**Q.1. What is the difference between composition and inheritance? Which paradigm is**

**preferred in React? Write a brief note about it in your own words.**

Inheritance and composition are two programming techniques developers use to establish relationships between classes and objects. Whereas inheritance derives one class from another, composition defines a class as the sum of its parts.

Classes and objects created through inheritance are tightly coupled because changing the parent or superclass in an inheritance relationship risk breaking your code. Classes and objects created through composition are loosely coupled, meaning that you can more easily change the component parts without breaking your code.

Because loosely coupled code offers more flexibility, many developers have learned that composition is a better technique than inheritance, but the truth is more complex. Choosing a programming tool is similar to choosing the correct kitchen tool: You wouldn't use a butter knife to cut vegetables, and in the same

way you shouldn't choose composition for every programming scenario.

In computing, reactive programming is a declarative programming paradigm concerned with data streams and the propagation of change.

**Q. 2. In what cases do we need to 'lift the state up'?**

Lifting up the State: As we know, every component in React has its own state. Because of this sometimes data can be redundant and inconsistent. So, by Lifting up the state we make the state of the parent component as a single source of truth and pass the data of the parent in its children.

Time to use Lift up the State: If the data in “parent and children's components” or in “cousin

components” is Not in Sync.

Example 1: If we have 2 components in our App. A -> B where, A is parent of B. keeping

the same data in both Component A and B might cause inconsistency of data.

Example 2: If we have 3 components in our App.

A

/

\

B

C

Where A is the parent of B and C. In this case, if there is some Data only in component B

but, component C also wants that data. We know Component C cannot access the data

because a component can talk only to its parent or child (Not cousins).

**Q. 3. What is a handler function. When do we use it?**

A handler is a routine/function/method which is specialized in a certain type of data or focused on certain special tasks. Examples: Event handler - Receives and digests events and signals from the surrounding system (e.g., OS or GUI). Memory handler - Performs certain special tasks on memory