USN											CMRIT CMR INSTITUTE OF TECHNOLOGY, BENGALURU. ACCREDITED WITH A++ GRADE BY NAAC

Internal Assessment Test 2 Set-1 – May 2025							
Sub:	Mobile Application Development			Sub Code:	BIS654C	Branch	ECE
Date:	27-05-2025	Duration:	90 min's	Max Marks: 50	Sem/Sec:	VI / A, B , C, D	OBE

	Answer any FIVE questions	MARKS	CO	RBT
1	a) Explain the use of ProgressBar in Android. Write the XML and Java code to implement an indeterminate horizontal ProgressBar. b) What is AutoCompleteTextView? How is it implemented using an ArrayAdapter? Explain with an example.	5 5	CO3	L1
2	a) Define Spinner. How can a Spinner be implemented in Android using XML and Java code? b) How is ListView different from Spinner? Write code to display a list of city names using ListView.	4 6	CO3	L1
3	a) What is an Activity in Android? Describe its role with an example. b) Explain Intent in Android. Differentiate between explicit and implicit intents with code examples.	5 5	CO4	L2
4	a) What is an Intent Filter? How is it used in AndroidManifest.xml? b) Describe the Activity Lifecycle with a labeled diagram and explain each stage.	5 5	CO4	L1
5	a) What is SQLite? Why is it preferred for local storage in Android apps? b) Write steps to create a SQLite database using SQLiteOpenHelper. Provide the onCreate and onUpgrade method examples.	4 6	CO5	L1
6	a) How do you open a database connection and insert records in SQLite? b) What are ContentValues and how are they used for inserting data in SQLite?	5 5	CO5	L2

Faculty Signature CCI Signature HOD Signature

Ans 1a-

Scheme- Explanation + Code- 2 + 3

ProgressBar is a visual indicator in Android that shows the progress of a task. It is a subclass of the View class and is typically used when the application is performing tasks like file downloads, buffering, or any background activity that requires user patience.

There are two main types of ProgressBars in Android:

- Indeterminate – Used when the duration of the task is unknown (e.g., loading content).
- Determinate – Used when the progress can be measured and displayed.

An indeterminate ProgressBar continuously animates, indicating that a task is ongoing but without displaying the actual progress.

A horizontal indeterminate ProgressBar combines the benefits of a progress bar style with continuous feedback.

Xml code-

```
<ProgressBar
    android:id="@+id/progressBar"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:indeterminate="true"
    android:visibility="visible" />
```

java code-

```
ProgressBar progressBar;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    progressBar = findViewById(R.id.progressBar);
    progressBar.setVisibility(View.VISIBLE);  // show progress
}
```

Ans 1b-

Scheme- Explanation + Code- 2 + 3

AutoCompleteTextView is a subclass of EditText in Android which provides suggestions to the user as they type. These suggestions can be from an array, database, or any other data source.
It is especially useful in cases like:

- City or country name selection
- Email ID or domain suggestion
- Search-based applications

Key Features:

- Extends EditText
- Dropdown list appears with suggestions
- Powered by an Adapter (commonly ArrayAdapter)

Xml code-

```
<AutoCompleteTextView
    android:id="@+id/autoCompleteTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="Enter Country Name" />
```

java code-

```
AutoCompleteTextView autoCompleteTextView;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    String[] countries = {"India", "Indonesia", "Italy", "Iceland"};
    autoCompleteTextView = findViewById(R.id.autoCompleteTextView);
    ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
        android.R.layout.simple_dropdown_item_1line, countries);
    autoCompleteTextView.setAdapter(adapter);
}
```

Ans 2a- Scheme- Explanation + Code- 2 + 2

A Spinner in Android is a UI widget that displays a dropdown list of options from which a user can select a single value. It is similar to a combo box in other programming environments. Spinners are useful when:

- You want to save screen space.
- You have a finite set of options (e.g., selecting a country, gender, or language).

Key Features:

- Shows selected item when not expanded.
- Opens a dropdown on tap.
- Works with adapters (especially ArrayAdapter).

