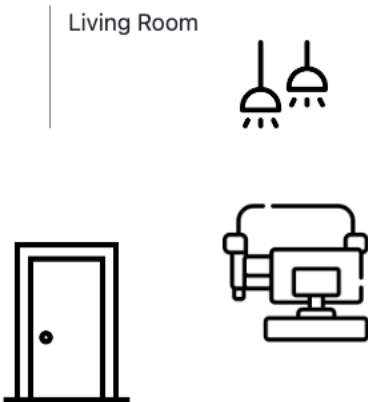
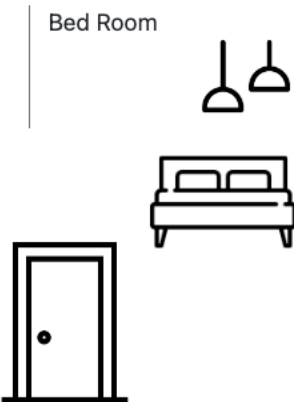
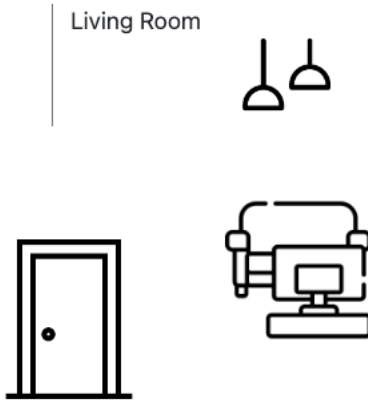
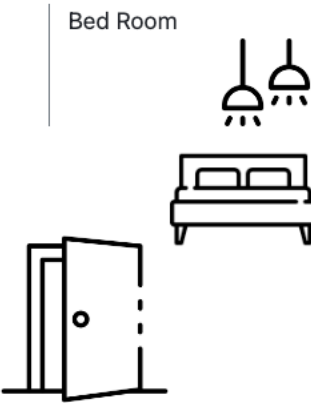


3B: Implement function `door2Func()` [3 marks]

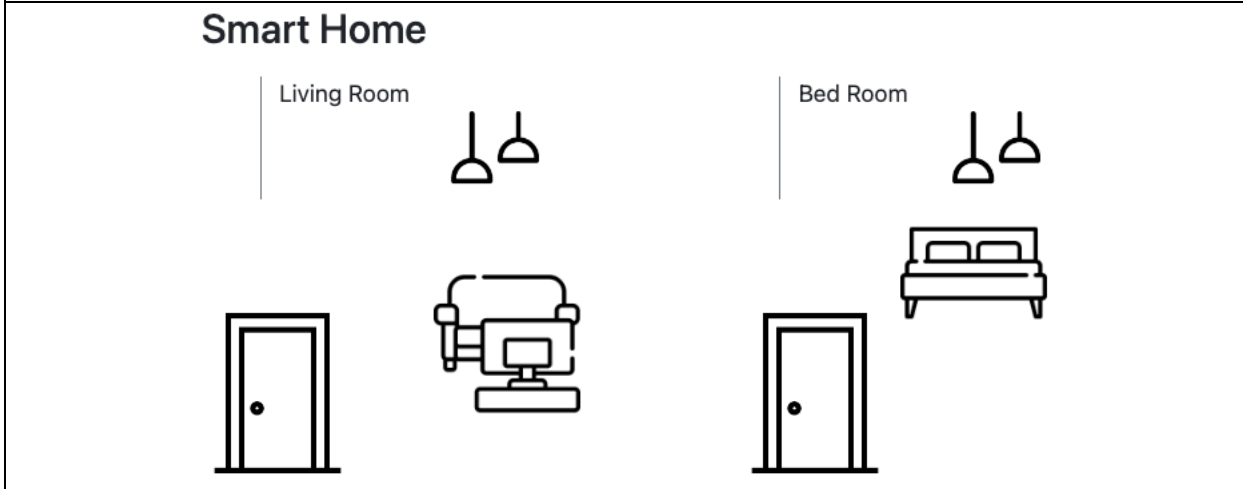
Add an appropriate event listener in appropriate place and implement the following behavior in `door2Func()` in `app.js`

- Whenever *door2* (the door to the bed room) is clicked, it toggles. That is, if it was initially closed, it “opens” when clicked, and vice versa.
- Whenever *door2* toggles to “open”, *light1* is “off” and *light2* (the light in the bed room) is “on”.
- Whenever *door2* toggles to “closed”, *light2* is “off” and the state of *light1* does not change.

