1. What is the purpose of useEffect in React?
useEffect is used to perform side effects in functional components such as API calls, DOM updates, timers, subscriptions, and event listeners after rendering.

2. What problem does useMemo solve?
useMemo prevents expensive calculations from re-running on every render by memoizing the computed value until dependencies change.

3. Why do we use useCallback in React?
useCallback memoizes function references so that functions are not recreated on every render, helping prevent unnecessary child re-renders.

4. Difference between useEffect and useMemo?
useEffect is used for side effects, while useMemo is used for performance optimization by memoizing values.

5. Difference between useMemo and useCallback?
useMemo memoizes values, while useCallback memoizes functions.

6. What happens if you don’t provide a dependency array?
The hook runs on every render, which can cause performance issues or infinite loops.

7. When does useMemo recompute its value?
useMemo recomputes its value only when one of its dependencies changes.

8. How does useCallback prevent unnecessary child re-renders?
By passing the same function reference to child components, React.memo prevents re-rendering unless dependencies change.

9. Can we perform API calls inside useMemo or useCallback?
No. API calls are side effects and should be done in useEffect. useMemo and useCallback must remain pure.

10. Real-time scenario using all three hooks together
Fetching products using useEffect, filtering products using useMemo, and passing event handlers to child components using useCallback.