Xml code-

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Java code-

```
Spinner spinner;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    spinner = findViewById(R.id.spinner);

    String[] cities = {"Delhi", "Mumbai", "Kolkata", "Chennai"};

    ArrayAdapter<String> adapter = new ArrayAdapter<>(
        this, android.R.layout.simple_spinner_item, cities);

    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    spinner.setAdapter(adapter);
}
```

Ans 2b- Scheme- Explanation + Code- 2 + 4

In Android development, ListView and Spinner are two frequently used UI components for displaying a list of items. However, they differ significantly in terms of appearance, user interaction, and use cases.

Feature	**ListView**	**Spinner**
Display Style	Vertically scrollable list	Dropdown menu style
Visible Items	Multiple items visible simultaneously	Only one item visible (collapsed view)
Screen Space Usage	Takes up more screen space	Space-efficient
Selection Behavior	Allows selection by clicking an item	Opens a dropdown from which the user selects
Use Case	Displaying large sets of data like messages or contact lists	Selecting from a small set of options like gender, language, or country
User Interaction	Scroll through and select an item; supports multiple interactions	Click to open, select one item

ListView is typically used when a long list of items is required to be displayed to the user in a vertically scrollable format, whereas Spinner is used when space needs to be conserved.

Xml code-

```
<ListView
   android:id="@+id/listView"
   android:layout_width="match_parent"
   android:layout_height="wrap_content" />
```

java code-

```
public class MainActivity extends AppCompatActivity {

   ListView listView;
   String[] cities = {"Delhi", "Mumbai", "Kolkata", "Chennai", "Bangalore"};

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      listView = findViewById(R.id.listView);

      ArrayAdapter<String> adapter = new ArrayAdapter<>(
         this,
         android.R.layout.simple_list_item_1,
         cities
```

```
        );

        listView.setAdapter(adapter);
    }
}
```

Ans 3a- Scheme- Explanation + Code- 2 + 3

An **Activity** in Android is one of the fundamental building blocks of an application. It represents a single screen with a user interface. Every application that has a visible user interface must have at least one activity, and all user interactions typically happen within these activities.
The Activity class is a subclass of ContextThemeWrapper and is a part of the Android app framework. It acts as an entry point for user interaction, manages the lifecycle of the app's screen, and handles events such as button clicks, screen rotations, and navigation between screens.
Each activity in an application is independent and can start another activity using an **Intent**. When a new activity starts, the current one is paused but remains in memory unless explicitly finished. The Android system manages the activity stack using the **back stack**, where the most recently opened activity appears at the top.

Java code-

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Ans 3b- Scheme- Explanation + Code- 2 + 3

An **Intent** in Android is a messaging object that allows application components to request functionality from other Android components. It is used to start activities, services, and broadcast receivers or to pass data between them.
There are two major types of intents:

1. **Explicit Intent**
2. **Implicit Intent**

1. Explicit Intent:
An explicit intent specifies the exact class to be started by using its fully qualified class name. It is commonly used to navigate from one activity to another within the same application.

Intent intent = new Intent(this, SecondActivity.class);
startActivity(intent);

2. Implicit Intent:
An implicit intent does not name a specific component; instead, it declares a general action to perform, which allows any application on the device that can handle the action to respond.

