**MINI PROJECT 2: REVISION COMPONENT (15 POINTS)**

Due Sunday, March 28 (11:59 PM ET)

*Provide your written responses directly in this document; submit your code (Mini Project) with all necessary files; upload to Canvas.*

**Student name: YOUR NAME HERE**

å

**In Mini Project 2, you are asked to submit a revision and elaboration of Mini Project 1 using advanced CSS**. For this project, you will do 2 things: first, complete peer critiques in assigned small groups to get feedback on how to revise and improve your Mini Project 1 submission, and second, submit a revised version of your Mini Project 1 that integrates the feedback you receive and applies advanced techniques in CSS using layout styles.

This document contains the questions and instructions for Component 2, the Mini Project revision. For details about the peer critiques, please see the Mini Project 2 assignment description on Canvas.

**Advanced CSS and layouts**

In class demonstrations, we used CSS for changing the appearance of content through simple things like color, fonts, and spacing. But CSS isn’t only useful for this alone! One of the more powerful uses of CSS is to create layouts for a web page, e.g., a multi-column layout in which we might place a visualization and explanatory text. In CSS, there are several ways we can control the layout of a page of content: **float**, **flexbox**, and **grid** are the most popular. They also differ dramatically in their implementation. The following guides provide more information about these approaches:  
  
A Complete Guide to Flexbox: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>  
A Complete Guide to Grid: <https://css-tricks.com/snippets/css/complete-guide-grid/>  
All About Floats: <https://css-tricks.com/all-about-floats/>

For Mini Project 2, you will implement one of these layout methods in your Mini Project 1 submission. Before that, however, please respond to the following questions.

1. (1.5 points) Skim through the 3 links provided above. **Briefly explain how each approach works (flexbox, grid, and float), explaining the key differences between the approaches.**
2. (0.5 point) For each of these approaches, you are provided a template that demonstrates the approach through HTML + CSS (e.g., for the CSS grid demonstration, the template folder is named *layout-template-grid*). Each template creates the same layout with a header, two-column middle section, and a footer, but through one of the three CSS layout approaches.  
     
   Select one of these layout approaches (flexbox, grid, or float) to study and implement in Mini Project 2. In the template folder of your selected approach, you will find two files: *index.html* and *style.css*. Open the template files in Visual Studio Code and open the *index.html* file in Chrome. In Visual Studio Code, examine the code for *index.html* and *style.css*: you will find a few elements in the <body> and also some CSS rules in the external stylesheet.  
     
   In the CSS, examine the following rule:  
     
   html, body {

margin: 0px;

padding: 0px;

}  
  
**What is the point of this CSS rule? Why are we setting margin and padding to 0px for the body and html elements? (What happens if we get rid of this rule? What does this tell you about the *user agent default style properties* for these elements?)**

1. (0.5 point) Also inside the CSS, you will find this rule:  
     
   \* {  
    box-sizing: border-box;  
   }  
     
   This rule features a property we haven’t seen before: **box-sizing.**  
     
   Read the W3Schools explanation page for this property:   
   <https://www.w3schools.com/css/css3_box-sizing.asp>.   
     
   **Then, explain what the box-sizing property does. Why is it helpful to use this style property? Why are we applying this property to all elements in the document, using the \* { } selector? (Can you think of a reason why we might want to *always* include this style property in a CSS stylesheet?)**
2. (0.5 point) Let’s look at one more thing inside the CSS file. Notice that we find the following rule:  
     
   #container {

width: 100%;

min-height: 100vh;

}  
  
Note that this selector, #container, matches the element in the HTML that has the ID value of “container.” To this element, we are applying the CSS style property named “min-height,” with a value of “100vh”. Pay specific attention to the “vh” in “100vh”. This is an accepted unit in CSS. **Using web resources (e.g., W3Schools), explain what this “vh” unit means and how it is used. What is it doing to our #container element here?** **(Why would this be useful?)**

1. (1 point) For your selected template, study the CSS and HTML files. **Then, provide a thorough explanation of how the layout you see displayed in Chrome is being rendered through the code in the template. Your explanation should discuss both the structure of the HTML elements inside the <body> themselves (which elements are in the HTML and how they are structurally related) *and* the style properties used in the CSS stylesheet.** *Note that the explanation will differ based on which layout approach you choose to investigate; not all layout approaches are the same in complexity or conceptual difficulty!*
2. (1 point) **Using your selected template and layout implementation (float, flex, or grid) as a starting point, transform your Mini Project 1 submission into a new layout of your choice.** You can use the provided template as a starting point for the layout and insert your Mini Project 1 content into that layout. For example, you could split the middle part of the document into 3 columns instead of two; you could get rid of the header and the footer section, and turn the layout into just a two-column layout with columns of equal width. It’s up to you – whatever you do, you will need to modify the CSS to make your layout work, and your modification must be substantively different from the template starting point. (In other words, don’t just change the relative sizes of the columns – that is not “substantively different.”)

1. (10 points) **Once you have your layout established, revise your Mini Project 1 submission, based on feedback received from Steven and from your peer review.** Your revision must be substantively different from your original Mini Project 1 submission – and note that “substantively different” will mean completely different things for every project! What I am looking for is demonstrated evidence of receiving, reflecting on, and integrating feedback received. *This does not mean you must adopt every piece of feedback you receive but rather that you should demonstrate that you’ve engaged with that feedback and responded to it in full through your design.*  
     
   An example of substantive revisions includes: changing a visualization in your project to a different chart type with a more appropriate encoding; redesigning all visualizations to make them more readable; revising your written narrative to integrate that narrative more cohesively with the visualizations; etc. If you simply take your Mini Project 1 content and insert it into your new CSS layout, without any modifications whatsoever, you will not receive credit for your submission.