This exam is open book, open note and open computer. You are not only allowed to, but encouraged to use online tools including codepen, jsbin, jsfiddle, formatters, validators, komodo editor and the development tools in a browser such as Chrome. Use of any chat capability or other means to communicate with another person is prohibited.

There are 4 questions and each one is worth 25 points. There is a blank page following every question to simplify your insertion of your answers.

You are to add your answers following each question. Your code is to be formatted properly so that it is easy to read. Do not worry about spaces between lines or Word formatting. We can look past any of those quirks, but we want the HTML, CSS and JavaScript to be formatted well and it all must be correct. Remember to save your work often.

When you complete the exam, you are to upload it to Sakai.

The recommended approach is to develop the answers with the tools that you have been using throughout the semester. You are likely to need to search for a property or value; you are not expected to have the properties and values memorized.

If you are having trouble with a small part of the requirements, complete what you can. We will be generous with partial credit.

You may ask an instructor for clarification of the questions, but not for help with answers.

By submitting this exam, you are pledging that you have neither given nor received unauthorized help.

Good luck!
1) Write the CSS to create a table that has the following format without adding any classes to the HTML elements of the table. Note that the borders below the “GDP in billions” line and the “% Population ages 65 and above” line are both double thick and there is 5px padding in all the cells.

```html
<table>
  <thead>
    <tr>
      <th>China</th>
      <th>1970</th>
      <th>1980</th>
      <th>1990</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>GDP in billions</td>
      <td>$118</td>
      <td>$216</td>
      <td>$526</td>
    </tr>
    <tr>
      <td>Population in millions</td>
      <td>818</td>
      <td>981</td>
      <td>1,135</td>
    </tr>
    <tr>
      <td>% Population ages 0-14</td>
      <td>40%</td>
      <td>35%</td>
      <td>29%</td>
    </tr>
    <tr>
      <td>% Population ages 15-64</td>
      <td>56%</td>
      <td>60%</td>
      <td>65%</td>
    </tr>
    <tr>
      <td>% Population ages 65 and above</td>
      <td>4%</td>
      <td>5%</td>
      <td>6%</td>
    </tr>
    <tr>
      <td>GDP per capita</td>
      <td>$144.59</td>
      <td>$220.44</td>
      <td>$463.08</td>
    </tr>
    <tr>
      <td>GDP per capita ages 15-64 only</td>
      <td>$259.23</td>
      <td>$370.34</td>
      <td>$713.27</td>
    </tr>
    <tr>
      <td>GDP per capita ages 15+ only</td>
      <td>$242.09</td>
      <td>$341.24</td>
      <td>$654.99</td>
    </tr>
  </tbody>
</table>
```
Sample Solution:

```css
table, tr, th, td {
    border: 1px solid black;
    border-collapse: collapse;
    padding: 5px;
}

th {
    background: black;
    color: white;
}

td:first-child {
    font-style: italic;
}

tr:nth-child(2), tr:nth-child(6) {
    border-bottom: 2px solid black;
}

td:nth-child(even) {
    background: lightblue;
}
```

Discussion:

We start by setting the base formatting for the whole table: collapsed borders of 1px solid black with padding of 5px everywhere. Remember that you have to set the borders correctly for the table, rows (tr) and data cells (th and td).

```css
table, tr, th, td {
    border: 1px solid black;
    border-collapse: collapse;
    padding: 5px;
}
```

We next want to change the top row to reverse the background and font color. This is easy to do because those are the only th cells in the table.

```css
th {
    background: black;
    color: white;
}
```

Next we will change the first column to italics. It is the first element of every row that is to be changed. We therefore want to change every td that is a first-child.

```css
td:first-child {
    font-style: italic;
}
```
Similarly, when we want to change the background color of the two columns, those are fixed positions for those td cells relative to the tr: specifically it is columns 2 and 4, which are all of the even columns so we can format them by identifying the even children

```html
td:nth-child(even) {  
  background: lightblue;  
}
```

The last requirement is to thicken 2 different row separators. We could actually do it with either the top or the bottom border. The description was given in terms of the bottom border, so we will do that. In order to identify which rows to change, we note their positions relative to the table tag. They are the 2\textsuperscript{nd} and 6\textsuperscript{th} rows in the table, so we can use nth-child to identify them.

```html
tr:nth-child(2), tr:nth-child(6) {  
  border-bottom: 2px solid black;  
}
```
2) Create the properly formatted HTML and CSS for a table inside of a figure and float it to the right of the following text, which is to be justified.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

The figure is to have a border around it and a caption of “Sample Table”. Both the table and caption are both to centered inside the figure with 5px of padding all around the figure. The table is to be a simple table of the form

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
C & D \\
\hline
\end{array}
\]

The format of the table is not important except that it needs an outside border. There is to be 20px between the paragraph and the figure and the top of the figure is line up with the top of the paragraph.

