1. **What and why do we use @composable?**

Ans. @composable:

What: @composable is an annotation in Jetpack Compose used to define a composable function. Composable functions are responsible for describing the UI elements based on the current state of the application.

Why: Composable functions are used to build UI components that automatically update when the underlying data changes. The annotation indicates that the function's output depends only on its input parameters, making it suitable for declarative UI programming.

2. **What is Surface?**

Ans. Surface:

What: Surface is a basic container element in Jetpack Compose that is used to draw a background color and elevation.

Why: It provides a way to apply a background color and shadow to a UI element, helping to visually separate and organize different parts of the UI.

3. **Why would you use Preview?**

Preview:

Why: The Preview annotation in Jetpack Compose is used to generate a preview of a composable function. This is particularly helpful during development as it allows you to see how a UI component will look without running the entire application.

4. **How to add components such as Text and Button?**

Ans. Adding Components (e.g., Text and Button):

You can add components by creating composable functions that describe the UI elements. For example:

```
@Composable
fun MyTextComponent() {
   Text(text = "Hello, World!")
}
@Composable
fun MyButtonComponent() {
   Button(onClick = { /* Handle button click */ }) {
      Text(text = "Click Me")
   }
}
```

5. How to create columns and rows in Kotlin?

Ans. Creating Columns and Rows:

You can use Column and Row composable functions to create vertical and horizontal layouts, respectively.

```
@Composable
fun MyColumnLayout() {
   Column {
      // Add composable components vertically  }}
@Composable
fun MyRowLayout() {
   Row {
      // Add composable components horizontally  }}
```

6. What is the state of a component in Compose?

Ans. State in Compose:

The state of a component in Jetpack Compose represents the current data that the UI is based on. State is mutable and can be changed, triggering recomposition of the UI to reflect the updated state

7. What is a lazy list?

Ans. Lazy List:

A lazy list in Jetpack Compose is a composable function that creates a list of items only for the visible items on the screen. It's efficient for handling large datasets without loading the entire list into memory.

8. What is/are the differences between Android Java way to build UI and Jetpack Compose for Kotlin?

Ans. Differences Between Android Java and Jetpack Compose:

Android Java UI development is imperative, where you define UI elements and their interactions imperatively in code. In contrast, Jetpack Compose is declarative, where you describe how the UI should look based on the current state, and the framework takes care of updating it when the state changes.

Jetpack Compose simplifies UI development with a more concise syntax, reduces boilerplate code, and provides a more intuitive way to handle UI changes.

Compose encourages the use of composable functions, making UI components modular and easier to test and maintain.

Jetpack Compose is specifically designed for modern Android development, while Android Java UI development relies on XML layouts and imperative programming.

9. Provide a screenshot that shows how many Codelabs you completed so far.

Ans. Screenshot :