The following continues from the example run in Part 3A.

app.html	
Initially (after Part 3A), <i>light1</i> is “on”; <i>door2</i> is “closed”; <i>light2</i> is “off”.	
<div> <div> <h3>Smart Home</h3> <div> <div>Living Room</div> <div>  </div> </div> <div> <div>Bed Room</div> <div>  </div> </div> </div> </div>	
When <i>door2</i> is clicked, <i>door2</i> “opens”; <i>light1</i> is “off” and <i>light2</i> is “on”	
<div> <div> <h3>Smart Home</h3> <div> <div>Living Room</div> <div>  </div> </div> <div> <div>Bed Room</div> <div>  </div> </div> </div> </div>	

When *door2* is clicked again, *door2* “closes” and *light2* is “off”



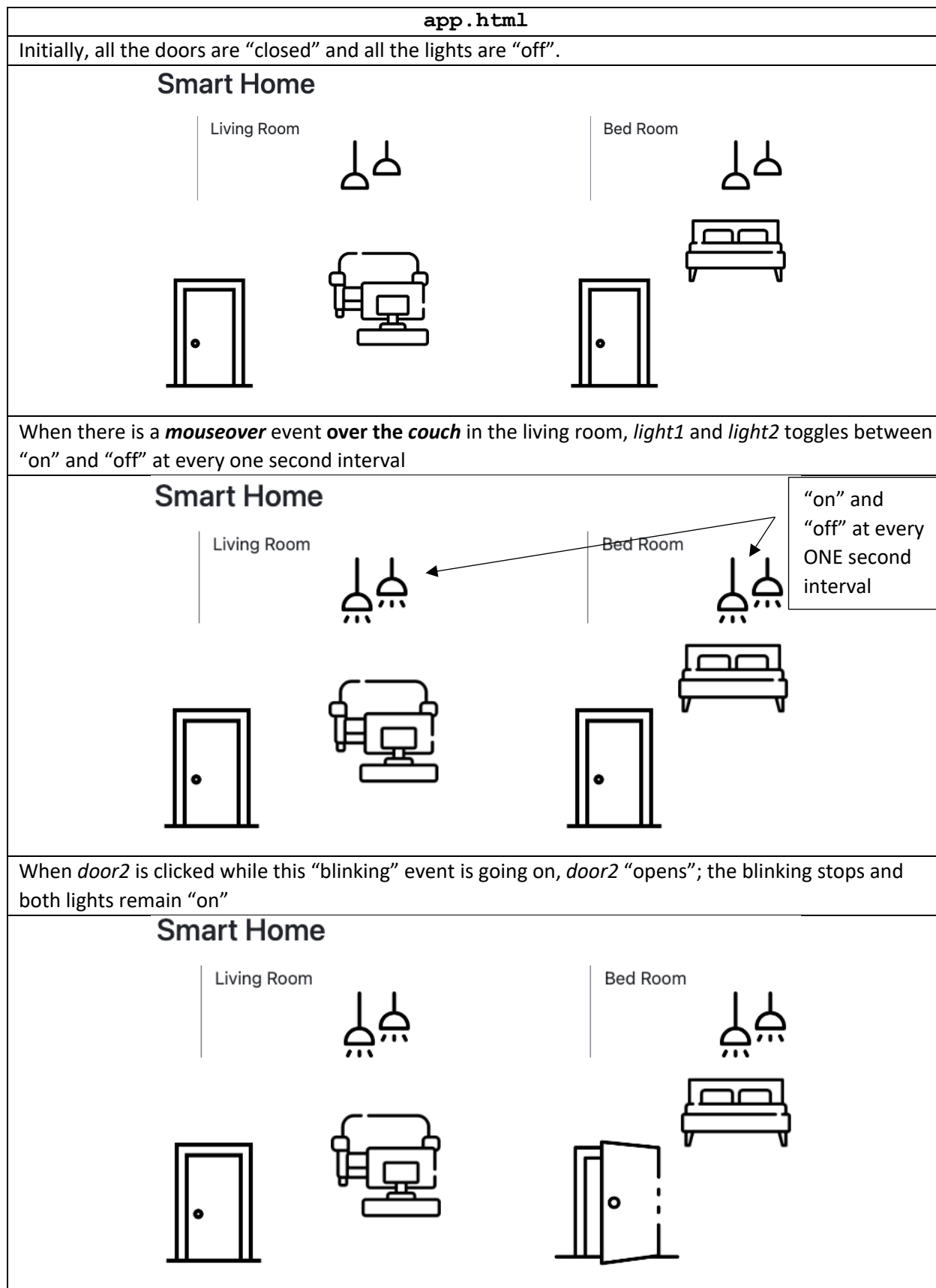
3C: Implement function `couchFunc ()` [4 marks]