The final result should look like

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Sample Solution:

HTML:
<figure>
 <table>
  <tr>
   <td>A</td>
   <td>B</td>
  </tr>
  <tr>
   <td>C</td>
   <td>D</td>
  </tr>
 </table>
<figcaption>Sample Table</figcaption>
</figure>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>

CSS:
figure {
  float: right;
  border: 1px solid black;
  padding: 5px;
  margin-top: 0px;
}
table {
  border: 1px solid black;
  margin-left: auto;
  margin-right: auto;
}
p {
  text-align: justify;
  margin-right: 20px;
  margin-top: 0px;
}
Discussion:

There are numerous ways to get the same effect. This is just one.

We’ll start with the HTML.

We know that we need to put the figure in front of the paragraph in order to have the paragraph float around the figure. Remember that anything prior to the floated element is not floated.

```html
<figure>
</figure>
<p>
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
</p>

Next we create the content for the figure. It is going to have a table and a figcaption.

```html
<figure>
  <table>
  </table>
  <figcaption>Sample Table</figcaption>
</figure>
```

And the table is as simple as it can be.

```html
<table>
  <tr>
    <td>A</td>
    <td>B</td>
  </tr>
  <tr>
    <td>C</td>
    <td>D</td>
  </tr>
</table>
```

That is all there is for the HTML, so let’s move on to the CSS.

We’ll start with the basics of the figure. We are told that it is to float, have a padding of 5px and a border.

```css
figure {
  float: right;
  border: 1px solid black;
  padding: 5px;
}
```
And the paragraph is to justified and have 20px between it and the figure. Since the figure is to the right, this means it should be the right margin that we set.

```css
p {
    text-align: justify;
    margin-right: 20px;
}
```

We see that the paragraph and figure are not aligned properly. There are lots of ways to do this, but the simplest is to push them to the top of the block that they are in with a top margin of 0.

```css
figure {
    float: right;
    border: 1px solid black;
    padding: 5px;
    margin-top: 0px;
}
p {
    text-align: justify;
    margin-right: 20px;
    margin-top: 0px;
}
```

The only thing left to do is to format the table and the figcaption. You can get away without actually formatting the caption as it takes up the full space of the figure and there left, right and center all look alike. In general, it would be better to have the figcaption centered so that if the table grew, it would remain centered.

But we do need to assure that there is a border around the table and that it is centered. Note that we do not need to put a width or display value here, as a table by default is formatted as a block with its width computed.

```css
table {
    border: 1px solid black;
    margin-left: auto;
    margin-right: auto;
}
```
3) Create the HTML and CSS (no JavaScript) for a form that asks for a person’s name and an appointment time. It is to have two buttons, one for making an appointment and one for canceling an appointment. It is to return a confirmation message.
   a. The person’s name is to be a text box with a placeholder that reads “Your Name”. It is to have a label “Student:”
   b. The time is to be dropdown that has 4 options: 11 am, 1 pm, 2 pm and 3 pm. It is to have a label “Meeting:”
   c. The confirmation message is to have a label “Confirmation:” and the input box is to be invisible.
   d. When clicked, the two buttons are both to call the same function called appoint that takes three parameters: the first is to be either ‘make’ or ‘cancel’, the second is to be the name, and the third the time. It returns a confirmation message to be displayed. We will provide the appoint function when we want to test the code. You are NOT to write any JavaScript function for this assignment. You only need to write the correct onclick statement.
   e. The name, time and confirmation are all to begin on a new line

No other formatting is required. An example of such a form is

```
Student: Your Name:
Meeting:
Make Appt Cancel Appt
Confirmation:
```
Sample Solution:

HTML:

```html
<form name="appt">
  <label> Student: <input type="text" name="patient" placeholder="Your Name" /></label>
  <label>Meeting:</label>
    <input name="tod" list="times" />
    <datalist id="times">
      <option>11 am</option>
      <option>1 pm</option>
      <option>2 pm</option>
      <option>3 pm</option>
    </datalist>
  </label>
  <button type="button" class="make" onclick="appt.confirm.value = appoint('make',appt.patient.value,appt.tod.value);">Make Appt</button>
  <button type="button" class="cancel" onclick="appt.confirm.value = appoint('cancel',appt.patient.value,appt.tod.value);">Cancel Appt</button>
  <label>Confirmation: <input type="text" name="confirm" /></label>
</form>
```

CSS:

```css
form {
  border: 1px black solid;
  font-family: sans-serif;
  display: table;
  padding: 10px;
}
label {
  display: block;
  padding-top: 10px;
  padding-bottom: 10px;
}
input[name="confirm"] {
  background: transparent;
  border: none;
}
.make {
  float: left;
}
.cancel {
  float: right;
}
label:last-child {
  clear: both;
}
```

Discussion:

We’ll start with defining a form. The form is to have a text field, a datalist, two buttons, and another text field.
<form>
  <input type="text" />
  <input list="times" />
  <button type="button" onclick="appt.confirm.value = appoint();" >Make Appt</button>
  <button type="button" >Cancel Appt</button>
  <input type="text" />
</form>

Because we are in a form, the form and every input field is going to need a name.

<form name="appt">
  <input name="patient" type="text" />
  <input name="tod" list="times" />
  <button type="button" onclick="appt.confirm.value = appoint();" >Make Appt</button>
  <button type="button" >Cancel Appt</button>
  <input name="confirm" type="text" />
</form>

And for human consumption, we want to put labels on each of these elements. Note that the labeling was already part of the button definition.

<form name="appt">
  <label> Student: <input type="text" name="patient" placeholder="Your Name" /></label>
  <label>Meeting: <input name="tod" list="times" /></label>
  <button type="button" onclick="appt.confirm.value = appoint();" >Make Appt</button>
  <button type="button" >Cancel Appt</button>
  <label>Confirmation: <input type="text" name="confirm" /></label>
</form>

The instructions called for a placeholder in the name field.

<label> Student: <input type="text" name="patient" placeholder="Your Name" /></label>

Next, let’s define the datalist for the dropdown. We had given the list an identifier of “times”, so that is the id that we need to use on the datalist and it is added immediately following the input entry. Note that using the text as a value works in browsers and validation but not in jsfiddle, jsbin, or odepen.

<input name="tod" list="times" />
<datalist id="times">
  <option>11 am</option>
  <option>1 pm</option>
  <option>2 pm</option>
  <option>3 pm</option>
</datalist>

The last thing that we need to do is to define an appropriate onclick. The instructions tell us that the function name is to be “appoint” and that it is to return the string that is to go in the confirmation field. That means that the single statement in the onclick is to be

    appt.confirm.value = appoint();

appoint will return a string that will be placed in the confirm text field.

Next we need to figure out what parameters are needed to be passed to appoint. The instructions tell ups that they are to be an indication of the operation to be performed, the patient’s name and the
appointment time. The second and third parameters are clearly fields from the form. The patient’s name can be retrieved from

```
appt.patient.value
```

since the form name is “appt”, the field name is “patient” and we want to extract the value.

Similarly, the appointment time is

```
appt.patient.value
```

That leaves only the first field to figure out. In this case, we determine whether we are making or cancelling an appointment but which button is selected. Since there is a different onclick for each button, we can simply put a literal string in the function that indicates which operation to perform. We now have the complete onclick statement. For the make appt button it will be

```
appt.confirm.value = appoint('make',appt.patient.value,appt.tod.value);
```

and for the cancel appt button,

```
appt.confirm.value = appoint('cancel',appt.patient.value,appt.tod.value);
```

That completes all of the HTML for the form.

If you want to test the code, you need an appoint function. Here is the most basic one that will work:

```
function appoint (op, patient, appt_time) {
    return(patient+" "+op+"'s an appointment at "+appt_time);
}
```

For the CSS, we will begin with the formatting for the form itself. We will add a border, ask the browser to compute its width using display:table and put a bit of padding around the edges.

```
form {
    border: 1px black solid;
    display: table;
    padding: 10px;
}
```

Because we want each of the labeled fields to begin on a new line, we want them treated as a block instead of inline and therefore define them to be display: block. To make them look less squashed, we put padding both above and below them.

```
label {
    display: block;
    padding-top: 10px;
    padding-bottom: 10px;
}
```

While it was not required by the assignment, one way to format the buttons was to float them both to the sides with float left and right. I chose to give the buttons class names in order to make this formatting straightforward.

```
<button type="button" class="make" >
<button type="button" class="cancel" >
```
If we do this, however, we need to clear the float on the text field that follows them. There are a lot of ways to define that field. I chose to identify it as the label that was the last element in the form (last-child).

```css
label:last-child {
    clear: both;
}
```

The last thing that needs to be done is to remove the formatting on the confirmation text box. This could be done with a class, but in this case I chose to identify with the unique attribute that it already had, its name.

```html
input[name="confirm"] {
    background: transparent;
    border: none;
}
```
4) Write the function “whatyear” that can be called by this form to generate a statement of the form “He is a freshman.” or “She is a junior.” Specifically, you are to use the gender radio buttons to decide whether the statement should start with “he” or she” and you are to use the year number to identify the class. You can assume that only the values 1 through 4 will be entered. The function is to return the full string.

