M.Sc.(Computer Science) Sem – II

Mobile App Development Technologies

Slip 1

1. **Write an application to create a splash screen.** **[10 Marks]**

Soln Step 1: Create a New Android Project

2. Open Android Studio.
3. Start a new Android project.
4. Choose "Empty Activity".
5. Name your activity (e.g., **`SplashActivity`**).
6. Finish the setup process.

Step 2: Design the Layout for Splash Screen

Navigate to **`res/layout/activity_splash.xml`** (you might need to create it if it doesn't automatically generate with your activity). Here's a simple layout using a **`Image View`**:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:theme="@style/Theme.AppCompat.NoActionBar">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop"
        android:src="@drawable/splash_image" />
</RelativeLayout>
```

Replace **`"@drawable/splash_image"`** with your actual image resource.

Step 3: Create `SplashActivity`

In your **`SplashActivity.java`**, set up the logic to wait for a few seconds and then switch to your main activity (e.g., **`MainActivity`**)

Java Code:

```
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import androidx.appcompat.app.AppCompatActivity;

public class SplashActivity extends AppCompatActivity {

    private static final int SPLASH_TIME_OUT = 3000; // Delay in milliseconds (3000ms = 3 seconds)

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

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                // Intent to start your main activity
                Intent i = new Intent(SplashActivity.this, MainActivity.class);
                startActivity(i);

                // Close this activity
                finish();
            }
        }, SPLASH_TIME_OUT);
    }
}
```

Step 4: Update the AndroidManifest.xml

Make sure `SplashActivity` is declared and set as the launcher activity in your `AndroidManifest.xml`:

XML Code:

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
```

```
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".SplashActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".MainActivity" />
</application>
```

Step 5: Run Your App

Run your application. You should see the splash screen displaying for a few seconds before it automatically switches to the main activity.

Make sure to replace `"MainActivity"` with the actual name of your main activity, and ensure that all resources (like images) are correctly placed in your resource directories.

2. Create table Student (roll_no, name, address, percentage). Create Applicationfor performing the following operation on the table. (Using SQLite database).

i] Insert record of 5 new student details.

ii] Show all the student details. [20 Marks]

Soln Step 1: Set Up the SQLiteOpenHelper

Create a helper class to manage database creation and version management. Navigate to the `java` directory in your project, right-click, and select **New -> Java Class**. Name this class `DatabaseHelper`.

Java Code:

```
import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;


public class DatabaseHelper extends SQLiteOpenHelper {
```

```
private static final String DATABASE_NAME = "School.db";
private static final int DATABASE_VERSION = 1;
private static final String TABLE_NAME = "Student";
private static final String COLUMN_ROLL_NO = "roll_no";
private static final String COLUMN_NAME = "name";
private static final String COLUMN_ADDRESS = "address";
private static final String COLUMN_PERCENTAGE = "percentage";

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

@Override
public void onCreate(SQLiteDatabase db) {
   String createTableStatement = "CREATE TABLE " + TABLE_NAME + " (" +
         COLUMN_ROLL_NO + " INTEGER PRIMARY KEY," +
         COLUMN_NAME + " TEXT," +
         COLUMN_ADDRESS + " TEXT," +
         COLUMN_PERCENTAGE + " REAL)";
   db.execSQL(createTableStatement);
}

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

```
    // Method to insert a student into the database
    public boolean addStudent(int rollNo, String name, String address, double percentage) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues cv = new ContentValues();

        cv.put(COLUMN_ROLL_NO, rollNo);
        cv.put(COLUMN_NAME, name);
        cv.put(COLUMN_ADDRESS, address);
        cv.put(COLUMN_PERCENTAGE, percentage);

        long insert = db.insert(TABLE_NAME, null, cv);
        return insert != -1;
    }

    // Method to get all students from the database
    public Cursor getAllStudents() {
        SQLiteDatabase db = this.getReadableDatabase();
        return db.rawQuery("SELECT * FROM " + TABLE_NAME, null);
    }
}
```

Step 2: Modify the MainActivity

Now, modify your `MainActivity` to use `DatabaseHelper` to insert records and display them.

Java Code:

```
import android.database.Cursor;
import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
```

```
    DatabaseHelper dbHelper;
    TextView displayTextView;

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

        dbHelper = new DatabaseHelper(MainActivity.this);
        displayTextView = findViewById(R.id.textViewDisplay);

        // Inserting students into database
        dbHelper.addStudent(1, "Alice", "123 Street", 88.5);
        dbHelper.addStudent(2, "Bob", "456 Lane", 92.0);
        dbHelper.addStudent(3, "Charlie", "789 Road", 85.0);
        dbHelper.addStudent(4, "David", "101 Blvd", 91.5);
        dbHelper.addStudent(5, "Eve", "202 Ave", 87.0);

        // Display all students
        Cursor cursor = dbHelper.getAllStudents();
        StringBuilder builder = new StringBuilder();
        while (cursor.moveToNext()) {
            builder.append("Roll No: ").append(cursor.getInt(0))
                    .append(", Name: ").append(cursor.getString(1))
                    .append(", Address: ").append(cursor.getString(2))
                    .append(", Percentage: ").append(cursor.getDouble(3))
                    .append("\n\n");
        }
        cursor.close();
        displayTextView.setText(builder.toString());
    }
}
```

Step 3: Update the Layout for MainActivity

Navigate to **`res/layout/activity_main.xml`** and add a **`TextView`** to display the student data:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

   <TextView
      android:id="@+id/textViewDisplay"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      android:padding="16dp"
      android:textSize="18sp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Slip 2

1. **Create an application that allows the user to enter a number in the textbox. Check whether the number in the textbox is perfect number or not. Print the message using Toast control.**

Soln Design the User Interface

Open the `activity_main.xml` file under `res/layout/` directory. Replace its content with the following XML code to design the user interface:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

   xmlns:app="http://schemas.android.com/apk/res-auto"

   xmlns:tools="http://schemas.android.com/tools"

   android:layout_width="match_parent"
```

```
android:layout_height="match_parent"

tools:context=".MainActivity">



<EditText

   android:id="@+id/editTextNumber"

   android:layout_width="0dp"

   android:layout_height="wrap_content"

   android:hint="Enter a number"

   android:inputType="number"

   app:layout_constraintTop_toTopOf="parent"

   app:layout_constraintLeft_toLeftOf="parent"

   app:layout_constraintRight_toRightOf="parent"

   app:layout_constraintHorizontal_bias="0.5"

   app:layout_constraintVertical_bias="0.3"/>



<Button

   android:id="@+id/checkButton"

   android:layout_width="wrap_content"

   android:layout_height="wrap_content"

   android:text="Check Perfect Number"

   app:layout_constraintTop_toBottomOf="@+id/editTextNumber"

   app:layout_constraintLeft_toLeftOf="parent"

   app:layout_constraintRight_toRightOf="parent"
```

```
    app:layout_constraintHorizontal_bias="0.5"

    app:layout_constraintVertical_bias="0.1"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Add Logic in `MainActivity`

Now, add the logic to **`MainActivity.java`** to check if the number is perfect and display the result using a **`Toast`**.

Java Code:

```
import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;



public class MainActivity extends AppCompatActivity {



  EditText editTextNumber;

  Button checkButton;



  @Override

  protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);
```

```
    editTextNumber = findViewById(R.id.editTextNumber);
    checkButton = findViewById(R.id.checkButton);

    checkButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (editTextNumber.getText().toString().isEmpty()) {
                Toast.makeText(MainActivity.this, "Please enter a number", Toast.LENGTH_SHORT).show();
                return;
            }

            int number = Integer.parseInt(editTextNumber.getText().toString());
            if (isPerfectNumber(number)) {
                Toast.makeText(MainActivity.this, number + " is a Perfect Number",
Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(MainActivity.this, number + " is not a Perfect Number",
Toast.LENGTH_LONG).show();
            }
        }
    });
  }

  private boolean isPerfectNumber(int number) {
```

```
        int sum = 0;
        for (int i = 1; i <= number / 2; i++) {
            if (number % i == 0) {
                sum += i;
            }
        }
        return sum == number && number != 0;
    }
}
```

Update the Manifest

Ensure that your `AndroidManifest.xml` has the correct settings:

XML Code:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="your.package.name">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppName">
        <activity android:name=".MainActivity">
            <intent-filter>
```

```
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />

      </intent-filter>

    </activity>

  </application>

</manifest>
```

2. Java Android Program to perform all arithmetic Operations using Calculator.

Soln activitymain.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <RelativeLayout
        android:layout_width="368dp"
        android:layout_height="495dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <Button
            android:id="@+id/btn_1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true"
            android:layout_below="@+id/edText1"
```

```
    android:layout_marginTop="60dp"
    android:onClick="PressOne"
    android:text="1"
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_0"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/btn_8"
    android:layout_toEndOf="@+id/btn_7"
    android:layout_toRightOf="@+id/btn_7"
    android:text="0"
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_9"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/btn_6"
    android:layout_toEndOf="@+id/btn_5"
    android:layout_toRightOf="@+id/btn_5"
    android:text="9"
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_8"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/btn_5"
    android:layout_toEndOf="@+id/btn_7"
    android:layout_toRightOf="@+id/btn_7"
    android:text="8"
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/btn_4"
```

```
    android:layout_alignStart="@+id/btn_4"
    android:layout_below="@+id/btn_4"
    android:text="7"
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/btn_5"
    android:layout_alignBottom="@+id/btn_5"
    android:layout_toEndOf="@+id/btn_5"
    android:layout_toRightOf="@+id/btn_5"
    android:text="6"
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/btn_2"
    android:layout_toEndOf="@+id/btn_4"
    android:layout_toRightOf="@+id/btn_4"
    android:text="5"
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/btn_1"
    android:layout_alignStart="@+id/btn_1"
    android:layout_below="@+id/btn_1"
    android:text="4"
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:layout_alignBaseline="@+id/btn_2"
    android:layout_alignBottom="@+id/btn_2"
    android:layout_toEndOf="@+id/btn_2"
    android:layout_toRightOf="@+id/btn_2"
    android:text="3"
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/btn_1"
    android:layout_alignBottom="@+id/btn_1"
    android:layout_toEndOf="@+id/btn_1"
    android:layout_toRightOf="@+id/btn_1"
    android:text="2"
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_Add"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/btn_6"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:backgroundTint="@android:color/darker_gray"
    android:text="+"
    android:textColor="@android:color/background_light"
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_Sub"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/btn_Add"
    android:layout_alignStart="@+id/btn_Add"
    android:layout_below="@+id/btn_Add"
    android:backgroundTint="@android:color/darker_gray"
    android:text="-"
    android:textColor="@android:color/background_light"
```

```
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_Mul"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/btn_Sub"
    android:layout_alignStart="@+id/btn_Sub"
    android:layout_below="@+id/btn_6"
    android:backgroundTint="@android:color/darker_gray"
    android:text="*"
    android:textColor="@android:color/background_light"
    android:textSize="18sp" />

<Button
    android:id="@+id/btn_Div"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/btn_Mul"
    android:layout_alignStart="@+id/btn_Mul"
    android:layout_below="@+id/btn_9"
    android:backgroundTint="@android:color/darker_gray"
    android:text="/"
    android:textColor="@android:color/background_light"
    android:textSize="18sp" />

<EditText
    android:id="@+id/edText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:layout_marginTop="22dp"
    android:ems="10"
    android:inputType="textPersonName"
    android:textAlignment="textEnd"
    android:textSize="24sp" />
```

```
        <Button
            android:id="@+id/btn_calc"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/btn_0"
            android:layout_toEndOf="@+id/btn_0"
            android:layout_toRightOf="@+id/btn_0"
            android:backgroundTint="@android:color/holo_green_light"
            android:text="="
            android:textColor="@android:color/background_light"
            android:textSize="18sp" />

        <Button
            android:id="@+id/btn_dec"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/btn_7"
            android:layout_toLeftOf="@+id/btn_8"
            android:layout_toStartOf="@+id/btn_8"
            android:text="."
            android:textSize="18sp" />

        <Button
            android:id="@+id/btn_clear"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"
            android:layout_below="@+id/btn_Div"
            android:backgroundTint="@android:color/holo_blue_dark"
            android:text="clear"
            android:textColor="@android:color/background_light"
            android:textSize="18sp" />

    </RelativeLayout>

</android.support.constraint.ConstraintLayout>
```

Mainactivity.java:

```
package com.example.user.calculator;


import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;


public class MainActivity extends AppCompatActivity {

    Button
btn_1,btn_2,btn_3,btn_4,btn_5,btn_6,btn_7,btn_8,btn_9,btn_0,btn_Add,btn_Sub
,btn_Mul,btn_Div,btn_calc,btn_dec,btn_clear;

    EditText ed1;


    float Value1, Value2;

    boolean mAddition, mSubtract, mMultiplication, mDivision ;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        btn_0 = (Button) findViewById(R.id.btn_0);

        btn_1 = (Button) findViewById(R.id.btn_1);

        btn_2 = (Button) findViewById(R.id.btn_2);

        btn_3 = (Button) findViewById(R.id.btn_3);

        btn_4 = (Button) findViewById(R.id.btn_4);

        btn_5 = (Button) findViewById(R.id.btn_5);

        btn_6 = (Button) findViewById(R.id.btn_6);
```

```
btn_7 = (Button) findViewById(R.id.btn_7);
btn_8 = (Button) findViewById(R.id.btn_8);
btn_9 = (Button) findViewById(R.id.btn_9);
btn_Add = (Button) findViewById(R.id.btn_Add);
btn_Div = (Button) findViewById(R.id.btn_Div);
btn_Sub = (Button) findViewById(R.id.btn_Sub);
btn_Mul = (Button) findViewById(R.id.btn_Mul);
btn_calc = (Button) findViewById(R.id.btn_calc);
btn_dec = (Button) findViewById(R.id.btn_dec);
btn_clear = (Button) findViewById(R.id.btn_clear);
ed1 = (EditText) findViewById(R.id.edText1);

btn_0.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ed1.setText(ed1.getText()+"0");
    }
});

btn_1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ed1.setText(ed1.getText()+"1");
    }
});
```

```
btn_2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ed1.setText(ed1.getText()+"2");
    }
});

btn_3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ed1.setText(ed1.getText()+"3");
    }
});

btn_4.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ed1.setText(ed1.getText()+"4");
    }
});

btn_5.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ed1.setText(ed1.getText()+"5");
    }
```

```
});


btn_6.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ed1.setText(ed1.getText()+"6");
    }
});


btn_7.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ed1.setText(ed1.getText()+"7");
    }
});


btn_8.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ed1.setText(ed1.getText()+"8");
    }
});


btn_9.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
        ed1.setText(ed1.getText()+"9");
    }
});

btn_dec.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ed1.setText(ed1.getText()+".");
    }
});

btn_Add.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        if (ed1 == null){
            ed1.setText("");
        }else {
            Value1 = Float.parseFloat(ed1.getText() + "");
            mAddition = true;
            ed1.setText(null);
        }
    }
});

btn_Sub.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
    public void onClick(View v) {
        Value1 = Float.parseFloat(ed1.getText() + "");
        mSubtract = true ;
        ed1.setText(null);
    }
});


btn_Mul.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Value1 = Float.parseFloat(ed1.getText() + "");
        mMultiplication = true ;
        ed1.setText(null);
    }
});


btn_Div.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Value1 = Float.parseFloat(ed1.getText()+"");
        mDivision = true ;
        ed1.setText(null);
    }
});
```

```
btn_calc.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Value2 = Float.parseFloat(ed1.getText() + "");

        if (mAddition == true){

            ed1.setText(Value1 + Value2 +"");
            mAddition=false;
        }

        if (mSubtract == true){
            ed1.setText(Value1 - Value2 +"");
            mSubtract=false;
        }

        if (mMultiplication == true){
            ed1.setText(Value1 * Value2 + "");
            mMultiplication=false;
        }

        if (mDivision == true){
            ed1.setText(Value1 / Value2+"");
            mDivision=false;
        }
```

