Midterm Part 1

Level 0.

On your computer, in the CSC209 folder, make a new folder MidtermPart1 (do not push it to github). In it, create f
Level1, Level2 etc. as you start working on each level of Part 1 of the midterm. Each such folder should have
in it the full code needed to solve the problem, specified for that level.

If you find the strict sequence of levels too slow for your taste, and if you want and can implement several levels at once, you are allowed to jump directly to a higher level. Be advised, though, that:

1. **Your grade will reflect the highest level that works.** In other words, if you jump from Level 3 to Level 8, but Level 8 is not done correctly, your grade will reflect Level 3, not Level 8 (and not even Level 4).
2. It is OK to use features that we did not cover in class (e.g. things that you figured out on your own from W3schools tutorials or other on-line resources, or that you may have already known). However, in such a case, you may be asked by the professor to describe the logic and techiques used in your code, in a one-on-one zoom or in-person meeting, prior to being assigned a grade for the exam.

Level 1.

From W3schools, read and try the example showing how to make a nice Comparison Table. Copy the code in a file *comparisonTable.html* in the Level1 folder.
Check that you can see it on your computer, while running MAMP.
NOTE: *From this level on, I will not repeat this last step and assume that you have checked that each submitted level shows up correctly in a browser.*

Level 2.

Copy the html from Level 1. Extract the css into a separate *comparisonTable.css* file, located in a folder named css in Level2.
NOTE: *From this level on, make sure you have the css style file in this separate folder.*

To do:

1. Figure out how the red cross and the green check sign have been encoded in html/css to show up in the table.
2. Replace the color of the green check sign to be blue.
3. Replace the headers of the table with Name, Part 1 and Part 2, instead of Feature, Basic and Pro.
4. Under the title, add a sentence or two explaining (the best you can) the role of the `link` tag in the head of the html file, what kind of information you think it contributes to the html document and which part of the html document might be using information from the linked file.
5. Figure out how the odd and even rows of the table have been encoded in html to show up in different shades in the table.
6. Replace the colors of the alternate rows with two choices of your own (e.g. white and light yellow).

Level 3.

Copy the Level2 folder (containing as much as you could do for Level2, but correct in the sense that the html

page can be opened and viewed in a browser), rename it as Level3 and modify the files as described below. **NOTE:***From this level on, I will not repeat these instructions and assume that you have copied a previously finished level in which you will work out the required modifications for each specific level.*

Notice that in the W3schools example, in each row there is a green *check* sign in the last column (with header *Pro*). In the *Basic* colum, however, you have either a red-cross or a green-check sign. Your first task is to *generate* one more such row in the table, using javascript. Start by removing all rows in the table, except the row containing the table header and the first row with "Sample text". Your task is now to insert a second row with "Sample text", but not manually (i.e. not by directly editing the html file). Instead, you will be using javascript, according to the following instructions. Insert javascript code in your html file doing the following:

- Declare a variable named ROW, which contains the entire string that creates a row in the table (other than the header row), but slightly modified: instead of the name of the css class that indicates whether a check or cross sign should appear in the row, you will place the string CHECKCROSSBASIC (for column Basic) and CHECKCROSSPRO (for column Pro).
  NOTE: Make sure you do not have hidden characters in this string, such as new line or tabs that could be added if you just copy and paste from the html file. Better to delete those hidden characters or, if you want, you can use the "\n" for a new line, etc.
- Immediately print this string using an alert box.
- Add javascript code to create (in a new variable) the html string describing a new table row. You do this by replacing the strings CHECKCROSSBASIC and CHECKCROSSBASIC in the ROW variable with the css class name for a check, resp. a cross, and storing the result in a new variable. Use alert to print the contents of this variable.
- Finally, append the new row to the innerHTML of the table, using DOM commands. *Hint:* You may need to add an id to the table tag and use the DOM command to "get the element by id", then get its "innerHTML". If you forgot how to do this, search in W3schools for document.getElementById and innerHTML, and follow the examples.
- When done, comment out the alert commands that were used while developing and debugging your code.

Level 4.

Extend the previous level so that the code to add a new row to the table is now in a javascript function addRow, which takes as parameters the desired options to be placed on a row in the BASIC and PRO columns, and inserts the row into the innerHTML of the table.

Level 5.

Extend the previous level so that the javascript function to add a new row to the table is in an external file, which is imported into the html using a script tag.

Level 6.

Extend the previous level so that you can add a number of rows specified by a javascript global variable NRROWS. E.g. you can have NRROWS = 10. Use a *for loop* to add that many rows to the table. For now, each row can have the same entry in the FEATURES, BASIC and PRO columns (next level will handle this).

Level 7.

Extend the previous level so that each row has its own entries in the three colums of the table. You will get these entries from three javascript arrays (stored in global variables): FEATURES, BASIC and PRO. For

instance, if NRROWS = 2, you might have FEATURES = ["Mary","Alice"], BASIC = ["fa-check","fa-remove"] and PRO = ["fa-remove","fa-check"].

Level 8.

Same as previous level, but define and use a function to add all rows (which should invoke in a for loop the function to add a single row).

Level 9: A- level.

Clean up and, if necessary, fix the organization of the code, remove debugging statements, make sure you have properly named variables and function names and all the code is readable and tidy.

Level 10: A level.

If you developed the code as described in the sequence of levels, you may not get alternate colors for the newly added rows in your table. For top credit, I leave it to you to figure out how to do this. **Hint:** use DOM to add elements to the table element, rather than the method described here, which works by manipulating a string and sets the innerHTML.