**Exercise 1: Validate a Phone Number**

**Problem**: Write a regular expression to validate US phone numbers in the format (123) 456-7890 or 123-456-7890.

**Exercise 2: Validate an Email Address**

**Problem**: Write a regular expression to validate email addresses with the format username@domain.com. Ensure that the domain part contains at least one period.

**Exercise 3: Extract Dates from a String**

**Problem**: Write a regular expression to extract dates in the format DD/MM/YYYY from a string.

**Exercise 4: Validate a Strong Password**

**Problem**: Write a regular expression to validate a password that is at least 8 characters long, contains at least one uppercase letter, one lowercase letter, one digit, and one special character.

**Exercise 5: Validate a Hexadecimal Color Code**

**Problem**: Write a regular expression to validate hexadecimal color codes. These can be in the format #RRGGBB or #RGB.

**Exercise 6: Extract All URLs from a Text**

**Problem**: Write a regular expression to extract all URLs from a text. URLs should start with http:// or https://.

**Exercise 7: Match a Valid IPv4 Address**

**Problem**: Write a regular expression to match valid IPv4 addresses.

**Exercise 8: Match a Valid Roman Numeral**

**Problem**: Write a regular expression to match Roman numerals from I to MMXXI.

**Exercise 9: Validate a US ZIP+4 Code**

**Problem**: Write a regular expression to validate US ZIP+4 codes (e.g., 12345-6789).

**Exercise 10: Match Words with Apostrophes**

**Problem**: Write a regular expression to match words that may contain apostrophes (e.g., "don't").

**Exercise 11: Validate a Username**

**Problem**: Create a form where the username must be 5-15 characters long and contain only letters, numbers, and underscores. Display an error message in red if the username is invalid.

**Exercise 12: Validate an Email Address**

**Problem**: Create a form to validate email addresses. Display an error message in red if the email is invalid.

**Exercise13: Validate a Phone Number**

**Problem**: Create a form to validate US phone numbers in the format (123) 456-7890 or 123-456-7890. Display an error message in red if the phone number is invalid.

**Exercise 14: Validate a Date of Birth**

**Problem**: Create a form to validate a date of birth in the format DD/MM/YYYY. Display an error message in red if the date is invalid.

**Exercise 15: Validate a Credit Card Number**

**Problem**: Create a form to validate credit card numbers with spaces (e.g., 1234 5678 9012 3456). Display an error message in red if the credit card number is invalid.

**Exercise 16:**

**Form Validation Exercise: Product Insertion Form**

**Objective**: Create a product insertion form with validation using HTML, CSS, and JavaScript. Implement regular expressions to validate input fields and display error messages in red if the validation fails.

**Requirements**:

1. **HTML Form**:
   * Include fields for the following product details:
     + Product Name (2-50 characters, letters and numbers only)
     + Price (positive number, up to 2 decimal places)
     + Category (dropdown with options: Electronics, Clothing, Home, Toys)
     + Stock Quantity (positive integer)
     + Brand (2-30 characters, letters and numbers only)
     + Model (2-30 characters, letters and numbers only)
     + Weight (positive number, up to 2 decimal places)
     + Color (2-30 characters, letters only)
     + Size (2-20 characters, letters and numbers only)
     + Description (5-500 characters)
2. **CSS**:
   * Style the form to be large, with a width of at least 800px, centered on the page.
   * Apply padding, border, and shadow to the form container.
   * Ensure all input fields, select dropdowns, and the textarea are styled consistently.
   * Error messages should be displayed in red below each field if the input does not meet the validation criteria.
   * Style the submit button to be large and provide a hover effect.
3. **JavaScript**:
   * Implement form validation using regular expressions.
   * Display error messages in red if the input does not match the expected format.
   * Prevent form submission if any field is invalid.

**Bonus**:

* Implement additional validation logic to handle edge cases and improve user experience.
* Ensure that the error messages are cleared when the input becomes valid.