React Day 1

Props and useState Assignment

Question 1: Simple Props

Create a functional component called `Greeting` that accepts a `name` prop and displays a greeting message, e.g., 'Hello, [name]!'.

Question 2: Multiple Props

Build a `ProfileCard` component that takes `name`, `age`, and `location` as props and displays them in a card layout.

Question 3: Conditional Rendering with Props

Create a `StatusMessage` component that takes a `status` prop. If `status` is 'success', display 'Operation was successful'. If `status` is 'error', display 'There was an error'.

Question 4: useState Basics

Create a `Counter` component that uses `useState` to track and display a count. Include buttons to increment and decrement the count.

Question 5: Props and useState Interaction

Create a `LikeButton` component. It should accept a `likeCount` prop, but internally manage the count using `useState`. Clicking the button should increase the like count.

React Day 1

Props and useState Assignment

Question 6: State Initialization with Props

Create a `Timer` component that accepts a `start` prop to set the initial time in seconds. Use `useState` to manage the timer's state and display the current time.

Question 7: Updating Parent State from Child

Create a `ColorPicker` component that allows the user to select a color. Pass a callback function via props to update the selected color in the parent component's state.

Question 8: Controlled vs Uncontrolled Components

Build a `TextInput` component with an input field. Use `useState` to control the input's value, and pass the current value back to the parent component using a prop.

Question 9: Props as Functions

Create a `ToggleSwitch` component. It should accept a `toggleState` prop which is a function. When the switch is clicked, it should call `toggleState` to update the parent component's state.

Question 10: Rendering Lists with Props

Create a `TodoList` component that accepts an array of todos via props. Render each todo item in the list.