Add appropriate event listener(s) in appropriate place(s) and implement the following behavior in `couchFunc ()` in `app.js`.

- When **all the doors are “closed”** and **all the lights are “off”**, if there is a ***mouseover*** event **over the couch** in the living room (simulating the scenario of a thief walking past the living room), *light1* and *light2* “blink”
 - i.e., toggles between “on” and “off” at every ONE second interval (simulating an alarm scenario).
 - Hint: explore `setInterval`
- When *door2* is clicked while this “blinking” event is going on,
 - door2* “opens”
 - the blinking (toggling) of lights stops
 - both lights remain “on”

Note: you may add additional helper functions and variables as you deem fit.



The following illustrates an example run.



Q4. Sales

[10 marks]

Given resources in folder Q4

-  vue.js
-  app.html
-  app.js

Given a “Sales” data of various teams in various years, compute its total sales, compute yearly sales, and display the results.

Note: Do this question using `Vue.js` code only. No JavaScript DOM manipulation code is allowed.

The data option of the Vue application is defined for you in `app.js` and is sufficient to implement the requirements; i.e. do not edit and do not add more properties.

```
...    const app = Vue.createApp({
        // DO NOT EDIT - start
        data() {
            return {
                sales: {
                    team1: {
                        2020: { Q1: 400, Q2: 200, Q3: 300, Q4: 300 },
                        2021: { Q1: 200, Q2: 700, Q3: 400, Q4: 500 },
                        2022: { Q1: 100, Q2: 200, Q3: 300, Q4: 800 },
                    },
                    team2: {
                        2020: { Q1: 200, Q2: 300, Q3: 400, Q4: 200 },
                        2021: { Q1: 400, Q2: 600, Q3: 800, Q4: 800 },
                        2022: { Q1: 100, Q2: 100, Q3: 400, Q4: 800 },
                    }
                }
            }
        },
        // DO NOT EDIT - end
    })
...
```

As shown above, `sales` data property represents the sales of each team for each quarter (Q1, Q2, Q3, Q4) of each year. For example, in **year 2020**, **team1** made the sales of

- \$400 in **Q1**
- \$200 in **Q2**
- \$300 in **Q3**
- \$300 in **Q4**

Hence, for year 2020, the total sales of team 1 is \$1200

Likewise, for year 2020, the total sales of team 2 is \$1100

Important Note:

- Do not HARD CODE the outputs when performing the tasks in Part 4A-C.
- Your code must work when the `sales` data property contains different set of teams and different set of years.
- We will be using automated test cases to test your code. Our automated test cases will contain different sales data. For example,

```

sales: {
  teamA: {
    2018: { Q1: 100, Q2: 100, Q3: 100, Q4: 100 },
    2019: { Q1: 200, Q2: 200, Q3: 200, Q4: 200 }
  },
  teamB: {
    2018: { Q1: 100, Q2: 100, Q3: 100, Q4: 100 },
    2019: { Q1: 200, Q2: 200, Q3: 200, Q4: 200 }
  },
  teamC: {
    2018: { Q1: 100, Q2: 100, Q3: 100, Q4: 100 },
    2019: { Q1: 200, Q2: 200, Q3: 200, Q4: 200 }
  }
}

```

On the other hand, for every team, the set of sale years will be the same. In the above example, every team has the sales data from year 2018 and 2019.

After completing all the tasks, the expected output when running `app.html` in the browser is as follows:

Total Sales: \$9500.00

Yearly Team Sales

```
{ "2020": { "team1": 1200, "team2": 1100 }, "2021": { "team1": 1800, "team2": 2600 }, "2022": { "team1": 1400, "team2": 1400 } }
```

Sales in Year 2020

team1	team2
1200	1100

Sales in Year 2021

team1	team2
1800	2600

Sales in Year 2022

team1	team2
1400	1400

Note: Ignore the text alignment differences (if any) from your browser and the one above.

4A: Compute Total Sales [2 marks]

- Add code in `totalSales` computed property in `app.js` so that it computes and returns the total of all the sales of all the years of all the teams, given in the `sales` data property defined in the Vue application.

If done correctly, the expected output upon loading `app.html` *with respect to this task* is as follows:

Total Sales: \$9500.00

4B: Compute Yearly Team Sales [3 marks]

- Add code in `yearlyTeamSales` computed property in `app.js` so that it computes and returns the yearly sales of each team, given in the `sales` data property defined in the Vue application.
 - Assume that every team has the same set of years.

- The returned data must be in the following format:

```
{ "year" : { "team" : total_sale_of_this_year_by_this_team }, ..., ... }
```

- For example, for the given `sales` data property, `yearlyTeamSales` computed property should return:

```
{
  "2020": { "team1": 1200, "team2": 1100 },
  "2021": { "team1": 1800, "team2": 2600 },
  "2022": { "team1": 1400, "team2": 1400 }
}
```

If done correctly, the expected output upon loading `app.html` *with respect to this task* is as follows:

Yearly Team Sales

```
{ "2020": { "team1": 1200, "team2": 1100 }, "2021": { "team1": 1800, "team2": 2600 }, "2022": { "team1": 1400, "team2": 1400 } }
```

4C: Implement Vue Component [5 marks]

- In `app.js`, `sales-comp` Vue component is defined but incomplete. It takes in two props: `year` and `team_sales` as input.
 - `year` is a string, e.g., "2020"
 - `team_sales` is an object of `key:value` pairs, where `key` is a string that represents a team and `value` is a number that represents the sale amount. An example of `team_sales` is

```
{ "team1": 1200, "team2": 1100 }
```

- Using the data provided in the two props: `year` and `team_sales`, add necessary code and *options*, e.g., *computed property* and *template* in the `sales-comp` component so that when the component is created,
 - it shows the header "Sales in Year <year>" (use `<h2>` tag)
 - it shows the sales of each team for a given year in the table format.
 - the table cell with *the highest sale* among the teams has the **'blue' background color**.
 - If there are more than one team with the same highest sale, all those teams must have the **'blue' background color**.

- For example, running the following code:

```
<sales-comp
  v-bind:year = '2020'
  v-bind:team_sales = '{ "team1": 1200, "team2": 1100 }'>
</sales-comp>
```

Sales in Year 2020

team1	team2
1200	1100

should produce the following output:

If done correctly, the expected output upon loading `app.html` *with respect to this task* is as follows:

Sales in Year 2020

team1	team2
1200	1100

Sales in Year 2021

team1	team2
1800	2600

Sales in Year 2022

team1	team2
1400	1400

~~ END OF PAPER ~~

[THIS PAGE IS INTENTIONALLY LEFT BLANK]