```
            }
        });


        btn_clear.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ed1.setText("");
            }
        });
    }


    public void PressOne(View view) {
    }
}
```

Slip 3

1. **Create an application that allows the user to enter a number in the textbox. Check whether the number in the textbox is Armstrong or not. Print the message accordingly in the label control.**

Soln Step 1: Create a New Android Project

7. Open Android Studio.
8. Click on "Start a new Android Studio project".
9. Choose "Empty Activity".
10. Name your activity (e.g., **`ArmstrongNumberActivity`**).
11. Set the language to Java.
12. Complete the setup and wait for the project to build.

Step 2: Design the User Interface

Update the **`activity_main.xml`** file under the **`res/layout/`** directory. Replace its content with the following XML code to create the layout:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editTextNumber"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Enter a number"
        android:inputType="number"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintVertical_bias="0.2"/>

    <Button
        android:id="@+id/checkButton"
        android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"

    android:text="Check Armstrong"

    app:layout_constraintTop_toBottomOf="@+id/editTextNumber"

    app:layout_constraintLeft_toLeftOf="parent"

    app:layout_constraintRight_toRightOf="parent"

    app:layout_constraintHorizontal_bias="0.5"

    app:layout_constraintVertical_bias="0.1"/>



  <TextView

    android:id="@+id/textViewResult"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:textSize="18sp"

    app:layout_constraintTop_toBottomOf="@+id/checkButton"

    app:layout_constraintLeft_toLeftOf="parent"

    app:layout_constraintRight_toRightOf="parent"

    app:layout_constraintHorizontal_bias="0.5"/>



</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implementing the Logic in `MainActivity.java`

Add the functionality to check if the number is an Armstrong number. Replace the contents of **`MainActivity.java`** with the following code:

Java Code:

```
import android.os.Bundle;
```

```
import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {


    EditText editTextNumber;

    Button checkButton;

    TextView textViewResult;


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);


        editTextNumber = findViewById(R.id.editTextNumber);

        checkButton = findViewById(R.id.checkButton);

        textViewResult = findViewById(R.id.textViewResult);


        checkButton.setOnClickListener(new View.OnClickListener() {

            @Override
```

```
        public void onClick(View v) {
            if (editTextNumber.getText().toString().isEmpty()) {
                textViewResult.setText("Please enter a number");
                return;
            }

            int number = Integer.parseInt(editTextNumber.getText().toString());
            if (isArmstrong(number)) {
                textViewResult.setText(number + " is an Armstrong number");
            } else {
                textViewResult.setText(number + " is not an Armstrong number");
            }
        }
    });
}

private boolean isArmstrong(int number) {
    int originalNumber, remainder, result = 0, n = 0;
    originalNumber = number;

    for (;originalNumber != 0; originalNumber /= 10, ++n);

    originalNumber = number;
```

```
        for (;originalNumber != 0; originalNumber /= 10) {
            remainder = originalNumber % 10;
            result += Math.pow(remainder, n);
        }
        return result == number;
    }
}
```

Step 4: Update the Android Manifest (if necessary)

Ensure the `AndroidManifest.xml` file has the correct activity declarations:

XML Code:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="your.package.name">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.YourAppName">
        <activity android:name=".MainActivity">
            <intent-filter>
```

```
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />

        </intent-filter>

    </activity>

  </application>

</manifest>
```

2. **Create an Android application which examine a phone number entered by auser with the given format.**
   - **Area code should be one of the following: 040, 041, 050, 0400, 044**
   - **There should 6 - 8 numbers in telephone number (+ area code).**

Soln Step 1: Create a New Android Project

13. Open Android Studio.
14. Click on "Start a new Android Studio project".
15. Choose "Empty Activity".
16. Name your activity (e.g., **`PhoneNumberValidationActivity`**).
17. Set the language to Java.
18. Finish the setup and wait for the project to build.

Step 2: Design the User Interface

Update the **`activity_main.xml`** file under the **`res/layout/`** directory. Replace its content with the following XML code to create the layout for the phone number validation:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

   xmlns:app="http://schemas.android.com/apk/res-auto"

   xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"

android:layout_height="match_parent"

tools:context=".MainActivity">



<EditText

   android:id="@+id/editTextPhoneNumber"

   android:layout_width="0dp"

   android:layout_height="wrap_content"

   android:hint="Enter phone number"

   android:inputType="phone"

   app:layout_constraintTop_toTopOf="parent"

   app:layout_constraintLeft_toLeftOf="parent"

   app:layout_constraintRight_toRightOf="parent"

   app:layout_constraintHorizontal_bias="0.5"

   app:layout_constraintVertical_bias="0.2"/>



<Button

   android:id="@+id/buttonValidate"

   android:layout_width="wrap_content"

   android:layout_height="wrap_content"

   android:text="Validate"

   app:layout_constraintTop_toBottomOf="@+id/editTextPhoneNumber"

   app:layout_constraintLeft_toLeftOf="parent"
```

```
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintHorizontal_bias="0.5"/>

  <TextView
    android:id="@+id/textViewResult"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    app:layout_constraintTop_toBottomOf="@+id/buttonValidate"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintHorizontal_bias="0.5"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implementing the Logic in `MainActivity.java`

Add the functionality to validate the phone number format. Replace the contents of **`MainActivity.java`** with the following code:

Java Code:

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
```

```
public class MainActivity extends AppCompatActivity {


    EditText editTextPhoneNumber;
    Button buttonValidate;
    TextView textViewResult;


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


        editTextPhoneNumber = findViewById(R.id.editTextPhoneNumber);
        buttonValidate = findViewById(R.id.buttonValidate);
        textViewResult = findViewById(R.id.textViewResult);


        buttonValidate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String phoneNumber = editTextPhoneNumber.getText().toString();
                validatePhoneNumber(phoneNumber);
            }
        });
```

```
    }

    private void validatePhoneNumber(String phoneNumber) {
        // Define the valid area codes
        String[] validAreaCodes = {"040", "041", "050", "0400", "044"};
        boolean isValid = false;

        // Check if the phone number matches the criteria
        for (String areaCode : validAreaCodes) {
            if (phoneNumber.startsWith(areaCode) && (phoneNumber.length() >= 6 && phoneNumber.length()
<= 8)) {
                isValid = true;
                break;
            }
        }

        if (isValid) {
            textViewResult.setText("Valid phone number");
        } else {
            textViewResult.setText("Invalid phone number");
        }
    }
}
```

Step 4: Run the Application

Build and run the application on an emulator or a physical device. You can now enter a phone number, click the "Validate" button, and see the validation result displayed below the button.

Slip 4

1. **Construct image switcher using setFactory().**

Soln activity_main.xml

```
<?xml                  version="1.0"                  encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageSwitcher
        android:id="@+id/imgsw"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    </ImageSwitcher>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:text="Prev"
        android:id="@+id/pr"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:text="Next"
        android:id                    =                    "@+id/nx"/>
</RelativeLayout>
```

MainActivity.java

```
package com.example.user.imageswitcher;


import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.ImageSwitcher;

import android.widget.ImageView;

import android.widget.ViewSwitcher;


public class MainActivity extends AppCompatActivity {

private ImageSwitcher sw;

private Button pr1,nx1;


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
```

```
sw = findViewById(R.id.imgsw);
pr1 = findViewById(R.id.pr);
nx1 = findViewById(R.id.nx);
sw.setFactory(new ViewSwitcher.ViewFactory() {
    @Override
    public View makeView() {
        ImageView imageView = new ImageView
                (getApplicationContext());
        imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);

        return imageView;
    }
});
pr1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        sw.setImageResource(R.drawable.im1);

    }
});
nx1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
            sw.setImageResource(R.drawable.im2);

        }

    });

    }

}
```

2. Write a program to search a specific location on Google Map.

Soln Step 1: Set Up Google Maps in Your Android Project

19. **Create a new Android project** in Android Studio.
20. Choose "Google Maps Activity" from the activity template list when prompted.
21. This will automatically add necessary dependencies for Google Maps in your **`build.gradle`** file.

Step 2: Obtain a Google Maps API Key

22. Go to the Google Cloud Console.
23. Create a new project or select an existing one.
24. Navigate to "APIs & Services" > "Credentials".
25. Click on “Create Credentials” and select “API key”.
26. Once the API key is created, you may need to restrict it. Google provides an option to restrict the key's usage to Android apps and specify package names and SHA-1 certificate fingerprints.
27. Copy the API key and paste it into your project's **`AndroidManifest.xml`** within the **`<application>`** tag:

XML Code:

```
<meta-data

   android:name="com.google.android.geo.API_KEY"

   android:value="YOUR_API_KEY"/>
```

Step 3: Modify Layout and Java Code

Update the layout and Java code to include a search functionality:

28. **Modify the `activity_maps.xml` layout file** to include an **`EditText`** for input and a **`Button`** to perform the search:

Xml Code:

```
<RelativeLayout

   xmlns:android="http://schemas.android.com/apk/res/android"

   xmlns:tools="http://schemas.android.com/tools"

   android:layout_width="match_parent"

   android:layout_height="match_parent"
```

```
    tools:context=".MapsActivity">

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

    <EditText
        android:id="@+id/locationSearch"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:hint="Enter location"/>

    <Button
        android:id="@+id/searchButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Search"
        android:layout_below="@+id/locationSearch"/>
</RelativeLayout>
```

Modify `MapsActivity.java` to handle the search functionality:

JAVA Code:

```
import android.location.Geocoder;
import android.location.Address;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```

```
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.CameraUpdateFactory;

import java.util.List;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

   private GoogleMap mMap;
   private EditText locationSearch;
   private Button searchButton;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_maps);
      locationSearch = findViewById(R.id.locationSearch);
      searchButton = findViewById(R.id.searchButton);

      SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
      mapFragment.getMapAsync(this);

      searchButton.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
```

```
            String location = locationSearch.getText().toString();
            List<Address> addressList = null;

            if (!location.isEmpty()) {
                Geocoder geocoder = new Geocoder(MapsActivity.this);
                try {
                    addressList = geocoder.getFromLocationName(location, 1);

                    if (addressList != null && !addressList.isEmpty()) {
                        Address address = addressList.get(0);
                        LatLng latLng = new LatLng(address.getLatitude(), address.getLongitude());
                        mMap.addMarker(new MarkerOptions().position(latLng).title("Marker in " + location));
                        mMap.animateCamera(CameraUpdateFactory.newLatLng(latLng));
                    } else {
                        Toast.makeText(MapsActivity.this, "Location not found", Toast.LENGTH_SHORT).show();
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    });
}

@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
}
}
```

Add Permissions

Ensure to add the necessary permissions in your **AndroidManifest.xml**:

XML Code:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

<uses-permission android:name="android.permission.INTERNET"/>
```

Run Your Application

Build and run your application. Now, you can enter a location name in the EditText field, press the "Search" button, and the map will move to the searched location with a marker indicating the specific spot.

Slip 5

1. **Java Android Program to Demonstrate Alert Dialog Box.**

Soln activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:text="Close app"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Mainactivity.java:

```
package com.example.alert;



import androidx.appcompat.app.AlertDialog;
```

```
import androidx.appcompat.app.AppCompatActivity;


import android.content.DialogInterface;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;


public class MainActivity extends AppCompatActivity {

    Button closeButton;

    AlertDialog.Builder builder;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        closeButton = (Button) findViewById(R.id.button);

        builder = new AlertDialog.Builder(this);

        closeButton.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View v) {


                //Uncomment the below code to Set the message and title
from the strings.xml file

                builder.setMessage(R.string.dialog_message)

                        .setTitle(R.string.dialog_title);


                //Setting message manually and performing action on button
click
```

```
                builder.setMessage("Do you want to close this application
?")
                        .setCancelable(false)
                        .setPositiveButton("Yes",
                                new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface
dialog, int id) {
                                        finish();

Toast.makeText(getApplicationContext(),"you choose yes action for
alertbox",

Toast.LENGTH_SHORT).show();
                                    }
                                })
                        .setNegativeButton("No", new
DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int
id) {
                                //  Action for 'NO' Button
                                dialog.cancel();
                                Toast.makeText(getApplicationContext(),"you
choose no action for alertbox",
                                        Toast.LENGTH_SHORT).show();
                            }
                        });
                //Creating dialog box
                AlertDialog alert = builder.create();
                //Setting the title manually
                alert.setTitle("AlertDialog");
                alert.show();
            }
```

```
        });
    }
}
```

2. **Create an Android application which will ask the user to input his / her name. A message should display the two items concatenated in a label. Change the format of the label using radio buttons and check boxes for selection. The user can make the label text bold, underlined or italic as well as change its color. Also include buttons to display the message in the label, clear the text boxes as well as label. Finally exit.**

Ans Step 1: Create a New Android Project

29. Open Android Studio.
30. Click on "Start a new Android Studio project".
31. Choose "Empty Activity".
32. Name your activity (e.g., **`TextFormatterActivity`**).
33. Set the language to Java.
34. Finish the setup and wait for the project to build.

Step 2: Design the User Interface

Update the **`activity_main.xml`** file under the **`res/layout/`** directory. Replace its content with the following XML code to create the layout:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

   xmlns:app="http://schemas.android.com/apk/res-auto"

   xmlns:tools="http://schemas.android.com/tools"

   android:layout_width="match_parent"

   android:layout_height="match_parent"

   tools:context=".MainActivity">



   <EditText

      android:id="@+id/editTextName"
```

```
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Enter your name"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"/>

<Button
    android:id="@+id/buttonDisplay"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Display"
    app:layout_constraintTop_toBottomOf="@id/editTextName"
    app:layout_constraintStart_toStartOf="parent"
    android:layout_marginTop="8dp"/>

<Button
    android:id="@+id/buttonClear"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Clear"
    app:layout_constraintTop_toBottomOf="@id/editTextName"
```

```
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="8dp"/>

<CheckBox
    android:id="@+id/checkBoxBold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Bold"
    app:layout_constraintTop_toBottomOf="@id/buttonDisplay"
    app:layout_constraintStart_toStartOf="parent"
    android:layout_marginTop="8dp"/>

<CheckBox
    android:id="@+id/checkBoxItalic"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Italic"
    app:layout_constraintTop_toBottomOf="@id/checkBoxBold"
    app:layout_constraintStart_toStartOf="parent"/>

<CheckBox
    android:id="@+id/checkBoxUnderline"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Underline"
```

```
    app:layout_constraintTop_toBottomOf="@id/checkBoxItalic"
    app:layout_constraintStart_toStartOf="parent"/>

  <TextView
    android:id="@+id/textViewDisplay"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Hello, User!"
    android:textSize="24sp"
    app:layout_constraintTop_toBottomOf="@id/checkBoxUnderline"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="16dp"/>

  <Button
    android:id="@+id/buttonExit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Exit"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Add Functionality in `MainActivity.java`

Now you will need to add Java code to handle the logic:

Java Code:

```
import android.graphics.Typeface;

import android.os.Bundle;

import android.text.Html;

import android.view.View;

import android.widget.Button;

import android.widget.CheckBox;

import android.widget.EditText;

import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {


    EditText editTextName;

    TextView textViewDisplay;

    CheckBox checkBoxBold, checkBoxItalic, checkBoxUnderline;

    Button buttonDisplay, buttonClear, buttonExit;


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
```

```
    editTextName = findViewById(R.id.editTextName);
    textViewDisplay = findViewById(R.id.textViewDisplay);
    checkBoxBold = findViewById(R.id.checkBoxBold);
    checkBoxItalic = findViewById(R.id.checkBoxItalic);
    checkBoxUnderline = findViewById(R.id.checkBoxUnderline);
    buttonDisplay = findViewById(R.id.buttonDisplay);
    buttonClear = findViewById(R.id.buttonClear);
    buttonExit = findViewById(R.id.buttonExit);

    buttonDisplay.setOnClickListener(v -> updateText());
    buttonClear.setOnClickListener(v -> {
        editTextName.setText("");
        textViewDisplay.setText("");
    });
    buttonExit.setOnClickListener(v -> finish());
}

private void updateText() {
    String name = editTextName.getText().toString();
    String baseHtml = "Hello, " + name + "!";
    if (checkBoxBold.isChecked()) {
        baseHtml = "<b>" + baseHtml + "</b>";
    }
    if (checkBoxItalic.isChecked()) {
        baseHtml = "<i>" + baseHtml + "</i>";
```