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://www.google.com"));
startActivity(intent);
```

An explicit intent is tightly bound to the internal structure of the app, while an implicit intent is flexible and leverages the Android ecosystem to allow components from other apps to handle the action. Explicit intents are best for internal navigation, and implicit intents are ideal when performing actions like sending emails, opening web pages, or sharing content.

Ans-4a- Scheme- Explanation + Code- 2 + 3
An **Intent Filter** in Android is a component of the application manifest that specifies the types of intents an activity, service, or broadcast receiver can respond to. It allows the Android system to determine which component is best suited to handle a given intent—especially **implicit intents**.

Structure of Intent Filter:
An intent filter is declared inside the <activity>, <service>, or <receiver> tag in the AndroidManifest.xml. It typically contains:

- <action>: Specifies the action name like android.intent.action.VIEW.
- <category>: Specifies additional info like android.intent.category.DEFAULT.
- <data>: Specifies the type of data (MIME type, URI scheme, etc.).

Xml code-
```
<activity android:name=".MainActivity">
   <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
</activity>
```

Ans-4b- Scheme- Explanation + Diagram- 2 + 3
In Android, an **Activity Lifecycle** represents the different states an activity goes through from the moment it is created until it is destroyed. Understanding these lifecycle methods is essential for managing app behavior, saving state, releasing resources, and improving user experience.

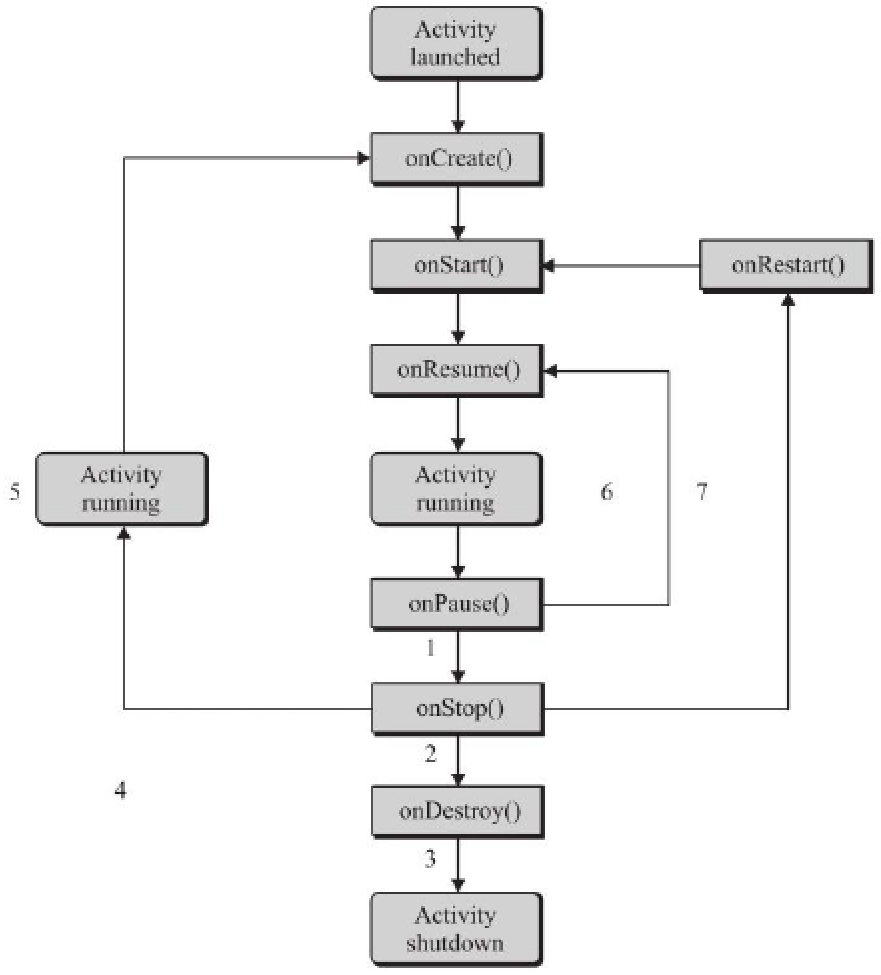

Figure 6.2:Active Lifecycle Diagram

Activity Lifecycle:

1. **onCreate()**
   Called when the activity is first created. Initialization of UI and data binding happens here.
2. **onStart()**
   The activity becomes visible to the user. At this stage, the activity is not yet in the foreground.
3. **onResume()**
   The activity starts interacting with the user. It is now in the foreground and at the top of the activity stack.
4. **onPause()**
   Another activity comes in front of this one. The current activity is still partially visible but not in focus. Ideal for saving small data and pausing animations.
5. **onStop()**
   The activity is no longer visible to the user. It is typically used to release resources like network connections or camera access.
6. **onRestart()**
   Called when the activity is being restarted after being stopped.
7. **onDestroy()**
   The activity is about to be destroyed. Used for final cleanup.

Ans-5a- Scheme- Definition + Explanation- 2 + 2

SQLite is a lightweight, relational, embedded database engine used in Android for storing structured data locally on the device. It is an open-source, serverless, self-contained SQL database engine.
Android includes SQLite as a built-in database for applications, allowing developers to perform standard SQL operations (such as INSERT, UPDATE, DELETE, and SELECT) using Java classes.

Key Features of SQLite:

- Serverless: No need for a separate server process.
- Zero Configuration: No setup or administration required.
- Cross-Platform: Works on many platforms with consistent behavior.
- Small Footprint: Minimal memory usage, perfect for mobile devices.
- Integrated: Available by default in Android SDK.

Why SQLite is Preferred in Android:

1. **Lightweight:**
   It consumes very little memory, making it ideal for resource-constrained environments like mobile devices.
2. **Persistent Local Storage:**
   Suitable for offline applications where data needs to be stored and retrieved without internet access.
3. **SQL Support:**
   Developers can use familiar SQL syntax to manage local data easily.
4. **No Network Dependency:**
   Unlike cloud databases, SQLite does not require internet connectivity.
5. **Integrated with Android:**
   Android provides native APIs like SQLiteOpenHelper, SQLiteDatabase, and Cursor for easy interaction.

Ans- 5b- Scheme- Explanation + Code- 2 + 4

To simplify the creation and management of SQLite databases in Android, the SDK provides a helper class called SQLiteOpenHelper. It manages the creation and versioning of the database.

Steps to Create a SQLite Database Using SQLiteOpenHelper:

Step 1: Create a Java Class that Extends SQLiteOpenHelper

