# React Hooks Assignment

## 1. Theme Switching with Context

Assignment:

Build a React application that allows users to switch between light and dark themes. Use useContext to manage the theme state globally. Provide a toggle button that changes the theme throughout the entire application when clicked.

Expansion:

Create a React application with a theme context provider that wraps the entire app. Inside this provider, define a state for the theme (e.g., light or dark) and a function to toggle between them. Use useContext in child components to access and update the current theme. Implement a toggle button in a top-level component that dispatches the theme change action when clicked.

## 2. Form Validation with useRef

Assignment:

Create a form with input fields for name, email, and password. Implement form validation using useRef to access and manipulate the form elements directly. Display error messages for invalid inputs in real-time without triggering unnecessary re-renders.

Expansion:

Build a form component with input fields and corresponding useRef references. Attach validation functions to the onBlur or onChange events of each input, updating useRef values for error messages. Display error messages dynamically based on the validation results without causing re-renders for every keystroke. Ensure the form submission is only allowed when all inputs are valid.

3. Performance Optimization with useCallback

Assignment:

Develop a list component that renders a dynamic number of items (Public API). Use useCallback to memoize the callback function for item rendering, ensuring that it doesn't recreate on each render.

Expansion:

Create a list component that receives an array of items as props. Use useCallback to memoize the rendering function for each item, preventing unnecessary re-renders when the list changes.

4. Memoization for Expensive Calculations

Assignment:

Build a calculator application with React that performs complex calculations. Utilize useMemo to memoize the result of expensive calculations, preventing unnecessary recalculations when the input values haven't changed. Display the memoized result on the UI for verification.

Expansion:

Develop a calculator component with input fields for numbers and buttons for operations. Implement a calculation function that performs complex operations. Use useMemo to memoize the result of the calculation based on input values. Display the memoized result on the UI, and demonstrate how it remains unchanged during subsequent input changes.

5. Fetching Data with useEffect

Assignment:

Build a React application that fetches and displays data from an external API. Use the useEffect hook to make the API call when the component mounts. Display the fetched data in a user-friendly format on the UI. Implement loading and error states for a better user experience.

Expansion:

Choose a public API (e.g., JSONPlaceholder, OpenWeatherMap).

Use useEffect to fetch data from the API when the component mounts.

Display the fetched data on the UI and include loading and error states.