**React Assignment**

**Module – 4**

**Qs 1 :- Explain Life cycle in Class Component and functional component with Hooks**

**Ans :**

Certainly! Let's break down the lifecycle phases step by step for both class components and functional components with hooks:

>> Class Components (with Lifecycle Methods):

1. Mounting Phase :

**Constructor :** This is where you initialize state and bind event handlers. It's like setting up your workspace before starting a project.

**Render :** This method returns the JSX representing the component's UI. Think of it as drawing the blueprint of your component.

**componentDidMount :** After the component is rendered in the DOM, this method is called. It's used for tasks that require the DOM, like fetching data from an API. It's like flipping the switch to turn on the component.

**2. Updating Phase :**

**Render :** Whenever the component's state or props change, the `render` method is called again to update the UI.

**componentDidUpdate :** After the component is updated, this method is called. You can perform actions here that depend on the changed state or props. It's like making adjustments to your workspace after changing the project requirements.

**3. Unmounting Phase :**

**componentWillUnmount :** Just before the component is removed from the DOM, this method is called. You can clean up any resources used by the component here, like event listeners or timers. It's like tidying up your workspace after completing a project.

**>> Functional Components (with Hooks):**

**1. Mounting Phase :**

**useState :** You use this hook to add state to your functional component. It's like getting your tools ready before starting work.

**useEffect (with empty dependency array) :** This hook is used to perform side effects after the component is rendered. When the dependency array is empty, it's similar to `componentDidMount`, as it runs once after the initial render. You can fetch data or set up subscriptions here.

**2. Updating Phase :**

**useState :** If the state changes, the component re-renders, and the updated state is reflected in the UI.

**useEffect (with dependency array) :** This hook is used to perform side effects after every render. When the dependency array contains variables, it's similar to `componentDidUpdate`, as it runs after every update. You can perform actions here based on the updated state or props.

**3. No Explicit Unmounting Phase :**

Functional components with hooks don't have an explicit unmounting phase like class components. React automatically handles cleanup when the component is unmounted or removed from the DOM.

In summary, both class components and functional components with hooks go through similar phases of mounting, updating, and unmounting, but they use different methods or hooks to achieve the same results.