```
public class MyDatabaseHelper extends SQLiteOpenHelper {
   private static final String DATABASE_NAME = "StudentDB";
   private static final int DATABASE_VERSION = 1;

   public MyDatabaseHelper(Context context) {
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
   }
```

```
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE students (id INTEGER PRIMARY KEY
AUTOINCREMENT, name TEXT, marks INTEGER);");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS students");
        onCreate(db);
    }
}
```

Step 2: Initialize the Database Helper in Activity

```
MyDatabaseHelper dbHelper = new MyDatabaseHelper(this);
SQLiteDatabase db = dbHelper.getWritableDatabase();
```

Constructor:
Calls the parent SQLiteOpenHelper constructor with context, database name, and version.
onCreate(SQLiteDatabase db):
Called when the database is created for the first time. This is where the schema is defined using SQL commands.
onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion):
Called when the database version is increased. It handles schema changes such as table alterations or deletions.

Ans 6a- Scheme- Explanation + Code- 2 + 3

Opening a SQLite Database in Android:
To open a database connection, we typically use the SQLiteOpenHelper class. The connection is established via getWritableDatabase() or getReadableDatabase() methods provided by the helper class.

- **getWritableDatabase()** returns an instance of SQLiteDatabase with read/write access.
- **getReadableDatabase()** returns a read-only instance if storage is full or write access isn't available.

Steps to Open a Database Connection and Insert Records:

Step 1: Create a Database Helper Class

```
public class MyDatabaseHelper extends SQLiteOpenHelper {

   private static final String DB_NAME = "StudentDB";
   private static final int DB_VERSION = 1;

   public MyDatabaseHelper(Context context) {
      super(context, DB_NAME, null, DB_VERSION);
   }

   @Override
   public void onCreate(SQLiteDatabase db) {
      db.execSQL("CREATE TABLE students (id INTEGER PRIMARY KEY
AUTOINCREMENT, name TEXT, marks INTEGER);");
   }

   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
      db.execSQL("DROP TABLE IF EXISTS students");
      onCreate(db);
   }
}
```

Step 2: Open the Database and Insert a Record

```
MyDatabaseHelper dbHelper = new MyDatabaseHelper(this);
SQLiteDatabase db = dbHelper.getWritableDatabase();

ContentValues values = new ContentValues();
values.put("name", "Ravi");
values.put("marks", 85);

long result = db.insert("students", null, values);
```

MyDatabaseHelper extends SQLiteOpenHelper and handles DB creation.
getWritableDatabase() returns a reference to the database.
insert() inserts the values into the table.

Ans 6b- Scheme- Explanation + Code- 2 + 3

ContentValues is a key class in Android used to store a set of values that the SQLiteDatabase class can process. It functions as a **name/value store**, where each column name (String) is associated with its corresponding value.

It is widely used with methods like:

- insert()
- update()

The keys must match the **column names** of the target database table.

Inserting Data Using ContentValues

```
ContentValues values = new ContentValues();
values.put("name", "Ravi");
values.put("marks", 85);

SQLiteDatabase db = dbHelper.getWritableDatabase();
db.insert("students", null, values);
```

- "name" and "marks" are column names from the students table.
- "Ravi" and 85 are the corresponding values to be inserted.
- The insert() method takes the table name, a nullColumnHack (can be null), and the ContentValues object.