React Test

Q1) Here we have class component that updates the state using the input from a form.

```
export class Profile extends Component {
 state = {
  name: "Backbencher",
  age: 23,
 };

 onNameChange = (e) => {
  this.setState({
   name: e.target.value,
  });
 };

 onAgeChange = (e) => {
  this.setState({
   age: e.target.value,
  });
 };

 render() {
  return (
   <div>
    <form>
     <input
      type="text"
      value={this.state.name}
      onChange={this.onNameChange}
     />
     <input
```

```
          type="text"
          value={this.state.age}
          onChange={this.onAgeChange}
        />
        <h2>
          Name: {this.state.name}, Age: {this.state.age}
        </h2>
      </form>
    </div>
  );
 }
}
```

Rewrite the same component using React hooks.

Q2) Here is a class component that prints Boom in console whenever it is mounted or updated.

```
export class Banner extends Component {
 state = {
  count: 0,
 };

 updateState = () => {
  this.setState({
   count: this.state.count + 1,
  });
 };

 componentDidMount() {
  console.log("Boom");
 }
 componentDidUpdate() {
  console.log("Boom");
 }

 render() {
  return (
   <div>
    <button onClick={this.updateState}>State: {this.state.count}</button>
   </div>
  );
 }
}
```

Remove the redundant console.log statement using React hooks.

Q3) Here we have a class component with a state value. Each time the button in component is clicked, the count is incremented.

```
class Counter extends Component {
 state = {
  count: 0,
 };

 incrementCount = () => {
  this.setState({
   count: this.state.count + 1,
  });
 };

 render() {
  return (
   <div>
    <button onClick={this.incrementCount}>Count: {this.state.count}</button>
   </div>
```

```
    );
  }
}
```

Rewrite this component using React hooks.

Q4) Understand the code below:

```
function Banner() {
  const [count, setCount] = useState(0);
  const [name, setName] = useState("");

  useEffect(() => {
    console.log("Count is updated");
  });

  return (
    <div>
      <button onClick={() => setCount(count + 1)}>State: {count}</button>
      <input
        type="text"
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
    </div>
  );
}
```

It logs "Count is updated" message even when updating the value in textbox. How can we show the log message only when the count state is updated?

Q5) What will be the output of the following code?. Explain the reason behind your answer.

```
import React, { createContext, useContext } from 'react';


const MyContext = createContext(1);
const MyComponent = () => (
   <>
       <p>{useContext(MyContext)}</p>
       <MyContext.Provider value={2}>
           <p>{useContext(MyContext)}</p>
       </MyContext.Provider>
   </>
);

export default MyComponent;
```

Q6) Which component will be rendered by the following code when navigating to '/login' route ? Give explanation for your answer.

```
ReactDOM.render((
<Router>
<div>
<Route path="/" render={Home} />
<Route path="/login" render={Login} />
</div>
</Router>),
document.getElementById('root')
);
```

Q7) Study the following piece of code and suggest changes such that only the Profile component is Rendered when the path is '/dashboard/profile'.

```
import React from 'react;
import { BrowserRouter, Route } from 'react-router-dom';
const App = () => {
  return (<div>App</div>)
}
const Dashboard = () => {
  return (<div>Dashboard</div>)
}
const Profile = () => {
  return (<div>Profile</div>)
}
const Router = () => {
  return (<BrowserRouter>
      <Route path='/' component={App}></Route>
      <Route path='/dashboard/profile' component={Profile}></Route>
      <Route path='/dashboard' component={Dashboard}></Route>
    </BrowserRouter>
  )
}
```

Q8) Explain the variations of useEffect.

Q9) We have a code snippet from a class component which registers and remove an event listener.

```
componentDidMount() {
  window.addEventListener("mousemove", this.handleMousePosition);
}

componentWillUnmount() {
  window.removeEventListener("mousemove", this.handleMousePosition);
}
```

Convert this code to React hooks format.

Q 10) Class component, ProviderComponent provides two context values.

```
export const NameContext = React.createContext();
export const AgeContext = React.createContext();

export class ProviderComponent extends Component {
  render() {
    return (
      <NameContext.Provider value="Backbencher">
        <AgeContext.Provider value="23">
          <Test2 />
        </AgeContext.Provider>
      </NameContext.Provider>
    );
  }
}
```

We have Test2 with following code.

```
import React from 'react'

function Test2() {
    return (
        <div>

        </div>
    )
}
export default Test2
```

Complete Test2 component to consume the context values and display the name and age.