```
    }

    if (checkBoxUnderline.isChecked()) {

        baseHtml = "<u>" + baseHtml + "</u>";

    }

    textViewDisplay.setText(Html.fromHtml(baseHtml, Html.FROM_HTML_MODE_LEGACY));

  }

}
```

Step 4: Run Your Application

Compile and run your application. The user can enter their name, choose formatting options, and then display the formatted text in the TextView. The clear button resets the form, and the exit button closes the application.

Slip 6

1. **Java Android Program to demonstrate login form with validation.**

Ans **Step 1: Create a New Android Project**

35. Open Android Studio.
36. Click on "Start a new Android Studio project".
37. Choose "Empty Activity".
38. Name your activity (e.g., `LoginActivity`).
39. Set the language to Java.
40. Finish the setup and wait for the project to build.

Step 2: Design the User Interface

Open the `activity_main.xml` file under the `res/layout/` directory. Replace its content with the following XML code to create the layout for the login form:

Xml Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

   xmlns:app="http://schemas.android.com/apk/res-auto"

   xmlns:tools="http://schemas.android.com/tools"

   android:layout_width="match_parent"

   android:layout_height="match_parent"
```

```
tools:context=".MainActivity">


<EditText

   android:id="@+id/emailEditText"

   android:layout_width="0dp"

   android:layout_height="wrap_content"

   android:hint="Email"

   android:inputType="textEmailAddress"

   app:layout_constraintTop_toTopOf="parent"

   app:layout_constraintLeft_toLeftOf="parent"

   app:layout_constraintRight_toRightOf="parent"

   app:layout_constraintHorizontal_bias="0.5"

   app:layout_constraintVertical_bias="0.2"

   android:layout_margin="20dp"/>


<EditText

   android:id="@+id/passwordEditText"

   android:layout_width="0dp"

   android:layout_height="wrap_content"

   android:hint="Password"

   android:inputType="textPassword"

   app:layout_constraintTop_toBottomOf="@+id/emailEditText"

   app:layout_constraintLeft_toLeftOf="parent"

   app:layout_constraintRight_toRightOf="parent"

   app:layout_constraintHorizontal_bias="0.5"

   android:layout_margin="20dp"/>
```

```
    <Button
        android:id="@+id/loginButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Login"
        app:layout_constraintTop_toBottomOf="@+id/passwordEditText"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        android:layout_marginTop="20dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implementing the Logic in `MainActivity.java`

Add the Java code to handle the login action and validation:

Java Code:

```
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    EditText emailEditText, passwordEditText;
    Button loginButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        emailEditText = findViewById(R.id.emailEditText);
        passwordEditText = findViewById(R.id.passwordEditText);
        loginButton = findViewById(R.id.loginButton);

        loginButton.setOnClickListener(v -> {
            String email = emailEditText.getText().toString().trim();
            String password = passwordEditText.getText().toString().trim();

            if (!isValidEmail(email)) {
                Toast.makeText(MainActivity.this, "Invalid email address", Toast.LENGTH_SHORT).show();
            } else if (password.isEmpty()) {
                Toast.makeText(MainActivity.this, "Password cannot be empty", Toast.LENGTH_SHORT).show();
            } else {
                // Perform your login action here
                Toast.makeText(MainActivity.this, "Login successful", Toast.LENGTH_SHORT).show();
            }
        });
    }

    private boolean isValidEmail(String email) {
        return android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches();
    }
}
```

Step 4: Update the Android Manifest

Ensure that your **`AndroidManifest.xml`** file is properly configured to run the activity:

XML Code:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="your.package.name">

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/Theme.YourAppName">
      <activity android:name=".MainActivity">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
</manifest>
```

2. **Write a program to search a specific location on Google Map.**

Soln Slip 4 Question 2 soln

Slip 7

1. **Java Android Program to Demonstrate ProgressBar.**

Soln activity_main.xml

```
<?xml                  version="1.0"                  encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="match_parent"
        android:layout_height="14dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toTopOf="@+id/button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"                    />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="88dp"
        android:text="25%"
        app:layout_constraintBottom_toTopOf="@+id/button2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"               />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="156dp"
        android:text="100%"
```

```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.533"
        app:layout_constraintStart_toStartOf="parent"                />

</android.support.constraint.ConstraintLayout>
```

Mainactivity.java:

```
package com.example.user.progressbar;


import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.ProgressBar;

public class MainActivity extends AppCompatActivity {

    ProgressBar prg;

    Button btn1,btn2;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

            super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);                    prg   =
(ProgressBar)findViewById(R.id.progressBar);

           btn1 = (Button)findViewById(R.id.button);

           btn2 = (Button)findViewById(R.id.button2);

        btn1.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View v) {

                prg.setProgress(25);
```

```
            }
        });
        btn2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                prg.setProgress(100);
            }
        });
    }
}
```

2. **Create table Employee (E_id, name, address, ph_no). Create Application forperforming the following operation on the table. (Using SQLite database).**
   - **i] Insert record of 5 new Employees.**
   - **ii] Show all the details of Employee.**

Ans Step 1: Set Up the Android Project

41. Open Android Studio and create a new project.
42. Select "Empty Activity" and name it (e.g., **`EmployeeActivity`**).
43. Set the language to Java.
44. Finish the project setup.

Step 2: Define the SQLiteOpenHelper

Create a helper class **`DatabaseHelper`** to manage database creation and version management. Here's how you can define it:

45. Right-click on your package under **`java`** directory → New → Java Class.
46. Name the class **`DatabaseHelper`** and make it extend **`SQLiteOpenHelper`**.

Here's an implementation of **`DatabaseHelper`**:

Java Code:

```
import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;
```

```
import android.database.sqlite.SQLiteOpenHelper;


public class DatabaseHelper extends SQLiteOpenHelper {


   private static final String DATABASE_NAME = "company.db";

   private static final String TABLE_NAME = "Employee";

   private static final String COL_1 = "E_id";

   private static final String COL_2 = "name";

   private static final String COL_3 = "address";

   private static final String COL_4 = "ph_no";


   public DatabaseHelper(Context context) {

      super(context, DATABASE_NAME, null, 1);

   }


   @Override

   public void onCreate(SQLiteDatabase db) {

      db.execSQL("CREATE TABLE " + TABLE_NAME + " (E_id INTEGER PRIMARY KEY, name TEXT, address
TEXT, ph_no TEXT)");

   }


   @Override

   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

      db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);

      onCreate(db);
```

```
    }

    public boolean insertData(String e_id, String name, String address, String ph_no) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COL_1, e_id);
        contentValues.put(COL_2, name);
        contentValues.put(COL_3, address);
        contentValues.put(COL_4, ph_no);
        long result = db.insert(TABLE_NAME, null, contentValues);
        return result != -1;
    }

    public Cursor getAllData() {
        SQLiteDatabase db = this.getReadableDatabase();
        return db.rawQuery("SELECT * FROM " + TABLE_NAME, null);
    }
}
```

Step 3: Modify the MainActivity

Add the functionality in **`MainActivity.java`** to insert data into the database and display it:

Java Code:

```
import android.database.Cursor;
import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
```

```
public class MainActivity extends AppCompatActivity {


    DatabaseHelper myDb;

    TextView textView;


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);


        textView = findViewById(R.id.textViewResult);

        myDb = new DatabaseHelper(this);


        insertEmployees();

        displayEmployees();

    }


    private void insertEmployees() {

        myDb.insertData("1", "John Doe", "123 Elm St", "1234567890");

        myDb.insertData("2", "Jane Doe", "234 Elm St", "2345678901");

        myDb.insertData("3", "Jim Beam", "345 Elm St", "3456789012");

        myDb.insertData("4", "Jill Hill", "456 Elm St", "4567890123");

        myDb.insertData("5", "Jack Jill", "567 Elm St", "5678901234");
```

```
    }


    private void displayEmployees() {
        Cursor res = myDb.getAllData();
        StringBuilder buffer = new StringBuilder();
        while (res.moveToNext()) {
            buffer.append("ID :").append(res.getString(0)).append("\n");
            buffer.append("Name :").append(res.getString(1)).append("\n");
            buffer.append("Address :").append(res.getString(2)).append("\n");
            buffer.append("Phone :").append(res.getString(3)).append("\n\n");
        }
        textView.setText(buffer.toString());
    }
}
```

Step 4: Define the Layout for MainActivity

Create a layout for `MainActivity` in `res/layout/activity_main.xml`:

Xml Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

```
<TextView
    android:id="@+id/textViewResult"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Employee Details"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:padding="16dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Slip 8

1. **Create a Application which shows Life Cycle of Activity.**

Ans Step 1: Set Up the Android Project

47. Open Android Studio.
48. Click on "Start a new Android Studio project".
49. Choose "Empty Activity".
50. Name your activity (e.g., **`LifeCycleActivity`**).
51. Set the language to Java.
52. Finish the setup and wait for the project to build.

Step 2: Design the User Interface

Open the **`activity_main.xml`** file under the **`res/layout/`** directory. Replace its content with the following XML code to create the layout:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textViewStatus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Activity Status"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:padding="20dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Add Life Cycle Methods in `MainActivity.java`

Modify the **`MainActivity.java`** file to implement the life cycle methods. Here's how you can add the necessary code:

Java Code:

```
import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private TextView textViewStatus;
    private StringBuilder lifeCycleStatus;
```

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


    textViewStatus = findViewById(R.id.textViewStatus);
    lifeCycleStatus = new StringBuilder("Activity Created\n");
    updateUI();
}


@Override
protected void onStart() {
    super.onStart();
    lifeCycleStatus.append("Activity Started\n");
    updateUI();
}


@Override
protected void onResume() {
    super.onResume();
    lifeCycleStatus.append("Activity Resumed\n");
    updateUI();
}


@Override
protected void onPause() {
    super.onPause();
    lifeCycleStatus.append("Activity Paused\n");
    updateUI();
```

```
    }

    @Override
    protected void onStop() {
        super.onStop();
        lifeCycleStatus.append("Activity Stopped\n");
        updateUI();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        lifeCycleStatus.append("Activity Destroyed\n");
        updateUI();
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        lifeCycleStatus.append("Activity Restarted\n");
        updateUI();
    }

    private void updateUI() {
        textViewStatus.setText(lifeCycleStatus.toString());
    }
}
```

2. **Create table Customer (id, name, address, ph_no). Create Application forperforming the following operation on the table. (Using SQLite database).**

    i] **Insert new customer details (At least 5 records).**

ii] Show all the customer details.

Ans Step 1: Set Up the Android Project

53. Open Android Studio.
54. Click "New Project" and select "Empty Activity".
55. Name your project, for example, "CustomerDatabaseApp".
56. Choose Java as the programming language.
57. Finish creating the project.

Step 2: Define the SQLite Helper Class

Create a helper class that extends **`SQLiteOpenHelper`**. Add this class to manage database creation and version management:

58. Right-click on the **`java`** directory → New → Java Class.
59. Name the class **`DatabaseHelper`**.

Here's the code for **`DatabaseHelper`**:

Java Code:

```
import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;


public class DatabaseHelper extends SQLiteOpenHelper {


   private static final String DATABASE_NAME = "CustomerDB";

   private static final String TABLE_NAME = "Customer";

   private static final String COL_1 = "id";

   private static final String COL_2 = "name";

   private static final String COL_3 = "address";

   private static final String COL_4 = "ph_no";


   public DatabaseHelper(Context context) {
```

```
    super(context, DATABASE_NAME, null, 1);
  }

  @Override
  public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE " + TABLE_NAME + " (id INTEGER PRIMARY KEY, name TEXT, address TEXT,
ph_no TEXT)");
  }

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

  public boolean insertData(String id, String name, String address, String phone) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(COL_1, id);
    contentValues.put(COL_2, name);
    contentValues.put(COL_3, address);
    contentValues.put(COL_4, phone);
    long result = db.insert(TABLE_NAME, null, contentValues);
    return result != -1; // return true if data is inserted correctly
  }

  public Cursor getAllData() {
    SQLiteDatabase db = this.getWritableDatabase();
```

```
        return db.rawQuery("SELECT * FROM " + TABLE_NAME, null);
    }
}
```

Step 3: Design the MainActivity Layout

Modify `res/layout/activity_main.xml` to include UI elements that display customer data and a button to load data:

Xml Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btnLoadData"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Load Customer Data"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="16dp"/>

    <TextView
        android:id="@+id/textViewData"
```

```
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:textSize="16sp"
    app:layout_constraintTop_toBottomOf="@+id/btnLoadData"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:padding="16dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 4: Implement MainActivity

Implement the main activity to handle database operations and display data:

Java Code:

```
import android.database.Cursor;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

   DatabaseHelper myDb;
   TextView textViewData;
   Button btnLoadData;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    myDb = new DatabaseHelper(this);
    textViewData = findViewById(R.id.textViewData);
    btnLoadData = findViewById(R.id.btnLoadData);

    insertInitialData(); // Call this method to insert data when the app first runs
    btnLoadData.setOnClickListener(v -> loadData());
}

private void insertInitialData() {
    myDb.insertData("1", "John Doe", "123 Elm St", "1234567890");
    myDb.insertData("2", "Jane Smith", "234 Oak St", "2345678901");
    myDb.insertData("3", "Mike Johnson", "345 Maple St", "3456789012");
    myDb.insertData("4", "Sally Brown", "456 Pine St", "4567890123");
    myDb.insertData("5", "Tom Clark", "567 Birch St", "5678901234");
}

private void loadData() {
    Cursor res = myDb.getAllData();
    StringBuilder buffer = new StringBuilder();
    while (res.moveToNext()) {
        buffer.append("ID: ").append(res.getString(0)).append("\n");
        buffer.append("Name: ").append(res.getString(1)).append("\n");
        buffer.append("Address: ").append(res.getString(2)).append("\n");
        buffer.append("Phone: ").append(res.getString(3)).append("\n\n");
```

```
    }

    textViewData.setText(buffer.toString());

  }

}
```

Slip 9

1. **Create an application that allows the user to enter a number in the textbox named „getnum". Check whether the number in the textbox "getnum" is Palindrome or not. Print the message accordingly in the label when the user clicks on the button "Check".**

Soln Step 1: Create a New Android Project

60. Open Android Studio.
61. Click on "Start a new Android Studio project".
62. Choose "Empty Activity".
63. Name your activity (e.g., **`PalindromeActivity`**).
64. Set the language to Java.
65. Finish the setup and wait for the project to build.

