**Task 01: Count 'e' Characters in a String**

Create a JavaScript script that prompts the user to enter a string and counts the number of occurrences of the character 'e', regardless of case (i.e., both 'e' and 'E' should be counted).

**Requirements:**

1. **User Input**
   * Prompt the user to enter a string using a browser prompt dialog.
   * Ensure the script handles the case where the user cancels the prompt or provides no input.
2. **Processing**
   * Convert the entire string to lowercase to handle both 'e' and 'E' uniformly.
   * Count each occurrence of the character 'e' in the string.
   * Handle strings of varying lengths and cases.
3. **Output**
   * Display the total count of 'e' characters in the string using an alert dialog.
   * Provide a message indicating the result clearly.

**Task 02: Array Operations**

Create a JavaScript script that allows the user to input three numeric values, stores these values in an array, and then performs and displays the results of three mathematical operations: addition, multiplication, and division. Format the output for clarity.

**Requirements:**

1. **User Input**
   * Prompt the user to enter three numeric values.
   * Validate that each input is a valid number.
2. **Array Storage**
   * Store the three validated numeric values in an array.
3. **Mathematical Operations**
   * **Addition**: Compute the sum of all three values.
   * **Multiplication**: Compute the product of all three values.
   * **Division**: Compute the result of dividing the first value by the second, then by the third (handle division by zero).
4. **Output**
   * Display the results of each operation in a formatted and user-friendly manner.

**Hints for Implementation:**

* **Input Handling**:  
  Use parseFloat() to convert input strings to numbers.  
  Validate inputs using isNaN() to ensure they are numeric.
* **Array Operations**:  
  Store the values in an array and use array indexing to access them.  
  Perform addition, multiplication, and division with proper error handling.
* **Output Formatting**:  
  Use toFixed(2) to format numerical results for consistent display.

**Task 03: Array Sorting**

Create a JavaScript script that allows the user to input five numeric values, stores these values in an array, sorts the array in both ascending and descending orders, and displays the sorted arrays.

**Requirements:**

1. **User Input**
   * Prompt the user to enter five numeric values.
   * Validate that each input is a valid number.
2. **Array Storage**
   * Store the five validated numeric values in an array.
3. **Sorting**
   * **Ascending Order**: Sort the array from smallest to largest.
   * **Descending Order**: Sort the array from largest to smallest.
4. **Display Output**
   * Display the sorted arrays clearly to the user.

**Example Scenario:**

* **User Input:**  
  Value 1: 8  
  Value 2: 3  
  Value 3: 5  
  Value 4: 12  
  Value 5: 7
* **Array Storage:**  
  [8, 3, 5, 12, 7]
* **Sorting Results:**  
  Ascending Order: [3, 5, 7, 8, 12]  
  Descending Order: [12, 8, 7, 5, 3]
* **Formatted Output:**  
  Ascending Order: "Sorted in ascending order: 3, 5, 7, 8, 12"  
  Descending Order: "Sorted in descending order: 12, 8, 7, 5, 3"

### 🧪 4. Regular Expression Object — User Information Collector and Validator

#### 🔹 **Objective:**

Create a JavaScript script that collects personal user information, validates it using **regular expressions**, and displays a welcome message upon successful validation.

#### 📋 **Requirements:**

##### 🧍 1. User Input Fields:

* **Name**: Must consist of **alphabetic characters only** (no numbers or special characters).
* **Mobile Number**: Must be **exactly 11 digits** and start with **"010"**, **"011"**, or **"012"**.
* **Email**: Must follow a **valid email format** using a regular expression.

##### ✅ 2. Validation Rules:

* Validate **each field** using appropriate regular expressions.
* If any input is **invalid**, notify the user and **prompt again**.
* Only proceed if **all inputs are valid**.

##### 🎉 3. Output:

If validation passes, show a **welcome message** like:

MSG: Welcome, JohnDoe! ,your mobile number is 01012345678, and your email is johndoe@example.com.

If **any validation fails**, display a **clear error message** for that field and prompt the user to **re-enter** it.

#### 💡 Example Scenario:

Copy code

Name: JohnDoe

Mobile Number: 01012345678

Email: johndoe@example.com

If all validations are correct → ✅ Show welcome message.  
If not → ❌ Prompt the user to correct only the invalid input.

### 🧮 5. Validate Arguments and Calculate Total

#### 🔹 **Objective:**

Write a function that accepts **any number of arguments**, and performs type-checking and total calculation.

#### 📋 **Requirements:**

* Use a **function statement**.
* If **any argument is a string**, throw a TypeError.
* If all are valid numbers, **return their total** using the arguments object.

#### 💡 Example:

calculateTotal(5, 10, 20); // ✅ returns 35

calculateTotal(2, "3", 5); // ❌ alert: String values are not allowed