Perla El Khoury:

Assignment 2:

JavaScript Array Methods:

slice(): Returns a shallow copy of a portion of an array into a new array object.

splice(): Changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.

concat(): Merges two or more arrays and returns a new array.

filter(): Creates a new array with all elements that pass the test implemented by the provided function.

```
function removeMiddleElement(arr) {
    if (arr.length <= 2) {
      return arr; // If the array has 0, 1, or 2 elements, return the
original array
    }

    const middleIndex = Math.floor(arr.length / 2);
    const newArray = [...arr.slice(0, middleIndex),
...arr.slice(middleIndex + 1)];
    return newArray;
  }
```

```
const originalArray = [1, 2, 3, 4, 5];
const newArray = removeMiddleElement(originalArray);
console.log(originalArray); // [1, 2, 3, 4, 5] (original array is not
modified)
console.log(newArray); // [1, 2, 4, 5]
```

1. Create a function that removes the middle element from an array:

```
function removeMiddle(arr) {
  if (arr.length <= 2) {
    return arr;
  }
  const middleIndex = Math.floor(arr.length / 2);
 return [...arr.slice(0, middleIndex), ...arr.slice(middleIndex +
1)];
}
```

2. Create a function called "countTrue" that returns the number of "true" values in an array:

```
function countTrue(arr) {
  return arr.filter(Boolean).length;
}
```

3. Create a function called "toArray" that converts an object into an array of key-value pairs:

```
function toArray(obj) {
  return Object.entries(obj);
}
```

4. Create a function called "luckyNumber" that checks if the digit 7 appears in an array of numbers:

```
function luckyNumber(arr) {
  return arr.some(num => num.toString().includes('7')) ? "LUCKY!" :
"there is no 7 in the array 😣";
}
```

5. Create a function called "oddishOrEvenish" that determines whether the sum of a number's digits is odd or even:

```
function oddishOrEvenish(num) {
const digitSum = num.toString().split('').reduce((sum, digit) => sum
+ parseInt(digit), 0);
return digitSum % 2 === 0 ? 'Evenish' : 'Oddish';
}
```

6. Create a function called "reverseOdd" that reverses all the words with odd length in a sentence:

```
function reverseOdd(sentence) {
return sentence.split(' ')
 .map(word => word.length % 2 === 1 ?
word.split('').reverse().join('') : word)
 .join(' ');
}
```

7. Create a function called "getHashTags" that retrieves the top 3 longest words in a newspaper headline and transforms them into hashtags:

```
function getHashTags(headline) {
 const words = headline.replace(/[^\w\s]/gi, '').split(' ');
 words.sort((a, b) => b.length - a.length);
return words.slice(0, 3).map(word => `#${word.toLowerCase()}`);
}
```

8. Research JSON files, how to read and write them in JavaScript:
   JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for humans to read and write, and easy for machines to parse and generate.
   To read a JSON file in JavaScript, you can use the built-in JSON.parse() function:

```
// Reading a JSON file
fetch('data.json')
  .then(response => response.json())
  .then(data => console.log(data));

To write a JSON file in JavaScript, you can use the built-in
JSON.stringify() function:
// Writing a JSON file
const data = { name: 'John', age: 30 };
const jsonData = JSON.stringify(data);
// Save the jsonData to a file
```

9. Implement Breadth-First Search (BFS) and Depth-First Search (DFS) for an HTML document using JavaScript:

```
// BFS implementation
function bfs(document) {
  const queue = [document.body];
  const visited = new Set();

  while (queue.length > 0) {
    const node = queue.shift();
    if (!visited.has(node)) {
      visited.add(node);
      console.log(node.tagName);
      for (const child of node.children) {
        queue.push(child);
      }
    }
  }
}

// DFS implementation
function dfs(document) {
  const stack = [document.body];
  const visited = new Set();

  while (stack.length > 0) {
    const node = stack.pop();
```

```
    if (!visited.has(node)) {
      visited.add(node);
      console.log(node.tagName);
      for (const child of node.children) {
        stack.push(child);
      }
    }
  }
}
```

10. Create a HTML form, that takes as a user input a number between 1 and 5 ○ Add a buton called "Build" that draws "stairs":

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Stair Builder</title>
    <body>
        <h1>Stair Builder</h1>
        <form id="stairForm">
            <label for="steps">Number of steps (1-5):</label>
            <input type="number" id="steps" name="steps" min="1"
max="5" required>
            <button type="button" id="buildBtn">Build</button>
            <button type="button" id="appendBtn"
class="hidden">Append</button>
            <button type="button" id="destroyBtn"
class="hidden">Destroy</button>
        </form>
        <div id="stairContainer" class="stairs"></div>
 </head>
<body>
    <style>
        .stairs {
            display: flex;
            flex-direction: column;
            align-items: flex-start;
        }
        .step {
            background-color: grey;
            height: 20px;
            margin: 2px 0;
        }
```

```
 .step.appended {
            background-color: yellow;
        }
        .hidden {
            display: none;
        }
    </style>


    <script>
        const buildBtn = document.getElementById('buildBtn');
        const appendBtn = document.getElementById('appendBtn');
        const destroyBtn = document.getElementById('destroyBtn');
        const stairContainer =
document.getElementById('stairContainer');
        const stepsInput = document.getElementById('steps');

        buildBtn.addEventListener('click', () => {
            if (stairContainer.children.length === 0) {
                const steps = parseInt(stepsInput.value);
                if (steps >= 1 && steps <= 5) {
                    createStairs(steps);
                    buildBtn.classList.add('hidden');
                    appendBtn.classList.remove('hidden');
                    destroyBtn.classList.remove('hidden');
                }
            }
        });

        appendBtn.addEventListener('click', () => {
            const steps = parseInt(stepsInput.value);
            if (steps >= 1 && steps <= 5) {
                createStairs(steps, true);
            }
        });

        destroyBtn.addEventListener('click', () => {
            stairContainer.innerHTML = '';
            buildBtn.classList.remove('hidden');
            appendBtn.classList.add('hidden');
            destroyBtn.classList.add('hidden');
        });

        function createStairs(num, append = false) {
 const existingSteps = stairContainer.children.length;
```

```
            for (let i = 0; i < num; i++) {
                const step = document.createElement('div');
                step.classList.add('step');
                if (append) {
                    step.classList.add('appended');
                }
                step.style.width = `${20 * (existingSteps + i
+1)}px`; // Incremental width for staircase effect
                stairContainer.appendChild(step);
            }
        }
    </script>
</body>
</html>
```