Step 2: Design the User Interface

Open the **`activity_main.xml`** file under the **`res/layout/`** directory. Replace its content with the following XML code to create the layout for the Palindrome checker:

Xml Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity">
```

```
<EditText
   android:id="@+id/getnum"
   android:layout_width="0dp"
   android:layout_height="wrap_content"
   android:hint="Enter a number"
   android:inputType="number"
   app:layout_constraintTop_toTopOf="parent"
   app:layout_constraintLeft_toLeftOf="parent"
   app:layout_constraintRight_toRightOf="parent"
   app:layout_constraintHorizontal_bias="0.5"
   app:layout_constraintVertical_bias="0.3"/>

<Button
   android:id="@+id/checkButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Check"
   app:layout_constraintTop_toBottomOf="@id/getnum"
   app:layout_constraintLeft_toLeftOf="parent"
   app:layout_constraintRight_toRightOf="parent"
   app:layout_constraintHorizontal_bias="0.5"
   app:layout_constraintVertical_bias="0.1"/>

<TextView
   android:id="@+id/resultTextView"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintTop_toBottomOf="@id/checkButton"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintHorizontal_bias="0.5"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implementing the Logic in `MainActivity.java`

Now, add the logic to **`MainActivity.java`** to check if the number is a palindrome and display the result:

Java Code:

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

public class MainActivity extends AppCompatActivity {

    EditText editTextNumber;
    Button checkButton;
    TextView resultTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    editTextNumber = findViewById(R.id.getnum);
    checkButton = findViewById(R.id.checkButton);
    resultTextView = findViewById(R.id.resultTextView);

    checkButton.setOnClickListener(v -> {
        String num = editTextNumber.getText().toString();
        if (isPalindrome(num)) {
            resultTextView.setText(num + " is a Palindrome");
        } else {
            resultTextView.setText(num + " is not a Palindrome");
        }
    });
}

private boolean isPalindrome(String str) {
    int i = 0;
    int j = str.length() - 1;
    while (i < j) {
        if (str.charAt(i) != str.charAt(j)) {
            return false;
        }
        i++;
```

```
            j--;

          }

          return true;

       }

    }
```

2. Java android program to create simple calculator.

Soln Step 1: Create a New Android Project

66. Open Android Studio.
67. Click on "Start a new Android Studio project".
68. Choose "Empty Activity".
69. Name your project (e.g., `BasicCalculator`).
70. Set the language to Java.
71. Finish the setup and wait for the project to build.
72. **Step 2: Design the User Interface**

Update the `activity_main.xml` file under `res/layout/` directory. Use the following XML code to create the layout for the calculator with basic operation buttons:

Xml Code:

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

    <EditText
        android:id="@+id/editTextResult"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal|numberSigned"
        android:ems="10"
        android:textSize="24sp"/>

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/editTextResult">

        <TableRow>
            <Button
                android:id="@+id/buttonAdd"
                android:text="+"
```

```
            android:layout_weight="1"/>
        <Button
            android:id="@+id/buttonSubtract"
            android:text="-"
            android:layout_weight="1"/>
        <Button
            android:id="@+id/buttonMultiply"
            android:text="*"
            android:layout_weight="1"/>
        <Button
            android:id="@+id/buttonDivide"
            android:text="/"
            android:layout_weight="1"/>
    </TableRow>
    <!-- Additional rows can be added here for more functions or numbers -->
  </TableLayout>
</RelativeLayout>
```

Step 3: Implementing the Logic in `MainActivity.java`

Now, let's add the functionality to perform arithmetic operations. Open **`MainActivity.java`** and replace its content with the following code:

Java Code:

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

  EditText editTextResult;
  Button addButton, subtractButton, multiplyButton, divideButton;

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

    editTextResult = findViewById(R.id.editTextResult);
    addButton = findViewById(R.id.buttonAdd);
    subtractButton = findViewById(R.id.buttonSubtract);
```

```
    multiplyButton = findViewById(R.id.buttonMultiply);
    divideButton = findViewById(R.id.buttonDivide);

    addButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            performOperation('+');
        }
    });

    subtractButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            performOperation('-');
        }
    });

    multiplyButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            performOperation('*');
        }
    });

    divideButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            performOperation('/');
        }
    });
}

private void performOperation(char operator) {
    String[] operands = editTextResult.getText().toString().split("\\" + operator);
    if (operands.length < 2) return; // Not enough operands

    try {
        double op1 = Double.parseDouble(operands[0].trim());
        double op2 = Double.parseDouble(operands[1].trim());
        double result = 0;

        switch (operator) {
```

```
            case '+':
                result = op1 + op2;
                break;
            case '-':
                result = op1 - op2;
                break;
            case '*':
                result = op1 * op2;
                break;
            case '/':
                if (op2 != 0) result = op1 / op2;
                else throw new ArithmeticException("Cannot divide by zero");
                break;
        }

        editTextResult.setText(String.valueOf(result));
    } catch (NumberFormatException nfe) {
        editTextResult.setText("Error");
    } catch (ArithmeticException ae) {
        editTextResult.setText(ae.getMessage());
    }
  }
}
```

Slip 10

1. **Create an application that allows the user to enter a number in the textboxnamed getnum. Check whether the number in the textbox getnum is Armstrong or not. Print the message using Toast control when the user clicks on the button Check.**

Soln Step 1: Create a New Android Project

73. Open Android Studio.
74. Click on "Start a new Android Studio project".
75. Choose "Empty Activity".
76. Name your activity (e.g., `ArmstrongActivity`).
77. Set the language to Java.
78. Finish the setup and wait for the project to build.

Step 2: Design the User Interface

Update the `activity_main.xml` file under the `res/layout/` directory to create a simple user interface. Replace its content with the following XML code:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

   xmlns:app="http://schemas.android.com/apk/res-auto"

   xmlns:tools="http://schemas.android.com/tools"

   android:layout_width="match_parent"

   android:layout_height="match_parent"

   tools:context=".MainActivity">



   <EditText

      android:id="@+id/getnum"

      android:layout_width="0dp"

      android:layout_height="wrap_content"

      android:hint="Enter a number"

      android:inputType="numberDecimal"

      app:layout_constraintTop_toTopOf="parent"

      app:layout_constraintLeft_toLeftOf="parent"

      app:layout_constraintRight_toRightOf="parent"

      app:layout_constraintHorizontal_bias="0.5"

      app:layout_constraintVertical_bias="0.4"

      android:layout_margin="20dp"/>



   <Button
```

```
        android:id="@+id/checkButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Check"
        app:layout_constraintTop_toBottomOf="@id/getnum"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        android:layout_marginTop="20dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Add Logic to `MainActivity.java`

Implement the logic to determine if a number is an Armstrong number when the button is clicked:

Java Code:

```
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {


    EditText editTextNumber;
    Button checkButton;
```

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



        editTextNumber = findViewById(R.id.getnum);
        checkButton = findViewById(R.id.checkButton);



        checkButton.setOnClickListener(v -> {
            if (editTextNumber.getText().toString().isEmpty()) {
                Toast.makeText(MainActivity.this, "Please enter a number",
Toast.LENGTH_SHORT).show();
                return;
            }
            int num = Integer.parseInt(editTextNumber.getText().toString());
            if (isArmstrong(num)) {
                Toast.makeText(MainActivity.this, num + " is an Armstrong number",
Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(MainActivity.this, num + " is not an Armstrong number",
Toast.LENGTH_LONG).show();
            }
        });
    }
```

```
    private boolean isArmstrong(int number) {
        int originalNumber, remainder, result = 0;
        originalNumber = number;

        while (originalNumber != 0) {
            remainder = originalNumber % 10;
            result += Math.pow(remainder, String.valueOf(number).length());
            originalNumber /= 10;
        }

        return result == number;
    }
}
```

2. Write a program to draw GUI by using Spinner, Buttons.

Soln **Step 1: Set Up the Android Project**

79. **Open Android Studio**.
80. Click on **"Start a new Android Studio project"**.
81. Choose **"Empty Activity"**.
82. Name your activity (e.g., **`SpinnerButtonActivity`**).
83. Set the language to **Java**.
84. Finish the setup and wait for the project to build.

Step 2: Design the User Interface

Open the **`activity_main.xml`** file under the **`res/layout/`** directory. Replace its content with the following XML code to create the layout containing a **`Spinner`** and two **`Buttons`**:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Spinner
        android:id="@+id/spinnerOptions"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:entries="@array/options_array"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginTop="32dp"/>

    <Button
        android:id="@+id/buttonActionOne"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Action 1"
```

```
    app:layout_constraintTop_toBottomOf="@id/spinnerOptions"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginTop="16dp"/>

  <Button
    android:id="@+id/buttonActionTwo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Action 2"
    app:layout_constraintTop_toBottomOf="@id/buttonActionOne"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginTop="16dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Define the Spinner Options

In the **res/values/** directory, open or create the **strings.xml** file and add the following lines to define the spinner options:

XML Code:

```
<resources>
    <string name="app_name">SpinnerButtonActivity</string>
    <string-array name="options_array">
        <item>Option 1</item>
```

```
        <item>Option 2</item>
        <item>Option 3</item>
    </string-array>
</resources>
```

Step 4: Implementing the Logic in `MainActivity.java`

Modify the `MainActivity.java` file to implement the functionality:

Java Code:

```
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    Spinner spinnerOptions;
    Button buttonActionOne, buttonActionTwo;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
setContentView(R.layout.activity_main);


spinnerOptions = findViewById(R.id.spinnerOptions);
buttonActionOne = findViewById(R.id.buttonActionOne);
buttonActionTwo = findViewById(R.id.buttonActionTwo);


// Set up the spinner with the array
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
      R.array.options_array, android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinnerOptions.setAdapter(adapter);


buttonActionOne.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      performAction("Action 1", spinnerOptions.getSelectedItem().toString());
   }
});


buttonActionTwo.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      performAction("Action 2", spinnerOptions.getSelectedItem().toString());
```

```
        }
    });

    spinnerOptions.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            // Optional: Handle item selection events
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            // Optional: Handle the case where nothing is selected
        }
    });
  }

  private void performAction(String action, String option) {
    Toast.makeText(MainActivity.this, action + " performed on " + option,
Toast.LENGTH_SHORT).show();
  }
}
```

Slip 11

1. **Create an Android Application to accept two numbers to calculate its Power and Average. Create two buttons: Power and Average. Display the appropriate result on the next activity on Button click.**

Soln Step 1: Set Up the Android Project

85. Open Android Studio.
86. Click "Start a new Android Studio project".
87. Choose "Empty Activity".
88. Name the project (e.g., `PowerAndAverageApp`).
89. Set the language to Java.
90. Finish the setup and wait for the project to build.

Step 2: Design the Main Activity

First, modify `activity_main.xml` to include two `EditText` fields for input and two buttons to trigger actions:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity">



    <EditText

        android:id="@+id/editTextNumber1"

        android:layout_width="0dp"

        android:layout_height="wrap_content"

        android:inputType="numberDecimal|numberSigned"

        android:hint="Enter number 1"
```

```
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    android:layout_marginTop="100dp"/>

<EditText
    android:id="@+id/editTextNumber2"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal|numberSigned"
    android:hint="Enter number 2"
    app:layout_constraintTop_toBottomOf="@+id/editTextNumber1"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    android:layout_marginTop="16dp"/>

<Button
    android:id="@+id/buttonPower"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Power"
    app:layout_constraintTop_toBottomOf="@+id/editTextNumber2"
```

```
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintHorizontal_bias="0.3"
    android:layout_marginTop="24dp"/>

  <Button
    android:id="@+id/buttonAverage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Average"
    app:layout_constraintTop_toBottomOf="@+id/editTextNumber2"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintHorizontal_bias="0.7"
    android:layout_marginTop="24dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Create a Second Activity

Create a new Activity to display results. Right-click on your app package directory → New → Activity → Empty Activity. Name it **`ResultActivity`**.

Modify **`activity_result.xml`** to include a **`TextView`** for displaying the result:

XML Code:

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

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ResultActivity">

    <TextView
        android:id="@+id/textViewResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintVertical_bias="0.3"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 4: Implement Logic in MainActivity

In `MainActivity.java`, handle button clicks to calculate power or average and then display results in `ResultActivity`:

JAVA Code:

```
import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
```

```
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {


    EditText editTextNumber1, editTextNumber2;
    Button buttonPower, buttonAverage;


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


        editTextNumber1 = findViewById(R.id.editTextNumber1);
        editTextNumber2 = findViewById(R.id.editTextNumber2);
        buttonPower = findViewById(R.id.buttonPower);
        buttonAverage = findViewById(R.id.buttonAverage);


        buttonPower.setOnClickListener(v -> {
            double num1 = Double.parseDouble(editTextNumber1.getText().toString());
            double num2 = Double.parseDouble(editTextNumber2.getText().toString());
            double result = Math.pow(num1, num2);
            showResult(result, "Power");
```

```
        });


        buttonAverage.setOnClickListener(v -> {
            double num1 = Double.parseDouble(editTextNumber1.getText().toString());
            double num2 = Double.parseDouble(editTextNumber2.getText().toString());
            double result = (num1 + num2) / 2;
            showResult(result, "Average");
        });
    }


    private void showResult(double result, String operation) {
        Intent intent = new Intent(this, ResultActivity.class);
        intent.putExtra("result", operation + " Result: " + result);
        startActivity(intent);
    }
}
```

Step 5: Display Results in ResultActivity

In `ResultActivity.java`, retrieve and display the results:

JAVA Code:

```
import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;


public class ResultActivity extends AppCompatActivity {
```

```
    TextView textViewResult;



    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_result);



        textViewResult = findViewById(R.id.textViewResult);

        String result = getIntent().getStringExtra("result");

        textViewResult.setText(result);

    }

}
```

2. Create an Android Application to perform following string operationaccording to user selection of radio button.

Soln **Step 1: Set Up the Android Project**

91. Open Android Studio.
92. Start a new Android project.
93. Select "Empty Activity".
94. Name your project (e.g., "StringOperations").
95. Choose Java as the programming language.
96. Finish the setup.

Step 2: Design the User Interface

Open `activity_main.xml` and design your UI with a `RadioGroup`, `EditText`, `Button`, and `TextView`:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editTextInput"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Enter text"
        android:inputType="text"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_margin="16dp"/>

    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@id/editTextInput"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginTop="16dp">

        <RadioButton
            android:id="@+id/radioButtonReverse"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Reverse String"/>

        <RadioButton
            android:id="@+id/radioButtonUppercase"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
        android:text="To Uppercase"/>

    <RadioButton
        android:id="@+id/radioButtonLowercase"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="To Lowercase"/>
</RadioGroup>

<Button
    android:id="@+id/buttonPerform"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Perform Operation"
    app:layout_constraintTop_toBottomOf="@id/radioGroup"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginTop="16dp"/>

<TextView
    android:id="@+id/textViewResult"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    app:layout_constraintTop_toBottomOf="@id/buttonPerform"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginTop="16dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implement the Logic in MainActivity

Now, add the functionality in `MainActivity.java` to handle the button click based on the selected radio button and perform the corresponding string operation:

JAVA Code:

```
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    EditText editTextInput;
    RadioGroup radioGroup;
    RadioButton radioButtonReverse, radioButtonUppercase, radioButtonLowercase;
    Button buttonPerform;
    TextView textViewResult;

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

        editTextInput = findViewById(R.id.editTextInput);
        radioGroup = findViewById(R.id.radioGroup);
        radioButtonReverse = findViewById(R.id.radioButtonReverse);
        radioButtonUppercase = findViewById(R.id.radioButtonUppercase);
        radioButtonLowercase = findViewById(R.id.radioButtonLowercase);
        buttonPerform = findViewById(R.id.buttonPerform);
        textViewResult = findViewById(R.id.textViewResult);

        buttonPerform.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String inputText = editTextInput.getText().toString();
                int selectedId = radioGroup.getCheckedRadioButtonId();

                if (selectedId == R.id.radioButtonReverse) {
                    textViewResult.setText(new StringBuilder(inputText).reverse().toString());
                } else if (selectedId == R.id.radioButtonUppercase) {
                    textViewResult.setText(inputText.toUpperCase());
```

```
            } else if (selectedId == R.id.radioButtonLowercase) {
                textViewResult.setText(inputText.toLowerCase());
            } else {
                textViewResult.setText("Select an operation");
            }
        }
    });
  }
}
```

Slip 12

1. **Construct an Android app that toggles a light bulb ON and OFF when theuser clicks on toggle button.**

Soln Step 1: Set Up the Android Project

97. **Open Android Studio** and start a new project.
98. Choose **Empty Activity**.
99. Name your activity (e.g., **`LightBulbActivity`**).
100. Set the language to **Java**.
101. Finish the setup and wait for the project to build.

Step 2: Add Light Bulb Images

Before designing the UI, you need to have two images of a light bulb: one representing the bulb when it is turned on, and another for when it is off. Place these images in the **`res/drawable`** folder of your project. You can name them **`light_on.png`** and **`light_off.png`**.

