Today we will create a web page that can convert values from decimal to binary and hexadecimal. We will explore the use of JavaScript functions as well as revisit HTML and CSS. We will restrict the values being converted to be positive numbers only.

A. Setting up the HTML
We’re going to set up our basic page HTML. The tags should be familiar to you, as you have used them in previous labs and homework assignments. If you are having trouble, revisiting these assignments may be useful.

1. Create a new “Lab8” folder inside your CSE3 folder. With Notepad++, open and save an HTML file with the filename DecimalConverter.html.
2. Set up the HTML skeleton as we have done in previous labs and homework assignments.
3. Make the title of the page “Decimal Converter”.
4. Center the code that will be in the body by putting in <center> tags.
5. Create a heading with the <h1> tag with the words “Decimal to Binary and Hexadecimal Converter”.
6. Underneath the heading, create a horizontal rule with the <hr> tag.
7. Underneath the horizontal rule, use the <h2> tag to create a secondary heading with the description: “Convert a decimal value to binary and hexadecimal!”’. Up until this point, your HTML should look something like this:

```
<html>
  <head>
    <title>Decimal Converter</title>
  </head>
  <body>
    <center>
      <h1>Decimal to Binary and Hexadecimal Converter</h1>
      <hr />
      <h2>Convert a decimal value to binary and hexadecimal!</h2>
    </center>
  </body>
</html>
```

8. Our page is going to have a table that eventually looks like this:

```
<table>
<thead>
<tr>
<th>New Value</th>
<th>Decimal</th>
<th>Binary</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert Value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

10. To create the first row, we will use the <th> tags, with the titles shown in the image above.
10. To create the data columns of the table, we will use the `<td>` tags.
11. To create the buttons, we will use the `<button>` tags. The button tag has a special attribute called `onclick`. We will leave this blank for now, but we will use it to add functionality later.
12. For the remaining blank cells, we will put a set of `<p>` tags, each with `id` attributes. There should be one for `decimalId`, `binaryId`, and `hexId`. This will be explained later.

Your table in HTML should now look something like this:

```html
<table border="2">
  <tr>
    <th><button onclick = "">New Value</button></th>
    <th>Decimal</th>
    <th>Binary</th>
    <th>Hexadecimal</th>
  </tr>
  <tr>
    <td><button onclick = "">Convert Value</button></td>
    <td>
      <p id="decimalId"></p>
    </td>
    <td>
      <p id="binaryId"></p>
    </td>
    <td>
      <p id="hexId"></p>
    </td>
  </tr>
</table>
```

B. Formatting with CSS

As we did before, we will use CSS to efficiently format and style our web page.

1. In between the `<head>` tags, create a set of `<style>` tags as before:

   ```html
   <style type="text/css"></style>
   ```

2. We want to use CSS to give our page background a color. Set the `body`'s `background-color` to a color of your choice.
3. As well as our table, set the `table`'s `background-color` to a color of your choice.
4. We will modify the `hr` rule on the page to be 50% width and a 4px height.
5. We will modify the `th` and `td` to be 100px width, 50px height. For the `td`, we will use `text-align` to align the inside to `center`.
6. Lastly, we will make `button` the same width and height as the table cells by using `inherit` on the `width` and `height`. 
7. The finished CSS should look something like this:

```html
<style type="text/css">
body {
    background-color: #99999;
}
table {
    background-color: #12C984;
}
hr {
    width: 50%;
    height: 4px;
}
th {
    width: 100px;
    height: 50px;
}
td {
    width: 100px;
    height: 50px;
    text-align: center;
}
button {
    width: inherit;
    height: inherit;
}
</style>
```

C. Our JavaScript functions

Functions are used in many programming languages to encapsulate a set of instructions to be executed. Oftentimes this is code that is used multiple times with varying input. This saves us the trouble of the same set of instructions copied in different places in the program. Instead, we simply perform a function call and the code within the function is executed. In our binary converter, we will implement four functions: one to interact with the user and take in a value to be converted which we will call `get_input()`, one to do conversion from decimal to binary called `convertToBinary()`, one to do conversion from decimal to hex called `convertToHex`, and one to call both functions called `convert()`.

1. Put `<script>` tags within our `<head>` tags to denote we are writing JavaScript code. All our JavaScript code will reside in here: `<script type="text/javascript"> </script>`

2. First let’s declare a variable called `userInput` that we will use to keep track of the user input. This will be the value we convert.

3. Next, we must declare our functions. The syntax looks similar to that of the loops; the code within the curly braces are executed each time the function is called. Functions also have a set of parenthesis. This is used to pass values to the function, but we won’t be using them in this case. Below is the `get_input()` function. Write the three other functions yourself.

