**React List and Hooks**

**Q.1 Explain Life cycle in Class Component and functional component with Hooks.**

**ANS -** There are two types of components in React: class components and functional components with hooks. Both of these types of components have a lifecycle that describes the various stages of the component's existence, from initialization to destruction.

**Class Component Lifecycle**

The lifecycle methods are defined as member functions of the component class. The class component lifecycle has three main phases:

1. Mounting: This phase occurs when the component is first created and added to the DOM. During this phase, the following lifecycle methods are called in order:

* **`constructor()`**: This method is called when the component is first initialized. It is used to set the initial state and bind event handlers.
* **`static getDerivedStateFromProps()`**: This method is called whenever the component receives new props. It is used to update the state based on the new props.
* **`render()`**: This method is called to generate the initial HTML markup for the component and its children.

**`componentDidMount()`**: This method is called immediately after the component is added to the DOM. It is used to perform any necessary setup, such as fetching data from an API or adding event listeners.

componentDidMount() {

window.addEventListener('unhandledRejection', handler);

}

**2.**Updating: This phase occurs when the component's props or state change. During this phase, the following lifecycle methods are called in order:

* **`static getDerivedStateFromProps()`**: This method is called whenever the component receives new props. It is used to update the state based on the new props.
* **`shouldComponentUpdate()`**: This method is called to determine if the component needs to be updated. It is used to optimize performance by preventing unnecessary updates.
* **`render()`**: This method is called to generate the updated HTML markup for the component and its children.
* **`getSnapshotBeforeUpdate()`**: This method is called immediately before the updated markup is rendered to the DOM. It is used to capture information about the DOM before it changes.
* **`componentDidUpdate()`**: This method is called immediately after the updated markup is rendered to the DOM. It is used to perform any necessary cleanup or additional setup.
* componentDidUpdate(prevProps, prevState) {
* const { counter } = this.props;
* if (this.props.counter !== prevState.counter) {
* // some action here
* }
* }
* 3.Unmounting: This phase occurs when the component is removed from the DOM. During this phase, the following lifecycle method is called:
* **`componentWillUnmount()`**: This method is called immediately before the component is removed from the DOM. It is used to perform any necessary cleanup, such as removing event listeners or timers.
* componentWillUnmount() {
* alert('The component is going to be unmounted');
* window.removeEventListener('unhandledRejection', handler);
* }

### Functional Component with Hooks Lifecycle

Functional components with hooks follow a similar lifecycle to class components, but the lifecycle methods are defined as separate functions using React Hooks. The functional component with hooks lifecycle has the following phases:

1. Mounting: This phase occurs when the component is first created and added to the DOM. During this phase, the following hooks are called in order:

* **`useState()`**: This hook is used to define and update the component's state.
* **`useEffect()`**: This hook is used to perform any necessary setup, such as fetching data from an API or adding event listeners.

1. Updating: This phase occurs when the component's props or state change. During this phase, the following hooks are called in order:

* **`useState()`**: This hook is used to define and update the component's state.
* **`useEffect()`**: This hook is used to perform any necessary cleanup or additional setup.
* useEffect(() => {
* // action here
* }, [props.counter]); // checks for changes in the values in this array
* 3.Unmounting: This phase occurs when the component is removed from the DOM. During this phase, the following hook is called:
* **`useEffect()`**: This hook is used to perform any necessary cleanup, such as removing event listeners or timers.
* **useEffect()**: This hook is used to perform any necessary cleanup, such as removing event listeners or timers.