Step 3: Design the User Interface

Open the **`activity_main.xml`** file and update the layout to include a **`ToggleButton`** and an **`ImageView`**:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"
```

```
    tools:context=".MainActivity">

    <ToggleButton
        android:id="@+id/toggleButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textOff="OFF"
        android:textOn="ON"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp"/>

    <ImageView
        android:id="@+id/imageViewLight"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/light_off"
        android:layout_centerInParent="true"/>
</RelativeLayout>
```

Step 4: Implement the Logic in `MainActivity.java`

Now, you need to handle the toggle button's change event to update the light bulb image accordingly. Update **`MainActivity.java`**:

JAVA Code:

```
import android.os.Bundle;
import android.widget.ImageView;
```

```
import android.widget.ToggleButton;

import androidx.appcompat.app.AppCompatActivity;



public class MainActivity extends AppCompatActivity {



    ToggleButton toggleButton;

    ImageView imageViewLight;



    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);



        toggleButton = findViewById(R.id.toggleButton);

        imageViewLight = findViewById(R.id.imageViewLight);



        toggleButton.setOnCheckedChangeListener((buttonView, isChecked) -> {

            if (isChecked) {

                imageViewLight.setImageResource(R.drawable.light_on);

            } else {

                imageViewLight.setImageResource(R.drawable.light_off);

            }

        });
```

```
    }

}
```

2. **Create an Android application which will ask the user to input his / her name. A message should display the two items concatenated in a label. Change the format of the label using radio buttons and check boxes for selection. The user can make the label text bold, underlined or italic as well as change its color. Also include buttons to display the message in the label, clear the text boxes as well as label. Finally exit.**

Soln Step 1: Set Up the Android Project

102. Open Android Studio.
103. Start a new Android Studio project.
104. Choose "Empty Activity".
105. Name your project (e.g., `TextFormatter`).
106. Set the language to Java.
107. Finish the setup and wait for the project to build.

Step 2: Design the User Interface

Update the `activity_main.xml` to include various UI components:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <EditText
        android:id="@+id/editTextName"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Enter your name"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="32dp"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"/>
```

```
<TextView
   android:id="@+id/textViewDisplay"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:textSize="18sp"
   app:layout_constraintTop_toBottomOf="@id/editTextName"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintEnd_toEndOf="parent"
   android:layout_marginTop="24dp"/>

<RadioGroup
   android:id="@+id/radioGroupColor"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="horizontal"
   app:layout_constraintTop_toBottomOf="@id/textViewDisplay"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintEnd_toEndOf="parent"
   android:layout_marginTop="16dp">

   <RadioButton
      android:id="@+id/radioRed"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Red"/>

   <RadioButton
      android:id="@+id/radioGreen"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Green"/>

   <RadioButton
      android:id="@+id/radioBlue"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Blue"/>
```

```
</RadioGroup>

<CheckBox
   android:id="@+id/checkBoxBold"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Bold"
   app:layout_constraintTop_toBottomOf="@id/radioGroupColor"
   app:layout_constraintStart_toStartOf="parent"
   android:layout_marginTop="16dp"/>

<CheckBox
   android:id="@+id/checkBoxItalic"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Italic"
   app:layout_constraintTop_toBottomOf="@id/radioGroupColor"
   app:layout_constraintStart_toEndOf="@id/checkBoxBold"
   android:layout_marginTop="16dp"/>

<CheckBox
   android:id="@+id/checkBoxUnderline"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Underline"
   app:layout_constraintTop_toBottomOf="@id/radioGroupColor"
   app:layout_constraintStart_toEndOf="@id/checkBoxItalic"
   android:layout_marginTop="16dp"/>

<Button
   android:id="@+id/buttonDisplay"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Display"
   app:layout_constraintTop_toBottomOf="@id/checkBoxBold"
   app:layout_constraintStart_toStartOf="parent"
   android:layout_marginTop="24dp"/>
```

```
    <Button
        android:id="@+id/buttonClear"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Clear"
        app:layout_constraintTop_toBottomOf="@id/checkBoxBold"
        app:layout_constraintStart_toEndOf="@id/buttonDisplay"
        android:layout_marginTop="24dp"/>


    <Button
        android:id="@+id/buttonExit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Exit"
        app:layout_constraintTop_toBottomOf="@id/checkBoxBold"
        app:layout_constraintStart_toEndOf="@id/buttonClear"
        android:layout_marginTop="24dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implement the Logic in MainActivity.java

Add the necessary logic to `MainActivity.java` to handle UI interactions:

Java Code:

```
import android.graphics.Color;
import android.graphics.Typeface;
import android.os.Bundle;
import android.text.Html;
import android.text.Spannable;
import android.text.SpannableString;
import android.text.style.StyleSpan;
import android.text.style.UnderlineSpan;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
```

```
public class MainActivity extends AppCompatActivity {

    EditText editTextName;
    TextView textViewDisplay;
    RadioGroup radioGroupColor;
    RadioButton radioRed, radioGreen, radioBlue;
    CheckBox checkBoxBold, checkBoxItalic, checkBoxUnderline;
    Button buttonDisplay, buttonClear, buttonExit;

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

        editTextName = findViewById(R.id.editTextName);
        textViewDisplay = findViewById(R.id.textViewDisplay);
        radioGroupColor = findViewById(R.id.radioGroupColor);
        radioRed = findViewById(R.id.radioRed);
        radioGreen = findViewById(R.id.radioGreen);
        radioBlue = findViewById(R.id.radioBlue);
        checkBoxBold = findViewById(R.id.checkBoxBold);
        checkBoxItalic = findViewById(R.id.checkBoxItalic);
        checkBoxUnderline = findViewById(R.id.checkBoxUnderline);
        buttonDisplay = findViewById(R.id.buttonDisplay);
        buttonClear = findViewById(R.id.buttonClear);
        buttonExit = findViewById(R.id.buttonExit);

        buttonDisplay.setOnClickListener(v -> displayText());
        buttonClear.setOnClickListener(v -> {
            editTextName.setText("");
            textViewDisplay.setText("");
        });
        buttonExit.setOnClickListener(v -> finish());
    }

    private void displayText() {
        String inputText = editTextName.getText().toString();
```

```
        Spannable spanText = new SpannableString(inputText);

        if (checkBoxBold.isChecked()) {
            spanText.setSpan(new StyleSpan(Typeface.BOLD), 0, spanText.length(),
Spannable.SPAN_INCLUSIVE_INCLUSIVE);
        }
        if (checkBoxItalic.isChecked()) {
            spanText.setSpan(new StyleSpan(Typeface.ITALIC), 0, spanText.length(),
Spannable.SPAN_INCLUSIVE_INCLUSIVE);
        }
        if (checkBoxUnderline.isChecked()) {
            spanText.setSpan(new UnderlineSpan(), 0, spanText.length(), 0);
        }

        int color = Color.BLACK; // Default color
        if (radioRed.isChecked()) color = Color.RED;
        else if (radioGreen.isChecked()) color = Color.GREEN;
        else if (radioBlue.isChecked()) color = Color.BLUE;

        textViewDisplay.setTextColor(color);
        textViewDisplay.setText(spanText);
    }
}
```

Slip 13

1. **Java android program to demonstrate Registration form with validation.**

Soln **Step 1: Set Up the Android Project**

108. Open Android Studio.
109. Start a new Android Studio project.
110. Choose "Empty Activity".
111. Name your project (e.g., "RegistrationForm").
112. Choose Java as the programming language.
113. Finish the setup.

Step 2: Design the User Interface

Update the **`activity_main.xml`** file to create the layout for the registration form. Here's an XML layout that includes EditText fields for name, email, and password, as well as a submit button:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity">



    <EditText

        android:id="@+id/editTextName"

        android:layout_width="0dp"

        android:layout_height="wrap_content"

        android:hint="Name"

        android:inputType="textPersonName"

        app:layout_constraintTop_toTopOf="parent"

        app:layout_constraintLeft_toLeftOf="parent"

        app:layout_constraintRight_toRightOf="parent"

        android:layout_marginTop="32dp"

        android:layout_marginStart="32dp"

        android:layout_marginEnd="32dp"/>
```

```
<EditText
    android:id="@+id/editTextEmail"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Email"
    android:inputType="textEmailAddress"
    app:layout_constraintTop_toBottomOf="@id/editTextName"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginTop="16dp"
    android:layout_marginStart="32dp"
    android:layout_marginEnd="32dp"/>


<EditText
    android:id="@+id/editTextPassword"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Password"
    android:inputType="textPassword"
    app:layout_constraintTop_toBottomOf="@id/editTextEmail"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginTop="16dp"
```

```
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"/>

    <Button
        android:id="@+id/buttonRegister"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Register"
        app:layout_constraintTop_toBottomOf="@id/editTextPassword"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginTop="24dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implement the Logic in MainActivity.java

Implement input validation and the response to the button click in **`MainActivity.java`**:

Java Code:

```
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
```

```
public class MainActivity extends AppCompatActivity {


    EditText editTextName, editTextEmail, editTextPassword;
    Button buttonRegister;


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


        editTextName = findViewById(R.id.editTextName);
        editTextEmail = findViewById(R.id.editTextEmail);
        editTextPassword = findViewById(R.id.editTextPassword);
        buttonRegister = findViewById(R.id.buttonRegister);


        buttonRegister.setOnClickListener(v -> {
            if (validateInput()) {
                // Proceed with registration process
                Toast.makeText(MainActivity.this, "Registration Successful", Toast.LENGTH_LONG).show();
            }
        });
    }
```

```
    private boolean validateInput() {
        // Validate name
        if (editTextName.getText().toString().trim().isEmpty()) {
            Toast.makeText(this, "Please enter your name", Toast.LENGTH_SHORT).show();
            return false;
        }

        // Validate email
        if (editTextEmail.getText().toString().trim().isEmpty() ||
!android.util.Patterns.EMAIL_ADDRESS.matcher(editTextEmail.getText().toString().trim()).matches()) {
            Toast.makeText(this, "Please enter a valid email", Toast.LENGTH_SHORT).show();
            return false;
        }

        // Validate password
        if (editTextPassword.getText().toString().trim().isEmpty() ||
editTextPassword.getText().toString().length() < 6) {
            Toast.makeText(this, "Password must be at least 6 characters", Toast.LENGTH_SHORT).show();
            return false;
        }

        return true;
    }
```

}

2. **Write a Java Android Program to Demonstrate List View Activity with alloperations Such as: Insert, Delete, Search.**

Soln Step 1: Set Up the Android Project

114. Open Android Studio.
115. Start a new Android Studio project.
116. Choose "Empty Activity".
117. Name your activity (e.g., **`ListViewActivity`**).
118. Set the language to Java.
119. Finish the setup.

Step 2: Design the User Interface

Modify **`activity_main.xml`** to include a **`ListView`**, **`EditText`** for input, and several buttons for operations:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity">



    <EditText

        android:id="@+id/editTextItem"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:hint="Enter text"

        android:inputType="text"/>
```

```
<Button
    android:id="@+id/buttonAdd"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Add"
    android:layout_below="@+id/editTextItem"/>

<Button
    android:id="@+id/buttonDelete"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Delete"
    android:layout_below="@+id/editTextItem"
    android:layout_toRightOf="@+id/buttonAdd"/>

<Button
    android:id="@+id/buttonSearch"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Search"
    android:layout_below="@+id/editTextItem"
    android:layout_toRightOf="@+id/buttonDelete"/>
```

```
    <ListView
        android:id="@+id/listViewItems"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/buttonAdd"
        android:layout_marginTop="16dp"/>
</RelativeLayout>
```

Step 3: Implement the Logic in `MainActivity.java`

Here's how you can implement the logic for adding, deleting, and searching items in the **`ListView`**:

JAVA Code:

```
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {
```

```
EditText editTextItem;
Button buttonAdd, buttonDelete, buttonSearch;
ListView listViewItems;
ArrayList<String> listItems;
ArrayAdapter<String> adapter;

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

    editTextItem = findViewById(R.id.editTextItem);
    buttonAdd = findViewById(R.id.buttonAdd);
    buttonDelete = findViewById(R.id.buttonDelete);
    buttonSearch = findViewById(R.id.buttonSearch);
    listViewItems = findViewById(R.id.listViewItems);

    listItems = new ArrayList<>();
    adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, listItems);
    listViewItems.setAdapter(adapter);

    buttonAdd.setOnClickListener(v -> {
        String item = editTextItem.getText().toString();
```

```
        if (!item.isEmpty()) {
            adapter.add(item);
            editTextItem.setText("");
            adapter.notifyDataSetChanged();
        } else {
            Toast.makeText(MainActivity.this, "Please enter text to add",
Toast.LENGTH_SHORT).show();
        }
    });

    buttonDelete.setOnClickListener(v -> {
        String item = editTextItem.getText().toString();
        if (listItems.contains(item)) {
            adapter.remove(item);
            editTextItem.setText("");
            adapter.notifyDataSetChanged();
        } else {
            Toast.makeText(MainActivity.this, "Item not found", Toast.LENGTH_SHORT).show();
        }
    });

    buttonSearch.setOnClickListener(v -> {
        String searchItem = editTextItem.getText().toString();
        if (listItems.contains(searchItem)) {
```

```
            listViewItems.setSelection(listItems.indexOf(searchItem));

            Toast.makeText(MainActivity.this, "Item found: " + searchItem,
Toast.LENGTH_SHORT).show();

        } else {

            Toast.makeText(MainActivity.this, "Item not found", Toast.LENGTH_SHORT).show();

        }

    });

  }

}
```

Slip 14

1. **Construct an Android application to accept a number and calculate anddisplay Factorial of a given number in TextView.**

Soln Step 1: Create a New Android Project

120. Open Android Studio.
121. Start a new project by selecting "Empty Activity".
122. Name your project, for example, "FactorialCalculator".
123. Choose Java as the programming language.
124. Finish the setup.

Step 2: Design the User Interface

Modify the `activity_main.xml` file to include an `EditText` for input, a `Button` to trigger the calculation, and a `TextView` to display the result. Here's a simple UI layout:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

   xmlns:app="http://schemas.android.com/apk/res-auto"

   xmlns:tools="http://schemas.android.com/tools"

   android:layout_width="match_parent"
```

```
android:layout_height="match_parent"

tools:context=".MainActivity">



<EditText

   android:id="@+id/editTextNumber"

   android:layout_width="0dp"

   android:layout_height="wrap_content"

   android:hint="Enter a number"

   android:inputType="number"

   app:layout_constraintTop_toTopOf="parent"

   app:layout_constraintStart_toStartOf="parent"

   app:layout_constraintEnd_toEndOf="parent"

   app:layout_constraintHorizontal_bias="0.5"

   app:layout_constraintVertical_bias="0.3"

   android:layout_margin="20dp"/>



<Button

   android:id="@+id/buttonCalculate"

   android:layout_width="wrap_content"

   android:layout_height="wrap_content"

   android:text="Calculate Factorial"

   app:layout_constraintTop_toBottomOf="@id/editTextNumber"

   app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"/>

    <TextView
        android:id="@+id/textViewResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        app:layout_constraintTop_toBottomOf="@id/buttonCalculate"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        android:layout_marginTop="20dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implement the Logic in `MainActivity.java`

In `MainActivity.java`, write the code to handle the button click, compute the factorial, and update the `TextView`:

JAVA code:

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

```
public class MainActivity extends AppCompatActivity {


    EditText editTextNumber;
    Button buttonCalculate;
    TextView textViewResult;


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


        editTextNumber = findViewById(R.id.editTextNumber);
        buttonCalculate = findViewById(R.id.buttonCalculate);
        textViewResult = findViewById(R.id.textViewResult);


        buttonCalculate.setOnClickListener(view -> {
            String numStr = editTextNumber.getText().toString();
            if (!numStr.isEmpty()) {
                int number = Integer.parseInt(numStr);
                long factorial = calculateFactorial(number);
                textViewResult.setText(String.format("Factorial of %d is: %d", number, factorial));
            } else {
                textViewResult.setText("Please enter a number");
```

```
        }
    });
}

private long calculateFactorial(int number) {
    long result = 1;
    for (int factor = 2; factor <= number; factor++) {
        result *= factor;
    }
    return result;
}
}
```

2. **Create an Android application, which show Login Form. After clicking LOGIN button display the "Login Successful…" message if username and password is same else display "Invalid Login" message in Toast Control.**

Soln **Step 1: Set Up the Android Project**

125. Open Android Studio.
126. Start a new project by selecting "Empty Activity".
127. Name your project, for example, "SimpleLoginApp".
128. Choose Java as the programming language.
129. Finish the setup.