```html
<form name="college">
  <label> <input type="radio" name="gender" value="M" /> Male</label>
  <label> <input type="radio" name="gender" value="F" /> Female</label>
  <label>Year: <input type="number" name="year" /></label>
  <button type="button" onclick="college.answer.value = whatyear(college.gender.value, college.year.value);">Generate</button>
  <label>Answer: <input type="text" name="answer" /></label>
</form>
```

**Sample Solution:**

```javascript
function whatyear(gender, year) {
    var sentence; /* the sentence to be returned */
    // First half of the sentence depends on the gender
    if (gender == "M") {
        sentence = "He is a ";
    } else {
        sentence = "She is a ";
    }
    // second half of the sentence depends on the year
    if (year == 1) {
        sentence += "freshman.");
    } else if (year == 2) {
        sentence += "sophomore.");
    } else if (year == 3) {
        sentence += "junior");
    } else {
        sentence += "senior.");
    }
    return (sentence);
}
```
Discussion:

The form shows that the function to be written is to have two parameters, gender and year, and is to return a string.

```javascript
function whatyear(gender, year) {
    return (sentence);
}
```

We will declare the variable that we are going to use even though this is not strictly required. We do not bother giving it an initial value.

```javascript
var sentence;
```

We set up the first half of the sentence by looking at the first parameter. If it is an “M”, the sentence should begin “He is a”. The only other option is an “F”. Rather than checking it explicitly, we simply assume that it is an “F” if it is not an “M” and begin the sentence “She is a”. Note that we set the sentence variable to that value so it would overwrite anything that was already there. This is why we did not need to initialize it. The syntax of the if statement is that the condition is in parentheses (and equal is written as ==) and the statements that you want executed are in { }. Each statement ends with a ;. Also remember that the else clause never has a test.

```javascript
if (gender=="M") {
    sentence = "He is a ";
} else {
    sentence = "She is a ";
}
```

The second half of the sentence needs to make a 4-way decision based on the second parameter, year. That means that we should be using the else if construct. When we find a value that matches, we simply add the correct year to the sentence. In this case we do not want to lose the beginning of the sentence, so we concatenate the new string to the end of the current string, thus preserving the “He” versus “She”.

```javascript
if (year == 1) {
    sentence = sentence + "freshman.";
} else if (year == 2) {
    sentence = sentence + "sophomore.";
} else if (year == 3) {
    sentence = sentence + "junior";
} else {
    sentence = sentence + "senior.";
}
Because concatenating a string to the end of a string is such a common thing to do, there is a shorthand that you can use. The shorthand for

\[
\text{sentence} = \text{sentence} + \text{“sophomore.”};
\]
is

\[
\text{sentence} += \text{“sophomore.”};
\]

You never need to use this shorthand, but it can save you typing.

The sentence is now properly formed and we simply return it.

Since the exam, we studied arrays and it would be pretty easy to replace the second half of this function with an array. Let’s take a look at how we would do it.

First we declare an array that has the four class years defined.

\[
\text{var class_name} = [\text{“freshman”}, \text{“sophomore”}, \text{“junior”}, \text{“senior”}];
\]

I can now simply use the year parameter to choose between the names. Well, almost. The numbers are 1 to 4 and the indices are 0 to 3, so I need to subtract 1. I also still need to add that period. The 4 lines of part 2, can now be collapsed to

\[
\text{sentence} = \text{sentence} + \text{class_name}[\text{year}\, -\, 1] + \text{“.”};
\]

The return is the same.
Grades

<table>
<thead>
<tr>
<th>Letter Grade</th>
<th>Percentage</th>
<th>Range</th>
<th>Recorded Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>A+</td>
<td>4</td>
<td>95-100</td>
<td>100</td>
</tr>
<tr>
<td>A</td>
<td>8</td>
<td>81-94</td>
<td>95</td>
</tr>
<tr>
<td>A-</td>
<td>6</td>
<td>77-80</td>
<td>92</td>
</tr>
<tr>
<td>B+</td>
<td>11</td>
<td>71-76</td>
<td>88</td>
</tr>
<tr>
<td>B</td>
<td>9</td>
<td>66-70</td>
<td>85</td>
</tr>
<tr>
<td>B-</td>
<td>13</td>
<td>62-65</td>
<td>82</td>
</tr>
<tr>
<td>C+</td>
<td>6</td>
<td>60-61</td>
<td>78</td>
</tr>
<tr>
<td>C</td>
<td>15</td>
<td>54-59</td>
<td>75</td>
</tr>
<tr>
<td>C-</td>
<td>8</td>
<td>46-53</td>
<td>72</td>
</tr>
<tr>
<td>D+</td>
<td>7</td>
<td>43-45</td>
<td>68</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
<td>29-42</td>
<td>65</td>
</tr>
<tr>
<td>D-</td>
<td>4</td>
<td>1-28</td>
<td>62</td>
</tr>
</tbody>
</table>