Sample Questions

Q.1: Match the following Kotlin concepts in Column X with the appropriate description in Column Y.

Column X	Column Y
(A) **Data Classes**	(i) A special type of class in Kotlin designed to hold data with automatic toString(), equals(), and hashCode() methods.
(B) **Sealed Classes**	(ii) A restricted class hierarchy that allows defining a set of types that can be used in when expressions.
(C) **Companion Object**	(iii) A way to define static methods and properties in a Kotlin class.
(D) **Lateinit Property**	(iv) A property that is initialized at a later stage instead of during declaration.
(E) **Delegation**	(v) A technique in Kotlin where a class delegates some of its responsibilities to another object.

Q.2: Android classifies screen densities into different categories (e.g., mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi). What is the purpose of these density buckets, and how does Android use them?

Q.3: Fix the Errors in Kotlin Lambda Functions

The following Kotlin code contains errors in lambda function syntax and usage. Identify and correct them.

```
fun main() {
    val multiply = (a: Int, b: Int) -> a * b
    println(multiply(4))
}
```

Instructions:

- Identify and fix the lambda function syntax error.
- Explain the correction.
- Provide the corrected code.

Q.4: Match the following Kotlin concepts in Column X with the appropriate description in Column Y.

Column X	Column Y
(A) **Mutable Collection**	(i) A collection type that allows adding, removing, or modifying elements.
(B) **Immutable Collection**	(ii) A collection type where elements cannot be changed after initialization.
(C) **Lazy Initialization**	(iii) A way to initialize a property only when it is accessed for the first time.
(D) **Generic Function**	(iv) A function that allows working with different data types by using type parameters.
(E) **Type Casting**	(v) The process of converting one type to another using as or is keywords in Kotlin.

Q.5: Consider the following XML layout file for an Android activity:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/titleText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome to My App"
        android:textSize="20sp"
        android:textStyle="bold"/>

    <Button
        android:id="@+id/submitButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        android:layout_gravity="center_horizontal"/>

</LinearLayout>
```

Based on the above XML layout file, answer the following questions:

1. Identify and describe the **parent layout** used in this XML. What does the `android:orientation="vertical"` attribute do?
2. Explain the function of the **`TextView`**, and **`Button`** elements in this layout.
3. How does the **`layout_gravity="center_horizontal"`** attribute in the `Button` affect its positioning?
4. If you wanted the `Button` to **span the full width of the screen**, what property would you modify?

Q.6: Fix the Errors in Kotlin Null Safety

The following Kotlin code contains errors related to null safety. Identify and correct the errors so the program executes without crashing.

```
fun main() {
    var name: String = null
    println(name.length)
}
```

Instructions:

- Identify and correct the null safety issue.
- Explain the role of Kotlin's null safety features.
- Provide the corrected code.

Q.7: Choosing the Right Kotlin Feature for Collections and Data Processing

Consider the following four scenarios involving Kotlin programming. For each scenario, indicate whether a **list filter, a map transformation, a lazy property, or a generic function** is the most appropriate approach, and provide your justification.

i. **Scenario:** You have a list of numbers and want to create a new list containing only the even numbers.
Which Kotlin concept is more appropriate in this scenario?

ii. **Scenario:** You have a list of names and want to transform each name into uppercase.
Which Kotlin concept is more appropriate in this scenario?

iii. **Scenario:** You need to declare a property that should only be initialized when it is accessed for the first time.
Which Kotlin concept is more appropriate in this scenario?

iv. **Scenario:** You are creating a function that should work with multiple data types and allow type flexibility while ensuring type safety.
Which Kotlin concept is more appropriate in this scenario?

Q.8: Match the following Kotlin concepts in Column X with the appropriate description in Column Y.

Column X	**Column Y**
(A) **Higher-Order Function**	(i) A function that takes another function as a parameter or returns a function.
(B) **Inline Function**	(ii) A function that eliminates function call overhead by inserting the function body directly at the call site.
(C) **Scope Functions**	(iii) A set of functions (let, apply, run, also, with) that help in object manipulation and scope restriction.
(D) **Suspend Function**	(iv) A function in Kotlin that is designed to work with coroutines and supports asynchronous programming.
(E) **Receiver Function**	(v) A function that allows calling an object method as if it were an extension function.

Q.9: Choosing the Right Kotlin Feature for Functionality

Consider the following four scenarios involving Kotlin programming. For each scenario, indicate whether a **higher-order function, an inline function, a suspend function, or a receiver function** is the most appropriate approach, and provide your justification.

i. **Scenario:** You need to pass a function as a parameter to another function and execute it within the function body.
Which Kotlin concept is more appropriate in this scenario?

ii. **Scenario:** You want to optimize a small function by eliminating function call overhead, ensuring its body is directly inserted at the call site.
Which Kotlin concept is more appropriate in this scenario?

iii. **Scenario:** You are implementing a coroutine-based function that needs to perform asynchronous operations without blocking the main thread.
Which Kotlin concept is more appropriate in this scenario?

iv. **Scenario:** You want to define a function that extends an existing class but is callable as if it were a method of that class.
Which Kotlin concept is more appropriate in this scenario?

Q.10: Fix the Errors in Kotlin Inheritance

The following Kotlin program has errors related to class inheritance. Identify and fix them.

