Lists & Hooks Assignment
(ReactJS Assignment Module-4)

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

React Lifecycle of Components :

Every React Component has a lifecycle of its own, lifecycle of a component can be defined as the series of methods that are invoked in different stages of the component’s existence. The definition is pretty straightforward but what do we mean by different stages? A React Component can go through four stages of its life as follows.

Initialization: This is the stage where the component is constructed with the given Props and default state. This is done in the constructor of a Component Class.

Mounting: Mounting is the stage of rendering the JSX returned by the render method itself.

Updating: Updating is the stage when the state of a component is updated and the application is repainted.

Unmounting: As the name suggests Unmounting is the final step of the component lifecycle where the component is removed from the page.

Implementing the Component Lifecycle methods

Steps to Create React Application

Step 1: Initialize the React Project using this command in the terminal.

```
npx create-react-app myapp
```

Step 2: Move to the project directory

```
cd myapp
```

Steps to Run the Application: Use this command in the terminal inside the project directory

```
npm start
```

Output: This output will be visible on the http://localhost:3000 on the browser window.

Functional Components in React :

ReactJS Functional components are some of the more common components that will come across while working in React. These are simply JavaScript functions. We can create a functional component in React by writing a JavaScript function. These functions may or may not receive data as parameters. In the functional Components, the return value is the JSX code to render to the DOM tree.

Ways to call the functional component:

We can call the functions in javaScript in other ways as follows:

1. Call the function by using the name of the function followed by the Parentheses.

```
// Example of Calling the function with function name followed by Parentheses
function Parentheses() {
   return (<h1>
               We can call function using name of the
               function followed by Parentheses
           </h1>);
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(Parentheses());
```

2. Call the function by using the functional component method.

```
// Example of Calling the function using component call
function Comp() {
   return (<h1> As usual we can call the function using component call</h1>);
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Comp />);
```

1. Call the function by using the name of the function followed by the Parentheses.

```
// Example of Calling the function with function name followed by Parentheses
function Parentheses() {
    return (<h1>
                We can call function using name of the
                function followed by Parentheses
            </h1>);
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(Parentheses());
```

2. Call the function by using the functional component method.

```
// Example of Calling the function using component call
function Comp() {
    return (<h1> As usual we can call the function using component
call</h1>);
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Comp />);
```

Step to run the application: Open the terminal and type the following command.

```
npm start
```

Output: Open the browser and our project is shown in the URL http://localhost:3000/