Step 2: Design the User Interface

Modify the `activity_main.xml` file to create the layout for the login form. Here's a simple layout with two `EditText` fields for username and password input, and a `Button` for submitting the login information:

XML Code:

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

```
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<EditText
   android:id="@+id/editTextUsername"
   android:layout_width="0dp"
   android:layout_height="wrap_content"
   android:hint="Username"
   android:inputType="textEmailAddress"
   app:layout_constraintTop_toTopOf="parent"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintHorizontal_bias="0.5"
   app:layout_constraintVertical_bias="0.4"
   android:layout_margin="20dp"/>

<EditText
   android:id="@+id/editTextPassword"
   android:layout_width="0dp"
   android:layout_height="wrap_content"
   android:hint="Password"
   android:inputType="textPassword"
   app:layout_constraintTop_toBottomOf="@id/editTextUsername"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintHorizontal_bias="0.5"
   android:layout_margin="20dp"/>

<Button
   android:id="@+id/buttonLogin"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="LOGIN"
   app:layout_constraintTop_toBottomOf="@id/editTextPassword"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintEnd_toEndOf="parent"
```

```
    app:layout_constraintHorizontal_bias="0.5"
    android:layout_marginTop="20dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implement the Logic in `MainActivity.java`

Now, write the logic to handle the login validation when the button is clicked:

JAVA Code:

```
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

  EditText editTextUsername, editTextPassword;
  Button buttonLogin;

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

    editTextUsername = findViewById(R.id.editTextUsername);
    editTextPassword = findViewById(R.id.editTextPassword);
    buttonLogin = findViewById(R.id.buttonLogin);

    buttonLogin.setOnClickListener(v -> {
      String username = editTextUsername.getText().toString();
      String password = editTextPassword.getText().toString();

      if (username.isEmpty() || password.isEmpty()) {
        Toast.makeText(MainActivity.this, "Username or password cannot be empty",
Toast.LENGTH_SHORT).show();
        return;
```

```
        }


        // Check if username and password are the same
        if (username.equals(password)) {
            Toast.makeText(MainActivity.this, "Login Successful…", Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText(MainActivity.this, "Invalid Login", Toast.LENGTH_LONG).show();
        }
      });
   }
}
```

Slip 15

1. **Construct an Android application to accept two numbers in two EditText, with four buttons as ADD, SUB, DIV and MULT and display Result using Toast Control.**

Soln Step 1: Set Up the Android Project

130. Open Android Studio.
131. Start a new Android Studio project.
132. Choose "Empty Activity".
133. Name your activity (e.g., "ArithmeticOperations").
134. Choose Java as the programming language.
135. Complete the setup.

Step 2: Design the User Interface

Update the `activity_main.xml` to include two `EditText` fields for the inputs and four buttons for the operations:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:layout_width="match_parent"

   android:layout_height="match_parent">



   <EditText

      android:id="@+id/editTextNumber1"
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:hint="Enter first number"
    android:layout_margin="16dp"/>

<EditText
    android:id="@+id/editTextNumber2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:hint="Enter second number"
    android:layout_below="@id/editTextNumber1"
    android:layout_margin="16dp"/>

<Button
    android:id="@+id/buttonAdd"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ADD"
    android:layout_below="@id/editTextNumber2"
    android:layout_marginTop="16dp"
    android:layout_marginLeft="16dp"/>
```

```
<Button
    android:id="@+id/buttonSubtract"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="SUB"
    android:layout_toRightOf="@id/buttonAdd"
    android:layout_alignTop="@id/buttonAdd"/>

<Button
    android:id="@+id/buttonMultiply"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="MULT"
    android:layout_toRightOf="@id/buttonSubtract"
    android:layout_alignTop="@id/buttonSubtract"/>

<Button
    android:id="@+id/buttonDivide"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="DIV"
    android:layout_toRightOf="@id/buttonMultiply"
```

```
    android:layout_alignTop="@id/buttonMultiply"/>

</RelativeLayout>
```

Step 3: Implement the Logic in `MainActivity.java`

Now, add the code to handle the button clicks and perform the respective arithmetic operations:

JAVA Code:

```
import android.os.Bundle;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;



public class MainActivity extends AppCompatActivity {



    EditText editTextNumber1, editTextNumber2;

    Button buttonAdd, buttonSubtract, buttonMultiply, buttonDivide;



    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);



        editTextNumber1 = findViewById(R.id.editTextNumber1);

        editTextNumber2 = findViewById(R.id.editTextNumber2);
```

```
    buttonAdd = findViewById(R.id.buttonAdd);
    buttonSubtract = findViewById(R.id.buttonSubtract);
    buttonMultiply = findViewById(R.id.buttonMultiply);
    buttonDivide = findViewById(R.id.buttonDivide);


    buttonAdd.setOnClickListener(v -> operate("add"));
    buttonSubtract.setOnClickListener(v -> operate("subtract"));
    buttonMultiply.setOnClickListener(v -> operate("multiply"));
    buttonDivide.setOnClickListener(v -> operate("divide"));
}


private void operate(String operation) {
    try {
        double num1 = Double.parseDouble(editTextNumber1.getText().toString());
        double num2 = Double.parseDouble(editTextNumber2.getText().toString());
        double result = 0;


        switch (operation) {
            case "add":
                result = num1 + num2;
                break;
            case "subtract":
                result = num1 - num2;
```

```
                break;
            case "multiply":
                result = num1 * num2;
                break;
            case "divide":
                if (num2 != 0) result = num1 / num2;
                else throw new ArithmeticException("Cannot divide by zero.");
                break;
        }
        Toast.makeText(this, "Result: " + result, Toast.LENGTH_LONG).show();
    } catch (NumberFormatException e) {
        Toast.makeText(this, "Please enter valid numbers", Toast.LENGTH_SHORT).show();
    } catch (ArithmeticException e) {
        Toast.makeText(this, e.getMessage(), Toast.LENGTH_SHORT).show();
    }
  }
}
```

2. Construct a bank app to display different menu like withdraw, deposit etc.

Soln **Step 1: Set Up the Android Project**

136. Open Android Studio.
137. Start a new project with an "Empty Activity".
138. Name your project, for example, "SimpleBankApp".
139. Choose Java as the programming language.
140. Finish the setup.

Step 2: Design the User Interface

In your project, update the `activity_main.xml` to create a layout. This layout will include buttons for the bank operations and text views to display account information:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity">


    <TextView

        android:id="@+id/textViewAccountNumber"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:text="Account Number: 123456"

        app:layout_constraintTop_toTopOf="parent"

        app:layout_constraintLeft_toLeftOf="parent"

        android:layout_marginTop="32dp"

        android:layout_marginStart="16dp"/>


    <TextView

        android:id="@+id/textViewAccountType"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"
```

```
    android:text="Account Type: Checking"
    app:layout_constraintTop_toBottomOf="@id/textViewAccountNumber"
    app:layout_constraintLeft_toLeftOf="@id/textViewAccountNumber"
    android:layout_marginTop="16dp"/>

<TextView
    android:id="@+id/textViewBalance"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Balance: $1000"
    app:layout_constraintTop_toBottomOf="@id/textViewAccountType"
    app:layout_constraintLeft_toLeftOf="@id/textViewAccountType"
    android:layout_marginTop="16dp"/>

<Button
    android:id="@+id/buttonDeposit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Deposit"
    app:layout_constraintTop_toBottomOf="@id/textViewBalance"
    app:layout_constraintLeft_toLeftOf="@id/textViewBalance"
    android:layout_marginTop="32dp"/>

<Button
```

```
    android:id="@+id/buttonWithdraw"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Withdraw"
    app:layout_constraintTop_toBottomOf="@id/textViewBalance"
    app:layout_constraintLeft_toRightOf="@id/buttonDeposit"
    android:layout_marginTop="32dp"
      android:layout_marginStart="16dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implement the Logic in `MainActivity.java`

Add simple interactions to your buttons. Here, we will just simulate the display changes:

JAVA Code:

```
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
   private TextView balanceTextView;
   private Button depositButton;
   private Button withdrawButton;
   private double balance = 1000;  // Initial balance, for demonstration purposes
   @Override
   protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        balanceTextView = findViewById(R.id.textViewBalance);
        depositButton = findViewById(R.id.buttonDeposit);
        withdrawButton = findViewById(R.id.buttonWithdraw);
        depositButton.setOnClickListener(v -> {
            balance += 100;  // Simulate a deposit of $100
            updateBalance();
        });

        withdrawButton.setOnClickListener(v -> {
            balance -= 100;  // Simulate a withdrawal of $100
            updateBalance();
        });
    }

    private void updateBalance() {
        balanceTextView.setText("Balance: $" + balance);
    }
}
```

Slip 16

1. **Create a Simple Android Application Which Send ―Hello‖ message from oneactivity to another with help of Button (Use Intent).**

Soln Step 1: Set Up the Android Project

141. Open Android Studio.
142. Start a new project by selecting "Empty Activity".
143. Name your project (e.g., "HelloMessageApp").
144. Choose Java as the programming language.
145. Complete the setup.

Step 2: Add a Second Activity

Before you start coding, add a second activity to your project:

146. Right-click on the **app** folder in the Android Studio Project view.
147. Go to **New** -> **Activity** -> **Empty Activity**.
148. Name this new activity **SecondActivity**.
149. Click **Finish**.

Android Studio will automatically update the manifest and create the necessary files.

Step 3: Design the User Interface for Both Activities

Main Activity (First Activity)

Update **activity_main.xml** to include a button:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity">



    <Button

        android:id="@+id/buttonSend"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"
```

```
    android:text="Send Hello"

    app:layout_constraintBottom_toBottomOf="parent"

    app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toTopOf="parent" />



</androidx.constraintlayout.widget.ConstraintLayout>
```

Second Activity

Update `activity_second.xml` to include a TextView to display the message:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

   xmlns:app="http://schemas.android.com/apk/res-auto"

   xmlns:tools="http://schemas.android.com/tools"

   android:layout_width="match_parent"

   android:layout_height="match_parent"

   tools:context=".SecondActivity">



   <TextView

      android:id="@+id/textViewMessage"

      android:layout_width="wrap_content"

      android:layout_height="wrap_content"

      android:text="Message will appear here"
```

```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 4: Implement the Logic in MainActivity.java

In `MainActivity.java`, handle the button click to send the message to `SecondActivity` using an Intent:

JAVA Code:

```
import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {


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


        Button buttonSend = findViewById(R.id.buttonSend);
        buttonSend.setOnClickListener(v -> {
            Intent intent = new Intent(MainActivity.this, SecondActivity.class);
            intent.putExtra("message", "Hello");
```

```
        startActivity(intent);
    });
  }
}
```

Step 5: Retrieve the Message in SecondActivity.java

In `SecondActivity.java`, get the intent and set the message in the TextView:

JAVA Code:

```
import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);
    TextView textViewMessage = findViewById(R.id.textViewMessage);
    String message = getIntent().getStringExtra("message");
    textViewMessage.setText(message);
  }
}
```

2. **Create an Android application, with two activity first activity will have an EditText and a Button where the user can enter player name and after clicking on button the entered name will be display in another Activity. Second activity has the BACK button to transition to first**

activity (Using Intent).

Soln Step 1: Set Up the Android Project

150. Open Android Studio.
151. Start a new project by selecting "Empty Activity".
152. Name your project, for example, "PlayerNameApp".
153. Choose Java as the programming language.
154. Finish the setup.

Step 2: Design the User Interface for Both Activities

First Activity (MainActivity)

Update `activity_main.xml` to include an `EditText` for the input and a `Button` to submit the name:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <EditText
        android:id="@+id/editTextPlayerName"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Enter player name"
        android:inputType="textPersonName"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="100dp"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"/>


    <Button
        android:id="@+id/buttonSubmit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
```

```
    app:layout_constraintTop_toBottomOf="@id/editTextPlayerName"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="20dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Second Activity (DisplayActivity)

Create a new activity named **`DisplayActivity`**. Update **`activity_display.xml`** to show the player's name and include a back button:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".DisplayActivity">

   <TextView
      android:id="@+id/textViewPlayerName"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="24sp"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      android:layout_marginTop="100dp"/>

   <Button
      android:id="@+id/buttonBack"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Back"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
```

```
        android:layout_marginBottom="32dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implement the Logic in Java

MainActivity.java

Implement the logic to send the player's name to the `DisplayActivity`:

JAVA Code:

```
import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {


    private EditText editTextPlayerName;
    private Button buttonSubmit;


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


        editTextPlayerName = findViewById(R.id.editTextPlayerName);
        buttonSubmit = findViewById(R.id.buttonSubmit);


        buttonSubmit.setOnClickListener(v -> {
            String playerName = editTextPlayerName.getText().toString();
            Intent intent = new Intent(MainActivity.this, DisplayActivity.class);
            intent.putExtra("PLAYER_NAME", playerName);
            startActivity(intent);
        });
    }
}
```

DisplayActivity.java

Retrieve the player's name and handle the back button:

JAVA Code:

```
import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;


public class DisplayActivity extends AppCompatActivity {


    private TextView textViewPlayerName;
    private Button buttonBack;


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


        textViewPlayerName = findViewById(R.id.textViewPlayerName);
        buttonBack = findViewById(R.id.buttonBack);


        // Retrieve the player's name sent from MainActivity
        String playerName = getIntent().getStringExtra("PLAYER_NAME");
        textViewPlayerName.setText(playerName);


        buttonBack.setOnClickListener(v -> {
            // Navigate back to the MainActivity
            Intent intent = new Intent(DisplayActivity.this, MainActivity.class);
            startActivity(intent);
        });
    }
}
```

Step 4: Update the AndroidManifest.xml

Ensure both activities are declared in the `AndroidManifest.xml`. Android Studio typically handles this automatically when you create `DisplayActivity`.

Slip 17

1. **Write an Android Program to demonstrate Activity life Cycle.**

Soln Slip 8 Question 1 solution

2. **Write a PhoneGap application to create a contact.**

   Options are:

   - **Searching for Contacts**
   - **Cloning Contacts**
   - **Removing Contacts.**

Soln Step 1: Set Up the Android Project

155. **Open Android Studio** and start a new project.
156. Choose **Empty Activity**.
157. Name your activity (e.g., `MainActivity`).
158. Make sure to select **Java** as the programming language.
159. Finish the setup.

Step 2: Design the User Interface for MainActivity

Modify `activity_main.xml` to include `EditText` for input and buttons for operations:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:layout_width="match_parent"

   android:layout_height="match_parent">



   <EditText

      android:id="@+id/editTextContactName"

      android:layout_width="match_parent"

      android:layout_height="wrap_content"

      android:hint="Enter Contact Name"

      android:inputType="textPersonName"/>



   <Button

      android:id="@+id/buttonSearch"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Search"
    android:layout_below="@id/editTextContactName"
    android:layout_alignParentStart="true"/>

  <Button
    android:id="@+id/buttonClone"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Clone"
    android:layout_below="@id/editTextContactName"
    android:layout_toRightOf="@id/buttonSearch"/>

  <Button
    android:id="@+id/buttonRemove"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Remove"
    android:layout_below="@id/editTextContactName"
    android:layout_toRightOf="@id/buttonClone"/>
</RelativeLayout>
```

Step 3: Add a Second Activity for Display

160. **Create a second activity** called **`DisplayActivity`**.
161. Update **`activity_display.xml`** to show the operation results:

XML Code:

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

```
    <TextView
        android:id="@+id/textViewResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Results show here"
        android:layout_centerInParent="true"/>
</RelativeLayout>
```

Step 4: Implement Logic in `MainActivity.java`

JAVA Code:

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    EditText editTextContactName;
    Button buttonSearch, buttonClone, buttonRemove;

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

        editTextContactName = findViewById(R.id.editTextContactName);
```

```
buttonSearch = findViewById(R.id.buttonSearch);
buttonClone = findViewById(R.id.buttonClone);
buttonRemove = findViewById(R.id.buttonRemove);

buttonSearch.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Implementation for search
        displayResult("Search Functionality not implemented");
    }
});

buttonClone.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Implementation for clone
        displayResult("Clone Functionality not implemented");
    }
});

buttonRemove.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Implementation for remove
        displayResult("Remove Functionality not implemented");
    }
});
```

```
    }

    private void displayResult(String message) {
        Intent intent = new Intent(this, DisplayActivity.class);
        intent.putExtra("result", message);
        startActivity(intent);
    }
}
```

Step 5: Implement Logic in `DisplayActivity.java`

JAVA Code:

```
import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class DisplayActivity extends AppCompatActivity {

    TextView textViewResult;

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

        textViewResult = findViewById(R.id.textViewResult);
        String result = getIntent().getStringExtra("result");
        textViewResult.setText(result);
    }
}
```

Step 6: Configure the Android Manifest

Make sure that both activities are declared in your `AndroidManifest.xml`.

Slip 18

1. **Create an Android Application that will change color of the screen and changethe font size of text view using xml.**

Soln Step 1: Set Up the Android Project

162. **Open Android Studio** and start a new project.
163. Choose **"Empty Activity"**.
164. Name your activity (e.g., `ColorAndTextSizeActivity`).
165. Choose **Java** as the programming language.
166. Finish the setup.

Step 2: Design the User Interface

Modify `activity_main.xml` to include a `TextView` for displaying text and buttons to change the background color and text size. Here's an example:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="match_parent"

    android:layout_height="match_parent">



    <TextView

        android:id="@+id/textViewSample"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:text="Sample Text"

        android:layout_centerInParent="true"

        android:textSize="18sp" />
```

```
    <Button
        android:id="@+id/buttonColor"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Change Color"
        android:layout_above="@id/textViewSample"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="20dp" />

    <Button
        android:id="@+id/buttonTextSize"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Change Text Size"
        android:layout_below="@id/textViewSample"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp" />
</RelativeLayout>
```

Step 3: Define Colors and Text Sizes in Resources

You can define multiple colors and text sizes in the `res/values/colors.xml` and `res/values/dimens.xml` files respectively. This will help you manage them centrally.