```
open class Animal {
    fun makeSound()
}

class Dog : Animal() {
    override fun makeSound() {
        println("Bark!")
    }
}

fun main() {
    val dog = Dog()
    dog.makeSound()
}
```

Instructions:

- Identify and correct the errors in class inheritance.
- Explain the corrections.
- Provide the corrected code.

Q.11: Choosing the Right Kotlin Feature for Object-Oriented Design

Consider the following four scenarios involving Kotlin programming. For each scenario, indicate whether an **interface, an abstract class, an open class, or an override function** is the most appropriate approach, and provide your justification.

i. **Scenario:** You need to define a contract for a class that provides method declarations but no default implementations.
Which Kotlin concept is more appropriate in this scenario?

ii. **Scenario:** You want to define a base class with some implemented methods while allowing derived classes to provide their own specific implementations.
Which Kotlin concept is more appropriate in this scenario?

iii. **Scenario:** You need to create a class that can be inherited by other classes but is not final by default.
Which Kotlin concept is more appropriate in this scenario?

iv. **Scenario:** You need to modify the behavior of a function from a parent class in a child class.
Which Kotlin concept is more appropriate in this scenario?

Q.12: Fix the Errors in Kotlin Functions

The following Kotlin code contains errors related to function syntax and return types**.** Identify and correct the errors so the code runs successfully.

```
fun addNumbers(a: Int, b: Int):
    return a + b

fun main() {
    val result = addNumbers(5, "10")
    println("Result: " + result)
}
```

Instructions:

- Identify and correct the errors.
- Explain the issues and why the corrected version works.

Q.13: What is the key difference between LinearLayout and ConstraintLayout in Android UI design?

Q.14: When should you prefer ConstraintLayout over LinearLayout in an Android app? Provide an example.

Q.15: Explain the significance of weight in LinearLayout and how it affects UI element distribution.

Q.16: Consider the following XML layout file for an Android activity:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/titleText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome to My App"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center"/>

    <EditText
        android:id="@+id/inputField"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter your name"
        android:layout_marginTop="16dp"/>

    <Button
        android:id="@+id/submitButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="20dp"/>

</LinearLayout>
```

Tasks & Questions:

1. Identify and explain the **layout type** used in this XML file. What does `android:orientation="vertical"` do?
2. How is the `Button` centered horizontally in this layout? What alternative method could be used?
3. Modify this XML to use **ConstraintLayout** instead of **LinearLayout** while maintaining the same UI design.
4. If you needed to **add another button aligned to the bottom of the screen**, which layout (LinearLayout or ConstraintLayout) would be more efficient, and why?

Q.17: What is the purpose of the onCreate method in an Android Activity lifecycle?

Q.18: Why is it necessary to call setContentView(R.layout.activity_main) inside onCreate?

Q.19: Explain how findViewById() is used in onCreate to reference UI elements in an XML layout. What are the advantages of defining UI in XML instead of programmatically in onCreate?

Q.20: Consider the following XML layout file for an Android Activity:

```
<?xml version="1.0" encoding="utf-8"?>
<ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/titleText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome!"
        android:textSize="24sp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>

    <EditText
        android:id="@+id/inputField"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Enter your name"
        app:layout_constraintTop_toBottomOf="@id/titleText"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="16dp"/>

    <Button
        android:id="@+id/submitButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        app:layout_constraintTop_toBottomOf="@id/inputField"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="20dp"/>

</ConstraintLayout>

import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // TODO: Initialize UI components and set click listener for
button
    }
}
```

Tasks & Questions:

- In the given MainActivity.kt, explain what super.onCreate(savedInstanceState) does.
- Complete the onCreate method by initializing EditText and Button using findViewById().
- Set a click listener on the submitButton so that when clicked, it updates the TextView (titleText) with the text entered in the EditText field.
- Convert this layout into LinearLayout while maintaining the same UI design and constraints.
- How would you modify this layout if you needed the Button to always remain fixed at the bottom of the screen?

Q.21: Explain the purpose of the following directories and files in an Android project:

- `manifests/AndroidManifest.xml`
- `java/com/example/myapp/MainActivity.kt`
- `res/layout/activity_main.xml`
- `res/drawable/`
- `gradle.build (Module: app)`

Q.22: Match the following Kotlin concepts in Column X with the appropriate description in Column Y.

Column X	Column Y
(A) **Interface**	(i) A contract that defines methods but does not provide implementations.
(B) **Abstract Class**	(ii) A class that cannot be instantiated and may have abstract methods.
(C) **Override Keyword**	(iii) A keyword used to provide a new implementation of an inherited function.
(D) **Primary Constructor**	(iv) The main constructor of a Kotlin class, declared in the class header.
(E) **Secondary Constructor**	(v) An additional constructor in a Kotlin class that provides an alternate way to initialize objects.

Q.23: What is the difference between the res/values/strings.xml file and res/layout/activity_main.xml?

Q.24: Why does Android separate the Java/Kotlin source code from the UI resources (layouts, images, strings, etc.)?

Q.25: What is the difference between DP (Density-independent Pixels) and PX (Pixels) in Android development? Why is DP preferred over PX for UI design?