- Write a function that takes an array of numbers and returns the largest number in that array.
  - Examples:

    ```
    findLargest([1, 3, 5, 7, 9]); // Output: 9
    findLargest([-10, -5, 0, 5, 10]); // Output: 10
    findLargest([34, 23, 12, 56, 89]); // Output: 89
    ```

- Create a function that takes a string and counts how many vowels (a, e, i, o, u) are in the string.
  - Examples:

    ```
    countVowels("Hello World"); // Output: 3
    countVowels("JavaScript"); // Output: 3
    countVowels("OpenAI"); // Output: 3
    ```

- Write a function that takes an array of numbers and returns the total sum of those numbers.
  - Examples:

    ```
    sumArray([1, 2, 3, 4, 5]); // Output: 15
    sumArray([-1, -2, -3, -4, -5]); // Output: -15
    sumArray([10, 20, 30]); // Output: 60
    ```

- Create a function that takes a string and returns a new string that is the reverse of the input string.
  - Examples:

    ```
    reverseString("JavaScript"); // Output: "tpircSavaJ"
    reverseString("hello"); // Output: "olleh"
    reverseString("OpenAI"); // Output: "IAnepO"
    ```

- Write a function that determines whether a given number is a prime number (a number greater than 1 that has no positive divisors other than 1 and itself).
  - Examples:

    ```
    isPrime(7); // Output: true
    isPrime(10); // Output: false
    isPrime(13); // Output: true
    ```

- Write a function that calculates the average (mean) of all numbers in an array.

  - Examples:

    ```
    findAverage([10, 20, 30, 40, 50]); // Output: 30
    findAverage([1, 2, 3, 4, 5]); // Output: 3
    findAverage([100, 200, 300]); // Output: 200
    ```

- Write a function that generates a Fibonacci sequence where each number is less than or equal to a given maximum number.

  - Examples:

    ```
    generateFibonacci(15); // Output: [0, 1, 1, 2, 3, 5, 8, 13]
    generateFibonacci(10); // Output: [0, 1, 1, 2, 3, 5, 8]
    generateFibonacci(50); // Output: [0, 1, 1, 2, 3, 5, 8, 13, 21,
    34]
    ```

- Write a function that identifies the longest word in a given sentence.

  - Examples:

    ```
    findLongestWord("The quick brown fox jumps over the lazy dog");
    // Output: "jumps"
    findLongestWord("JavaScript is awesome"); // Output: "JavaScript"
    findLongestWord("Hello world!"); // Output: "Hello"
    ```

- Create a function that checks if a string reads the same forwards and backwards, ignoring spaces, punctuation, and case.

  - Examples:

    ```
    isPalindrome("A man, a plan, a canal, Panama"); // Output: true
    isPalindrome("Racecar"); // Output: true
    isPalindrome("Hello"); // Output: false
    ```

- Write a function that converts a string so that each word starts with an uppercase letter and the rest are lowercase.

  - Examples:

    ```
    toTitleCase("hello world"); // Output: "Hello World"
    toTitleCase("javascript is fun"); // Output: "JavaScript Is Fun"
    toTitleCase("OPENAI"); // Output: "Openai"
    ```

- Write a function that finds the item that appears the most often in an array.

  - Examples:

    ```
    mostFrequentItem([1, 2, 2, 3, 3, 3, 4, 4, 4, 4]); // Output: 4
    mostFrequentItem([
      "apple",
      "banana",
      "apple",
      "orange",
      "banana",
      "banana",
    ]); // Output: 'banana'
    mostFrequentItem([10, 20, 10, 30, 20, 10]); // Output: 10
    ```

- Write a function that sorts an array of objects based on a specific property, such as age or name.

  - Examples:

    ```
    sortObjects(
      [
        { name: "Alice", age: 25 },
        { name: "Bob", age: 30 },
        { name: "Charlie", age: 20 },
      ],
      "age"
    );
    // Output: [{ name: "Charlie", age: 20 }, { name: "Alice", age: 25 }, { name: "Bob", age: 30 }]

    sortObjects(
      [
        { name: "Zara", salary: 50000 },
        { name: "John", salary: 70000 },
        { name: "Doe", salary: 60000 },
      ],
      "salary"
    );
    // Output: [{ name: "Zara", salary: 50000 }, { name: "Doe", salary: 60000 }, { name: "John", salary: 70000 }]

    sortObjects(
      [
        { name: "Alice", score: 85 },
        { name: "Bob", score: 90 },
        { name: "Charlie", score: 80 },
      ],
      "score"
    );
    // Output: [{ name: "Charlie", score: 80 }, { name: "Alice", score: 85 }, { name: "Bob", score: 90 }]
    ```

- Write a function that returns an array of elements that are present in both of the given arrays.

  - Examples:

    ```
    findIntersection([1, 2, 3, 4], [3, 4, 5, 6]); // Output: [3, 4]
    findIntersection(["a", "b", "c"], ["b", "c", "d"]); // Output:
    ['b', 'c']
    findIntersection([7, 8, 9], [9, 10, 11]); // Output: [9]
    ```

- Write a function that flattens a nested array (an array within an array) into a single array.

  - Examples:

    ```
    flattenArray([[1, 2, [3]], [4, 5], [6]]); // Output: [1, 2, 3, 4,
    5, 6]
    flattenArray([1, [2, [3, [4]]], 5]); // Output: [1, 2, 3, 4, 5]
    flattenArray([["a", "b"], ["c", ["d", "e"]], "f"]); // Output:
    ['a', 'b', 'c', 'd', 'e', 'f']
    ```