React Practice Questions:

1. What is React, and what are its key features?

2. What is the virtual DOM, and how does it improve performance?

3. What are React hooks? Name some commonly used hooks.

4. What is JSX, and how does it relate to React?

5. What is state in React, and how do you manage it in components?

6. What is the difference between a functional component and a class component?

7. What is state in React, and how do you manage it in components?

8. What are props in React, and how are they different from state?

9. Explain the React component lifecycle.

10. What is the purpose of keys in React lists, and why are they important?

11. What is Redux, and how does it work with React?

12. Create a functional component that fetches data from an API and displays it. Use the useEffect hook to make the API request when the component mounts.

13. Create a custom hook called useLocalStorage that lets you store and retrieve a value from localStorage. Use it in a component to save the name and display it.

14. Use the React Context API to share a value (e.g., theme) between multiple components without passing props manually through every level of the component tree.

15. Create an error boundary component that catches JavaScript errors in its child components and displays a fallback UI.

16. Implement React.memo to optimize a component that re-renders unnecessarily when props haven't changed.

17. Debouncing in React: Create a search input field where the user types a query. Implement debouncing so that the search query is only submitted after a delay (e.g., 500ms) to avoid making unnecessary API calls

18. **Problem**: Create a component that updates the document title based on a counter. Use useEffect to update the title every time the counter changes.

19. **Problem**: Create a component that fetches a list of users from one API and a list of posts from another API. Use two useEffect hooks to handle the different fetch requests simultaneously.

20. **Problem:** Create a React component called UserProfile that takes in a name, age, and location as props. The component should display these details in a structured format. Ensure that the component handles default values in case props are missing.

21. **Problem:** Write a React component that accepts name (string), age (number), and isMember (boolean) as props. Use PropTypes to validate that the correct data types are passed.

22. **Problem:** Create a component MessageBox that accepts two props: message (required) and isImportant (optional, default false). If isImportant is true, the message should be displayed with a red border and a bold font.

24. **Problem:** Create a Card component that takes a prop type (either "basic" or "premium"). If the type is "premium", the component should render additional content. If the type is "basic", it should render a simple card with a title and description.

25. **Problem:** Create a component Counter that accepts a prop onCountChange, which is a function. Each time a button is clicked, the Counter should increase the count by one and call the onCountChange function with the updated count.

26. **Problem:** Create a component ExpensiveComponent that accepts a value prop. The component renders a calculation that takes a long time. Use React.memo to prevent unnecessary re-renders when the value prop hasn't changed.

27. **Problem:** Create a component Product that accepts props name, price, and currency. Use destructuring to extract the props in the function signature and render the product details accordingly.

28. **Question:** Create a form with two inputs: name and email. When the user submits the form, log the state to the console.

29. **Question:** Implement a simple counter using useReducer instead of useState. The counter should support incrementing and decrementing.

30. **Question:** Create a parent component that contains the state and a child component that displays and updates the state.