Res/values/colors.xml

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
```

```
    <color name="colorRed">#FF0000</color>
    <color name="colorBlue">#0000FF</color>
    <color name="colorGreen">#00FF00</color>
</resources>
```

Res/values/dimens.xml

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="text_size_small">12sp</dimen>
    <dimen name="text_size_medium">18sp</dimen>
    <dimen name="text_size_large">24sp</dimen>
</resources>
```

Step 4: Implement the Logic in `MainActivity.java`

In **`MainActivity.java`**, handle button clicks to change the background color and text size dynamically. Here's how you could implement it:

JAVA Code:

```
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
```

```
public class MainActivity extends AppCompatActivity {


    TextView textViewSample;
    Button buttonColor, buttonTextSize;
    RelativeLayout layout;
    boolean colorChanged = false;
    boolean textSizeChanged = false;


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


        textViewSample = findViewById(R.id.textViewSample);
        buttonColor = findViewById(R.id.buttonColor);
        buttonTextSize = findViewById(R.id.buttonTextSize);
        layout = (RelativeLayout) textViewSample.getParent();


        buttonColor.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (!colorChanged) {
                    layout.setBackgroundColor(getResources().getColor(R.color.colorBlue));
```

```
            colorChanged = true;
        } else {
            layout.setBackgroundColor(getResources().getColor(R.color.colorRed));
            colorChanged = false;
        }
      }
    });

    buttonTextSize.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        float size = textViewSample.getTextSize();
        if (!textSizeChanged) {
            textViewSample.setTextSize(getResources().getDimension(R.dimen.text_size_large));
            textSizeChanged = true;
        } else {
            textViewSample.setTextSize(getResources().getDimension(R.dimen.text_size_small));
            textSizeChanged = false;
        }
      }
    });
  }
}
```

2. **Create table Project (id, name, dept, city). Create Application to perform thefollowing**

operations. (using SQLite database)

i] Add at least 5 records.

ii] Display all the records.

Soln **Step 1: Set Up the Android Project**

167. **Open Android Studio** and start a new project.
168. Choose "Empty Activity".
169. Name your activity (e.g., **`DatabaseActivity`**).
170. Choose Java as the programming language.
171. Finish the setup.

Step 2: Add SQLite Database Helper Class

Create a new Java class named **`DatabaseHelper`** that extends **`SQLiteOpenHelper`**. This class will manage database creation and version management.

JAVA Code:

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import
android.database.sqlite.SQLiteOpenHelper;


public class DatabaseHelper extends
SQLiteOpenHelper {


   private static final String DATABASE_NAME
= "company.db";
   private static final String TABLE_NAME =
"Project";
   private static final String COL_1 = "ID";
   private static final String COL_2 = "NAME";
   private static final String COL_3 = "DEPT";
   private static final String COL_4 = "CITY";


   public DatabaseHelper(Context context) {
      super(context, DATABASE_NAME, null, 1);
   }


   @Override
```

```
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE " +
TABLE_NAME + " (ID INTEGER PRIMARY
KEY AUTOINCREMENT, NAME TEXT,
DEPT TEXT, CITY TEXT)");
    }

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

    public boolean insertData(String name, String
dept, String city) {
        SQLiteDatabase db =
this.getWritableDatabase();
        ContentValues contentValues = new
ContentValues();
        contentValues.put(COL_2, name);
        contentValues.put(COL_3, dept);
        contentValues.put(COL_4, city);
        long result = db.insert(TABLE_NAME, null,
contentValues);
        return result != -1; // return true if data is
inserted correctly
    }

    public Cursor getAllData() {
        SQLiteDatabase db =
this.getReadableDatabase();
        return db.rawQuery("SELECT * FROM "
+ TABLE_NAME, null);
    }
}
```

Step 3: Modify the MainActivity

Implement functionality in **`MainActivity.java`** to add records to the database and display them.

JAVA Code:

```
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import
androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends
AppCompatActivity {

   DatabaseHelper myDb;
   TextView resultView;
   Button btnAddData, btnViewAll;

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

      myDb = new DatabaseHelper(this);
      resultView =
findViewById(R.id.result_text);
      btnAddData =
findViewById(R.id.add_data_button);
      btnViewAll =
findViewById(R.id.view_all_button);

      addData();
      viewAll();
   }
```

```
   public void addData() {
      btnAddData.setOnClickListener(v -> {
         boolean isInserted =
myDb.insertData("Alice", "IT", "New York")
                  & myDb.insertData("Bob",
"HR", "Los Angeles")
                  & myDb.insertData("Charlie",
"Finance", "Chicago")
                  & myDb.insertData("David",
"IT", "Miami")
                  & myDb.insertData("Eve",
"HR", "Boston");
         if(isInserted)
            Toast.makeText(MainActivity.this,
"Data Inserted",
Toast.LENGTH_LONG).show();
         else
            Toast.makeText(MainActivity.this,
"Data not Inserted",
Toast.LENGTH_LONG).show();
      });
   }


   public void viewAll() {
      btnViewAll.setOnClickListener(v -> {
         Cursor res = myDb.getAllData();
         if(res.getCount() == 0) {
            Toast.makeText(MainActivity.this, "No
Data Found", Toast.LENGTH_LONG).show();
            return;
         }


         StringBuilder buffer = new
StringBuilder();
         while (res.moveToNext()) {
            buffer.append("Id
:").append(res.getString(0)).append("\n");
            buffer.append("Name
:").append(res.getString(1)).append("\n");
```

```
            buffer.append("Dept
:").append(res.getString(2)).append("\n");
            buffer.append("City
:").append(res.getString(3)).append("\n\n");
        }

        resultView.setText(buffer.toString());
    });
  }
}
```

Step 4: Update the Layout

Update `res/layout/activity_main.xml` to include buttons and a TextView for display:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLa
yout
xmlns:android="http://schemas.android.com/apk
/res/android"

xmlns:app="http://schemas.android.com/apk/res
-auto"

xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">

  <Button
    android:id="@+id/add_data_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Add Data"

app:layout_constraintTop_toTopOf="parent"

app:layout_constraintStart_toStartOf="parent"
    android:layout_marginStart="20dp"
```

```
        android:layout_marginTop="20dp"/>

    <Button
        android:id="@+id/view_all_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="View All"

app:layout_constraintTop_toTopOf="parent"

app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginEnd="20dp"
        android:layout_marginTop="20dp"/>

    <TextView
        android:id="@+id/result_text"
        android:layout_width="0dp"
        android:layout_height="0dp"

app:layout_constraintBottom_toBottomOf="par
ent"

app:layout_constraintTop_toBottomOf="@+id/a
dd_data_button"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintEnd_toEndOf="parent"
        android:layout_margin="20dp"/>

</androidx.constraintlayout.widget.Constraintlay
out>
```

Slip 19

1. **Write an Android Program to Change the Image Displayed on the Screen.**

Soln Step 1: Set Up the Android Project

172. Open Android Studio.
173. Start a new project by selecting "Empty Activity".
174. Name your project, for example, "ImageSwitcherApp".
175. Choose Java as the programming language.
176. Finish the setup.

Step 2: Add Images to Your Project

177. Prepare a few images that you want to display and switch between. Make sure they are appropriately sized and formatted for Android (common formats are PNG and JPG).
178. Place these images in the `res/drawable` folder in your Android project. You can drag and drop them into this folder from your file explorer.

Step 3: Design the User Interface

Update the `activity_main.xml` file to include an `ImageView` for displaying the images and a `Button` to trigger the image change:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity">



    <ImageView

        android:id="@+id/imageView"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:src="@drawable/image1"  <!-- Assuming image1 is a default image in your drawable
folder -->

        app:layout_constraintBottom_toBottomOf="parent"
```

```
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

  <Button
    android:id="@+id/buttonChangeImage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Change Image"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    android:layout_marginBottom="24dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 4: Implement the Logic in `MainActivity.java`

Add the logic to **`MainActivity.java`** to handle the button click and change the image displayed:

JAVA Code:

```
import android.os.Bundle;
import android.widget.Button;
import android.widget.ImageView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
```

```
    private ImageView imageView;
    private Button buttonChangeImage;
    private int[] images = {R.drawable.image1, R.drawable.image2, R.drawable.image3};
    private int currentImageIndex = 0;

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

        imageView = findViewById(R.id.imageView);
        buttonChangeImage = findViewById(R.id.buttonChangeImage);

        buttonChangeImage.setOnClickListener(v -> {
            currentImageIndex++;  // Increment the index to show the next image
            if (currentImageIndex >= images.length) {
                currentImageIndex = 0;  // Reset index to show the first image again
            }
            imageView.setImageResource(images[currentImageIndex]);  // Set the new image resource
        });
    }
}
```

2. **Construct an Android Application to create two option menu as Find Factorial and Find Sum of Digits. Accept a number and calculate Factorial and Sum of Digits of a given number by clicking Menu.**

Soln Step 1: Set Up the Android Project

179. **Open Android Studio** and start a new project.
180. Choose **"Empty Activity"**.
181. Name your project, such as **`MathOperationsApp`**.
182. Set the language to **Java**.
183. Finish the setup.

Step 2: Design the User Interface

Update **`activity_main.xml`** to include an **`EditText`** for input and a **`TextView`** to display results:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editTextInput"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Enter a number"
        android:inputType="number"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="100dp"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"/>

    <TextView
        android:id="@+id/textViewResult"
        android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:textSize="24sp"
    app:layout_constraintTop_toBottomOf="@id/editTextInput"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="50dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implement the Menu Options

Add menu items for calculating the factorial and the sum of digits by creating a new XML file in the `res/menu` folder named `menu_main.xml`:

XML Code:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item
    android:id="@+id/action_factorial"
    android:title="Find Factorial"/>
  <item
    android:id="@+id/action_sum_of_digits"
    android:title="Find Sum of Digits"/>
</menu>
```

Step 4: Code the MainActivity

In `MainActivity.java`, implement the logic to handle menu selections:

JAVA Code:

```
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

  private EditText editTextInput;
  private TextView textViewResult;
```

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


    editTextInput = findViewById(R.id.editTextInput);
    textViewResult = findViewById(R.id.textViewResult);
}


@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}


@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.action_factorial) {
        calculateFactorial();
        return true;
    } else if (id == R.id.action_sum_of_digits) {
        calculateSumOfDigits();
        return true;
    }
    return super.onOptionsItemSelected(item);
}


private void calculateFactorial() {
    try {
        int number = Integer.parseInt(editTextInput.getText().toString());
        long factorial = 1;
        for (int i = 1; i <= number; i++) {
            factorial *= i;
        }
        textViewResult.setText(String.format("Factorial: %d", factorial));
    } catch (Exception e) {
```

```
            textViewResult.setText("Invalid input!");
        }
    }

    private void calculateSumOfDigits() {
        try {
            int number = Integer.parseInt(editTextInput.getText().toString());
            int sum = 0;
            while (number != 0) {
                sum += number % 10;
                number /= 10;
            }
            textViewResult.setText(String.format("Sum of digits: %d", sum));
        } catch (Exception e) {
            textViewResult.setText("Invalid input!");
        }
    }
}
```

Slip 20

1. **Write an application to accept two numbers from the user and displays them.But Reject input if both numbers are greater than 20 and asks for two new numbers.**

Soln Step 1: Set Up the Android Project

184. **Open Android Studio** and start a new project.
185. Choose "**Empty Activity**".
186. Name your project, such as `NumberValidationApp`.
187. Choose Java as the programming language.
188. Finish the setup.

Step 2: Design the User Interface

Update `activity_main.xml` to include two `EditText` fields for input and a `Button` to submit the numbers:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<EditText
    android:id="@+id/editTextNumber1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Enter first number"
    android:inputType="number"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="100dp"
    android:layout_marginStart="32dp"
    android:layout_marginEnd="32dp"/>

<EditText
    android:id="@+id/editTextNumber2"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Enter second number"
    android:inputType="number"
```

```
    app:layout_constraintTop_toBottomOf="@id/editTextNumber1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="16dp"
    android:layout_marginStart="32dp"
    android:layout_marginEnd="32dp"/>

  <Button
    android:id="@+id/buttonSubmit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Submit"
    app:layout_constraintTop_toBottomOf="@id/editTextNumber2"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="20dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implement the Logic in `MainActivity.java`

Implement the logic to read the numbers, validate them, and provide feedback to the user:

JAVA Code:

```
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
```

```
import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {


    private EditText editTextNumber1, editTextNumber2;
    private Button buttonSubmit;


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


        editTextNumber1 = findViewById(R.id.editTextNumber1);
        editTextNumber2 = findViewById(R.id.editTextNumber2);
        buttonSubmit = findViewById(R.id.buttonSubmit);


        buttonSubmit.setOnClickListener(v -> {
            int number1 = Integer.parseInt(editTextNumber1.getText().toString());
            int number2 = Integer.parseInt(editTextNumber2.getText().toString());


            if (number1 > 20 && number2 > 20) {
```

```
            Toast.makeText(MainActivity.this, "Both numbers must not be greater than 20. Please re-enter.", Toast.LENGTH_LONG).show();

            editTextNumber1.setText("");

            editTextNumber2.setText("");

        } else {

            Toast.makeText(MainActivity.this, "Numbers accepted: " + number1 + " and " + number2, Toast.LENGTH_LONG).show();

        }

    });

  }

}
```

2. Java Android Program to send email with attachment.

Soln Step 1: Set Up the Android Project

189. **Open Android Studio** and start a new project.
190. Select "**Empty Activity**".
191. Name your project, such as `EmailWithAttachment`.
192. Set the language to **Java**.
193. Finish the setup.

