**CSC 336 Web Programming, Spring 2021**

**Prof. Bei Xiao**

Total points (60+10 pts).

Due: March 24 (Wed), 7pm, 2021.

Instructions:

1. Download the midterm code folder in blackboard. Each question contains starter codes that you need.
2. Please upload each question as firstname\_lastname\_QuestionX.zip that contains .js, .css, .html and your resources files. Make sure you upload correctly.
3. The exam is open notes, open source and open book. But you can’t discuss with anyone and **you can’t share your solutions**. Please do not copy solutions from the Internet. **If caught, you will be reported to the authority.**
4. If you finish the exam, please make sure you finalize the submission and leave the room quietly.
5. Please make sure your code compile and run on the browser (use Firefox and chrome to look at your solution).
6. Cell phone and other smart phone devices are forbidden during the exam.
7. No late exam can be graded.
8. Partial credits will be given to all the steps involved. But code which doesn’t compile will receive little points. Make sure your JS code works step by step.
9. All JS code must be unobtrusive to receive full credits.
10. (10Pts) CSS selectors.

Text, letter

Description automatically generated

Part 1: Please write the ids of the elements selected by the given selectors:

1. li
2. ol li
3. ul>li
4. ol,ul
5. #list-1 li

q 1) Part 1:  
  
1. li  
“cookie-1”, “cookie-2”, “list-id”, “cookie-3”, “cookie-4”, “cookie-5”, “cookie-6”  
  
2. ol li  
  
cookie-3, cookie-4, cookie-5, cookie-5  
  
3. ul > li  
  
cookie-1, cookie-2, list-id  
  
4. ol,ul  
  
list-1, list-2, list-3  
  
5. list-1 li  
  
cookie-1, cookie-2, list-id

Part 2: Text

Description automatically generated with medium confidence

Please write down the CSS Selectors that selects the following elements. Your selector must not select other elements in the document. The tags are lettered on the line that they are opened.

1. D
2. E, G
3. A, B, C,D,E,F,G,H,I,J,K
4. H,J,K
5. K,L

q 1) Part 2:  
  
1. .clinton  
  
2. .trump  
  
3. article, section   
  
4. article section  
  
5. section > section, aside

1. (10pts) HTML/CSS coding

Write the CSS code necessary to create the following appearance.

The page uses the HTML code below. You are not allowed to modify the HTML code.

Text

Description automatically generated

<body>

<h1>Set!</h1>

<div class="row">

<div class="card">

<img src="green-diamond-striped.png" alt="green striped diamond" />

</div>

<div class="card">

<img src="green-diamond-striped.png" alt="green striped diamond" />

<img src="green-diamond-striped.png" alt="green striped diamond" />

<img src="green-diamond-striped.png" alt="green striped diamond" />

</div>

<div class="card">

<img src="green-diamond-striped.png" alt="green striped diamond" />

<img src="green-diamond-striped.png" alt="green striped diamond" />

</div>

</div>

</body>

Write the CSS to add styles to this HTML to meet the expected output with the following appearance details. Remember to set flex properties to meet the screenshot output:

● Card positioning:

○ The horizontal spacing for the three cards is achieved by setting the appropriate flex property in the appropriate CSS selector to the space-between value.

● Card details:

○ The divs containing card images are each 200px wide and 75px tall with a solid black border of 0.35em width and a border radius of 15px.

○ The horizontal spacing for the images in each individual card is achieved by setting the appropriate flex property in the appropriate CSS selector to the space-around value.

○ All images should have a specified height of 60px and be vertically centered in each card.

● Heading:

○ The Set! heading is Helvetica font with a default of sans-serif and is centered on the page with 5px of spacing between the text and the top border of the cards

1. (20pts) JS/DOM

Write JavaScript code to add behavior to the following page for keeping track of a to-do-list. The page UI allows the user to type an item into a text box. The user can click the "add" button to add the item to the bottom of the list. Each word in the phrase should be inserted as a li, inside a ul with the id of list. If the user wishes to remove an item he or she can type the text of the item he or she wishes to remove in the text box and click the "remove" button. This should be case insensitive. For example, if the list only contains "foo" and the user tries to remove "FoO", it should be removed. If the user tries to remove an item that is in the list multiple times only the first occurrence should be removed. The items should have background colors that alternate between white and yellow (first white, then yellow, then white, yellow, etc.). This should still be the case no matter how many items are removed or added and no matter what order these operations are done in. You may not use the CSS3 nth-child pseudo selector to do this. The code should work for multiple clicks of the buttons. On each click it should clear any previous information you typed in the input boxes. Do not use any JavaScript libraries such as jQuery or Prototype.

Here is the relevant HTML code for the page:

<h1> My super nifty to-do list </h1>

<ul id=”list”></ul>

<div>

<input type = ‘text’ id=’item’/>

<button id=’add’>add</button>

</div>

The screen shots shows the state after items have been added, and the state after items have been removed.

Hint: write two functions: additems() and removeitems()

(function() {

"use strict";

window.onload = function() {

document.getElementById("add").onclick = addItem;

}

function addItem(){

// your code goes here

}

})();

Graphical user interface, application

Description automatically generated  
4. (20Pts ) Grade calculator

Write **an unobtrusive JavaScript** code to add behavior to the page that has a user interface for entering grades on homework assignments. You can download the HTML and CSS code in the folder Question 1.

You will compute the percentage of points earned, with an optional **curve**. When "Compute!" is clicked, your JS code should use the values in the text boxes to compute the percentage (rounded to the nearest percent). If the **"Curve +5"** checkbox is checked, add +5 percent up to a maximum of 100% total.

You should insert the percentage into the page as a new div added to the end (bottom) of the existing page section with the id of resultsarea. If the overall percentage is 60% or more, give your newly created div a CSS class of pass; otherwise give it a class of fail. Each time the user clicks **"Compute!",** you will insert such a new div; this means that several divs would be there after several clicks of "Compute!".

In the code shown there are 3 assignments, but your code should work for any number of assignments ≥ 1. When **"Clear"** is clicked, all text in all of the input text boxes should be erased. Assume valid input; that is, assume that when "Compute!" is clicked, the user will have already typed valid text into every box that can be interpreted as an integer. You may assume that Prototype is also included in the page.

The current screen shots show the initial state after scores have been entered and Compute has been clicked as well as when the Curve+5 is checked.

Graphical user interface, table

Description automatically generatedGraphical user interface, application, table

Description automatically generated

Hint: Write two functions. ComputeClick() and ClearClick()

You might use : document.querySelectorAll() to access different classes.

5 (Extra credit 10pts) Money Counter.

Use the images of the US bills.

Text

Description automatically generated

Write a page that shows the images. When the user clicks on an image, the page updates a total by the appropriate amount. Here’s some sample output:

Graphical user interface, website

Description automatically generated

Hint: You can follow the pseudo code:

When someone clicks on a bill:  
Get the current total from the page.  
Add the bill's value to the total.  
Show the new total on the page.