**ASSIGNMENT ON REACT HOOKS**

THEORY EXERCISE

**Question 1: What are React hooks? How do useState() and useEffect() hooks work in functional components?**

React Hooks are special functions that allow developers to use state, lifecycle methods, and other React features in functional components, without converting them into class components.  
useState(): Allows functional components to maintain internal state.  
Example:  
const [count, setCount] = useState(0);  
useEffect(): Used to perform side effects like data fetching, subscriptions, or manual DOM manipulation.  
Example:  
useEffect(() => { console.log('Component mounted or updated'); }, [dependencies]);

**Question 2: What problems did hooks solve in React development? Why are hooks considered an important addition to React?**

Hooks solve several problems:  
1. Complexity in class components.  
2. Code reuse difficulties.  
3. Confusing 'this' keyword usage.  
Importance:  
- Enable cleaner functional components.  
- Promote reusable logic through custom hooks.  
- Reduce boilerplate code and improve readability.

**Question 3: What is useReducer? How do we use it in a React app?**

useReducer is a hook for complex state management.  
Syntax:  
const [state, dispatch] = useReducer(reducer, initialState);  
Example:  
const initialState = {count:0};  
function reducer(state, action){  
 switch(action.type){  
 case 'increment': return {count:state.count+1};  
 case 'decrement': return {count:state.count-1};  
 default: return state;  
}  
}  
const [state, dispatch] = useReducer(reducer, initialState);

**Question 4: What is the purpose of useCallback & useMemo Hooks?**

useCallback: Returns a memoized function to prevent unnecessary re-creation on every render.  
useMemo: Returns a memoized value computed from a function to avoid expensive recalculations.

**Question 5: What’s the difference between useCallback & useMemo Hooks?**

| Feature | useCallback | useMemo |  
|---------|-------------|--------|  
| Returns | Memoized function | Memoized value |  
| Usage | Avoid re-creating functions unnecessarily | Avoid re-computing expensive values |  
| Syntax | useCallback(fn, [deps]) | useMemo(() => computeValue(), [deps]) |

**Question 6: What is useRef? How does it work in a React app?**

useRef is a hook that returns a mutable ref object.  
It is used to:  
1. Access DOM elements directly.  
2. Persist values across renders without triggering re-renders.  
Example:  
const inputRef = useRef(null);  
function focusInput(){ inputRef.current.focus(); }  
return <input ref={inputRef} />;