```javascript
function get_input() {
    // code that executes when get_input is called
}
```
4. For the function `get_input()`, we will prompt the user for a value to convert:
```javascript
userInput = prompt("Please enter a decimal value to convert.");
```

5. JavaScript has a useful command that allows us to modify existing HTML code dynamically. Put this as the next line of code.
```javascript
document.getElementById("decimalId").innerHTML = userInput;
```
In plain English, the command is saying:
1. Look through this document (html page).
2. Find the element (tag) in this document by its id attribute (getElementById).
3. Get the HTML code within this element's tags (innerHTML).
4. Replace the HTML code within the tag with a new value (the equal sign and then new value).

6. Let's take a look back to the table HTML and find the `<p>` tags that we defined earlier. The last line of code we wrote will insert the value stored in `userInput` into where it says “userInput goes here”. That line of code says to look through the document and get the tag with the id called `decimalId`, which is the first set of `<p>` tags. Now `innerHTML` will get the text in between the opening and closing `<p>` tag. We then set the value of `userInput` as the text in between the opening and closing tag.
```html
<td>
  <p id="decimalId">userInput goes here</p>
</td>
<td>
  <p id="binaryId"></p>
</td>
<td>
  <p id="hexId"></p>
</td>
```

7. That’s it for the `get_input()` function. Now let’s go back to the HTML code for our “New Value” button. In between the quotations for the `onclick` attribute, put `get_input()`. This is a function call, and it will call the `get_input()` function to execute the code that the function specifies. To make any function call with no parameters, you will simply need to type the function name.
```html
<th><button onclick = "get_input()">New Value</button></th>
```
Now if you run the code and click the “New Value” button, you should see a prompt, and once you enter a value, you should see it reflected in the cell under “Decimal”.

8. In the `convert()` function, we will call the other two functions that we will write the code for in a moment. Call `convertToBinary()` and `convertToHex()`. Hint: see last step for how to make a function call.

9. Let’s go back to our HTML code once more and for the `onclick` to the “Convert” button, add the function call to `convert()`.

10. Let’s start defining `convertToBinary()`. In this function, we will convert the decimal number to a binary number using successive division, which will be continually dividing the number by the base until the quotient equals zero. The remainders from each of these divisions will compose of the answer.

11. First, create a new variable called `input` and make it equal to `userInput`. We do this because we don’t want to overwrite the `userInput` variable later on. Then create another variable called `binary`. This will be the variable that will hold our binary value.
12. Use a **while loop** with the condition that `input > 0`. Inside the loop, get the remainder of `input` divided by 2 by doing a **modulus** operation, and assign it to a new variable called `remainder`. This is the least significant bit, which is the last bit in our binary value. Concatenate it to the front of our `binary` variable with `binary = remainder + binary`. Finally, update our `input` value by dividing it by 2 and rounding it down to the closest integer value with `input = Math.floor(input/2)`. Your **while loop** should look something like this:

```javascript
while(input > 0) {
    var remainder = input % 2;
    binary = remainder + binary;
    input = Math.floor(input / 2);
}
```

13. After the loop, we will need to display the binary value that we calculated. We will input that in the cell under “Binary” using the `document.getElementById()` function to set the innerHTML of the tag with the id `binaryId` to be equal to `binary`:

```javascript
document.getElementById("binaryId").innerHTML = binary;
```

14. If you run your HTML page now, you should be able to convert a decimal number into binary.

15. Now all that is left to do is to finish the function `convertToHex()` to be able to convert our decimal value to hex. This will be similar to the `convertToBinary()` function with the exception of the base being 16 (as it is hex), and when we encounter a remainder of 10-15, we will need to convert those to the appropriate letters of A-F.

16. In our `convertToHex()` function, create two variables, one called `input` and another called `hex`, which will hold our final hexadecimal value.

17. Again, create a **while loop** with the condition of `input > 0`. Calculate the remainder of `input` with `input % 16` and assign it to a new variable called `remainder`. Now we will need to check if the remainder is 10, 11, 12, 13, 14, or 15. If it is, change the remainder to the appropriate hex value of A, B, C, D, E, or F using **if** and **else if** statements. Your **if** statements should look like this:

```javascript
if(remainder == 10) {
    remainder = "A";
}
else if(remainder == 11) {
    remainder = "B";
}
else if(remainder == 12) {
    remainder = "C";
}
else if(remainder == 13) {
    remainder = "D";
}
else if(remainder == 14) {
    remainder = "E";
}
else if(remainder == 15) {
    remainder = "F";
}
```
18. To finish off the while loop, like we did before, concatenate the remainder to the front of our hex variable with \( \text{hex} = \text{remainder} + \text{hex} \). Finally, update our input value by dividing it by 16 and rounding it down to the closest integer value with \( \text{input} = \text{Math.floor(input/16)} \).

19. After the loop, we will need to display the hex value that we calculated. We will input that in the cell under “Hex” using the `document.getElementById()` function to set the innerHTML of the tag with the id hexId to be equal to hex.

\[
\text{document.getElementById("hexId").innerHTML = hex;}
\]

20. You should now be able to successfully convert decimal values into binary and hexadecimal!

**D. Putting it all online**

1. Modify your CSE3Page.html to include a link to your DecimalConverter.html.
2. Use Secure File Transfer Client to update your CSE folder online.

**Check-off:**

- Updated CSE3Page.html online
- Working binary converter (DecimalConverter.html)