Step 2: Add Permissions

To access files from the device's storage that you might want to attach to the email, you need to declare the appropriate permissions in your `AndroidManifest.xml`:

XML Code:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.emailwithattachment">

  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

  <application
    ...
    android:requestLegacyExternalStorage="true" > <!-- This is for devices running Android 10 or higher -->
    ...
  </application>
</manifest>
```

Step 3: Design the User Interface

Update **`activity_main.xml`** to include a button to send an email:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/buttonSendEmail"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send Email with Attachment"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 4: Implement the Logic in `MainActivity.java`

In **`MainActivity.java`**, implement the functionality to invoke an email client with an attachment:

JAVA Code:

```
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.FileProvider;

import java.io.File;

public class MainActivity extends AppCompatActivity {
```

```
    private Button buttonSendEmail;

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

        buttonSendEmail = findViewById(R.id.buttonSendEmail);
        buttonSendEmail.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                sendEmail();
            }
        });
    }

    private void sendEmail() {
        Intent emailIntent = new Intent(Intent.ACTION_SEND);
        emailIntent.setType("vnd.android.cursor.dir/email");
        String to[] = {"example@example.com"};
        emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Subject of Email");
        emailIntent.putExtra(Intent.EXTRA_TEXT, "Body of Email");

        // Attach a file
        File file = new File(Environment.getExternalStorageDirectory().getPath() +
"/path/to/your/file.extension");
        Uri uri = FileProvider.getUriForFile(this, "com.example.emailwithattachment.fileprovider", file);
        emailIntent.putExtra(Intent.EXTRA_STREAM, uri);
        emailIntent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

        try {
            startActivity(Intent.createChooser(emailIntent, "Send email using..."));
        } catch (android.content.ActivityNotFoundException ex) {
            // Handle case where no email app is available
        }
    }
}
```

Step 5: Setup FileProvider

You need a `FileProvider` to securely share the file with the email app. Define it in your `AndroidManifest.xml` inside the `<application>` tag:

XML Code:

```
<provider
   android:name="androidx.core.content.FileProvider"
   android:authorities="com.example.emailwithattachment.fileprovider"
   android:exported="false"
   android:grantUriPermissions="true">
   <meta-data
      android:name="android.support.FILE_PROVIDER_PATHS"
      android:resource="@xml/file_paths" />
</provider>
```

Create a `file_paths.xml` file in the `res/xml` folder:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
   <external-path
      name="external_files"
      path="."/>
</paths>
```

Slip 21

1. **Write an Android Program to demonstrate Activity life Cycle.**

Soln Slip 8 Question 1 Solution

2. **Create an Android Application that writes data to the SD Card**

Soln **Step 1: Set Up the Android Project**

194. **Open Android Studio** and start a new project.
195. Choose **"Empty Activity"**.
196. Name your project, such as `WriteToSDCardApp`.
197. Choose Java as the programming language.
198. Finish the setup.

Step 2: Configure Permissions

To write to the external storage (SD card), you need to request the necessary permissions in your `AndroidManifest.xml`:

XML Code:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.writetosdcardapp">
```

```
    <!-- Permissions to write to external storage -->
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

    <application
        ...
        android:requestLegacyExternalStorage="true"> <!-- This line is needed for devices running Android 10+ -->
        ...
    </application>
</manifest>
```

Step 3: Design the User Interface

Update **`activity_main.xml`** to include a **`Button`** for triggering the write operation and a **`TextView`** to display status:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

```
    <Button
        android:id="@+id/buttonWriteFile"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Write to SD Card"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textViewStatus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Status: Idle"
        app:layout_constraintTop_toBottomOf="@id/buttonWriteFile"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="20dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 4: Implement the Logic in `MainActivity.java`

Add the code to perform the write operation:

JAVA Code:

```
import android.Manifest;
```

```
import android.content.pm.PackageManager;

import android.os.Bundle;

import android.os.Environment;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.ActivityCompat;

import androidx.core.content.ContextCompat;



import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;



public class MainActivity extends AppCompatActivity {



    private TextView textViewStatus;

    private static final int PERMISSION_REQUEST_CODE = 100;



    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
```

```
        textViewStatus = findViewById(R.id.textViewStatus);
        Button buttonWriteFile = findViewById(R.id.buttonWriteFile);

        buttonWriteFile.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (checkPermission()) {
                    writeFile();
                } else {
                    requestPermission();
                }
            }
        });
    }

    private boolean checkPermission() {
        int write = ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE);
        return write == PackageManager.PERMISSION_GRANTED;
    }

    private void requestPermission() {
```

```
        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, PERMISSION_REQUEST_CODE);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == PERMISSION_REQUEST_CODE && grantResults.length > 0 && grantResults[0]
== PackageManager.PERMISSION_GRANTED) {
            writeFile();
        } else {
            textViewStatus.setText("Permission Denied");
        }
    }

    private void writeFile() {
        String fileName = "SampleFile.txt";
        String content = "This is a sample file content";

        File file = new File(Environment.getExternalStorageDirectory(), fileName);
        try (FileOutputStream fos = new FileOutputStream(file)) {
            fos.write(content.getBytes());
            fos.close();
            textViewStatus.setText("File written successfully: " + file.getAbsolutePath());
```

```
        } catch (IOException e) {
            textViewStatus.setText("Error writing file: " + e.getMessage());
        }
    }
}
```

Slip 22

1. **Write an Java Android Program to Change the Image on the Screen.**

Soln Step 1: Set Up the Android Project

199. **Open Android Studio** and create a new project.
200. Choose "**Empty Activity**".
201. Name your project, for example, "ImageChangerApp".
202. Ensure **Java** is selected as the programming language.
203. Finish the project setup.

Step 2: Add Images to Your Project

Before you can change images, you need to have those images in your project:

204. **Prepare your images**: Ensure you have a few images saved on your computer, ideally PNG or JPEG format.
205. **Add images to your project**:

- Navigate to the **`app -> res -> drawable`** folder in Android Studio.
- Right-click on the **`drawable`** folder, select **New -> Image Asset**.
- Upload your images here. Make sure each image has a unique and identifiable name, e.g., **`image1`**, **`image2`**.

Step 3: Design the User Interface

Update your **`activity_main.xml`** to include an **`ImageView`** for displaying the image and a **`Button`** to change the image:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/image1"  <!-- Assuming image1 is a default image in your drawable folder -->
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/buttonChangeImage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Change Image"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView"
```

```
    android:layout_marginTop="16dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 4: Implement the Logic in `MainActivity.java`

Now, add the logic to handle the button click and change the image displayed in the **`ImageView`**:

JAVA Code:

```
import android.os.Bundle;

import android.widget.Button;

import android.widget.ImageView;

import androidx.appcompat.app.AppCompatActivity;



public class MainActivity extends AppCompatActivity {



    private ImageView imageView;

    private Button buttonChangeImage;

    private int[] images = {R.drawable.image1, R.drawable.image2, R.drawable.image3};

    private int currentImageIndex = 0;



    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);



        imageView = findViewById(R.id.imageView);
```

```
        buttonChangeImage = findViewById(R.id.buttonChangeImage);



        buttonChangeImage.setOnClickListener(v -> {
            currentImageIndex++;  // Increment the index to show the next image
            if (currentImageIndex >= images.length) {
                currentImageIndex = 0;  // Reset index to show the first image again
            }
            imageView.setImageResource(images[currentImageIndex]);  // Set the new image resource
        });
    }
}
```

2. Perform following numeric operation according to user selection of radiobutton.

Soln **Step 1: Set Up the Android Project**

206. **Open Android Studio** and start a new project.
207. Choose "**Empty Activity**".
208. Name your project, for example, "NumericOperationsApp".
209. Set the language to **Java**.
210. Finish the setup.

Step 2: Design the User Interface

Update your **`activity_main.xml`** to include an **`EditText`** for input, a **`RadioGroup`** with **`RadioButton`**s for options, a **`Button`** to trigger the operation, and a **`TextView`** to display the result:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<EditText
    android:id="@+id/editTextNumber"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Enter a number"
    android:inputType="numberSigned"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="32dp"
    android:layout_marginStart="32dp"
    android:layout_marginEnd="32dp"/>

<RadioGroup
    android:id="@+id/radioGroupOptions"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintTop_toBottomOf="@id/editTextNumber"
```

```
    app:layout_constraintStart_toStartOf="parent"
    android:layout_marginTop="16dp">

    <RadioButton
        android:id="@+id/radioButtonOddEven"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Check Odd or Even" />

    <RadioButton
        android:id="@+id/radioButtonPosNeg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Check Positive or Negative" />

    <RadioButton
        android:id="@+id/radioButtonSquare"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Calculate Square" />
</RadioGroup>

<Button
```

```
        android:id="@+id/buttonCalculate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Calculate"
        app:layout_constraintTop_toBottomOf="@id/radioGroupOptions"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="24dp"/>

    <TextView
        android:id="@+id/textViewResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Result"
        app:layout_constraintTop_toBottomOf="@id/buttonCalculate"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="20dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implement the Logic in `MainActivity.java`

JAVA Code:

```
import android.os.Bundle;
import android.widget.Button;
```

```
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    EditText editTextNumber;
    RadioGroup radioGroupOptions;
    RadioButton radioButtonOddEven, radioButtonPosNeg, radioButtonSquare;
    Button buttonCalculate;
    TextView textViewResult;

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

        editTextNumber = findViewById(R.id.editTextNumber);
        radioGroupOptions = findViewById(R.id.radioGroupOptions);
        buttonCalculate = findViewById(R.id.buttonCalculate);
```

```
    textViewResult = findViewById(R.id.textViewResult);



    buttonCalculate.setOnClickListener(v -> {
        int selectedId = radioGroupOptions.getCheckedRadioButtonId();
        if (selectedId == -1) {
            textViewResult.setText("Please select an option");
            return;
        }
        int number = Integer.parseInt(editTextNumber.getText().toString());
        calculateResult(number, selectedId);
    });
}



private void calculateResult(int number, int selectedId) {
    switch (selectedId) {
        case R.id.radioButtonOddEven:
            textViewResult.setText(number % 2 == 0 ? "Even" : "Odd");
            break;
        case R.id.radioButtonPosNeg:
            textViewResult.setText(number >= 0 ? "Positive" : "Negative");
            break;
        case R.id.radioButtonSquare:
            textViewResult.setText("Square: " + (number * number));
            break;
```

```
        }
    }
}
```

Slip 23

1. **Write a Java android program to demonstrate implicit intent.**

Soln Step 1: Set Up the Android Project

211. **Open Android Studio** and start a new project.
212. Select **"Empty Activity"**.
213. Name your project, such as **`ImplicitIntentDemo`**.
214. Choose Java as the programming language.
215. Finish the setup.

Step 2: Design the User Interface

Update the **`activity_main.xml`** file to include an **`EditText`** for the user to enter a URL and a **`Button`** to open the URL:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity">



    <EditText
```

```
    android:id="@+id/editTextUrl"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Enter URL"
    android:inputType="textUri"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="100dp"
    android:layout_marginStart="32dp"
    android:layout_marginEnd="32dp"/>

  <Button
    android:id="@+id/buttonOpenUrl"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Open URL"
    app:layout_constraintTop_toBottomOf="@id/editTextUrl"
    app:layout_constraintStart_toStartOf="@id/editTextUrl"
    app:layout_constraintEnd_toEndOf="@id/editTextUrl"
    android:layout_marginTop="20dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 3: Implement the Logic in `MainActivity.java`

Add the code to handle the button click, creating an implicit intent to open the entered URL:

JAVA Code:

```
import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.widget.Button;

import android.widget.EditText;

import androidx.appcompat.app.AppCompatActivity;



public class MainActivity extends AppCompatActivity {



    private EditText editTextUrl;

    private Button buttonOpenUrl;



    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);



        editTextUrl = findViewById(R.id.editTextUrl);

        buttonOpenUrl = findViewById(R.id.buttonOpenUrl);



        buttonOpenUrl.setOnClickListener(v -> {

            String url = editTextUrl.getText().toString();
```

```
        if (!url.isEmpty()) {
            openUrl(url);
        }
    });
}

private void openUrl(String url) {
    if (!url.startsWith("http://") && !url.startsWith("https://")) {
        url = "http://" + url;
    }
    Uri webpage = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
}
```

Step 4: Add Internet Permission

To open a URL in a web browser, your app needs permission to access the internet. Add this line to your `AndroidManifest.xml`:

XML Code:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

2. **Create an Android application which will ask the user to input his / her name. A message should display the two items concatenated in a label. Change the format of the label using radio buttons and check boxes for selection. The user can make the label text bold, underlined or italic as well as change its color. Also include buttons to display the message in the label, clear the text boxes as**

well as label. Finally exit.

Soln Step 1: Set Up the Android Project

216. **Open Android Studio** and start a new project.
217. Select **"Empty Activity"**.
218. Name your project, such as **`TextFormatterApp`**.
219. Choose Java as the programming language.
220. Finish the setup.

Step 2: Design the User Interface

Update **`activity_main.xml`** to include the user interface elements such as **`EditText`**, **`TextView`**, **`RadioGroup`**, **`CheckBox`**, and **`Button`** components:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="match_parent"

    android:layout_height="match_parent">



    <EditText

        android:id="@+id/editTextName"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:hint="Enter your name"

        android:layout_margin="16dp" />



    <Button

        android:id="@+id/buttonDisplay"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"
```

```
    android:text="Display"
    android:layout_below="@id/editTextName"
    android:layout_alignParentStart="true"
    android:layout_marginTop="16dp" />

<Button
    android:id="@+id/buttonClear"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Clear"
    android:layout_below="@id/editTextName"
    android:layout_toRightOf="@id/buttonDisplay"
    android:layout_marginTop="16dp"
    android:layout_marginStart="16dp"/>

<Button
    android:id="@+id/buttonExit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Exit"
    android:layout_below="@id/editTextName"
    android:layout_toRightOf="@id/buttonClear"
    android:layout_marginTop="16dp"
    android:layout_marginStart="16dp"/>
```

```
<RadioGroup
    android:id="@+id/radioGroupColor"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/buttonDisplay"
    android:layout_marginTop="16dp">

    <RadioButton
        android:id="@+id/radioButtonRed"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Red" />

    <RadioButton
        android:id="@+id/radioButtonGreen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Green" />

    <RadioButton
        android:id="@+id/radioButtonBlue"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Blue" />
</RadioGroup>

<CheckBox
    android:id="@+id/checkBoxBold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Bold"
    android:layout_below="@id/radioGroupColor"
    android:layout_alignParentStart="true" />

<CheckBox
    android:id="@+id/checkBoxItalic"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Italic"
    android:layout_below="@id/radioGroupColor"
    android:layout_toRightOf="@id/checkBoxBold"/>

<CheckBox
    android:id="@+id/checkBoxUnderline"
    android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"

        android:text="Underline"

        android:layout_below="@id/radioGroupColor"

        android:layout_toRightOf="@id/checkBoxItalic"/>



    <TextView

        android:id="@+id/textViewDisplay"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:textSize="18sp"

        android:layout_below="@id/checkBoxBold"

        android:layout_centerHorizontal="true"

        android:layout_marginTop="20dp"/>

</RelativeLayout>
```

Step 3: Implement the Logic in `MainActivity.java`

Handle the button clicks and text formatting:

JAVA Code:

```
import android.graphics.Typeface;

import android.os.Bundle;

import android.text.Html;

import android.text.Spannable;

import android.text.SpannableString;

import android.text.style.StyleSpan;

import android.text.style.UnderlineSpan;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;

public class MainActivity extends AppCompatActivity {

    private EditText editTextName;
    private Button buttonDisplay, buttonClear, buttonExit;
    private TextView textViewDisplay;
    private RadioGroup radioGroupColor;
    private RadioButton radioButtonRed, radioButtonGreen, radioButtonBlue;
    private CheckBox checkBoxBold, checkBoxItalic, checkBoxUnderline;

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

```
    editTextName = findViewById(R.id.editTextName);
    buttonDisplay = findViewById(R.id.buttonDisplay);
    buttonClear = findViewById(R.id.buttonClear);
    buttonExit = findViewById(R.id.buttonExit);
    textViewDisplay = findViewById(R.id.textViewDisplay);
    radioGroupColor = findViewById(R.id.radioGroupColor);
    radioButtonRed = findViewById(R.id.radioButtonRed);
    radioButtonGreen = findViewById(R.id.radioButtonGreen);
    radioButtonBlue = findViewById(R.id.radioButtonBlue);
    checkBoxBold = findViewById(R.id.checkBoxBold);
    checkBoxItalic = findViewById(R.id.checkBoxItalic);
    checkBoxUnderline = findViewById(R.id.checkBoxUnderline);

    buttonDisplay.setOnClickListener(v -> displayText());
    buttonClear.setOnClickListener(v -> {
        editTextName.setText("");
        textViewDisplay.setText("");
    });
    buttonExit.setOnClickListener(v -> finish());

    setupRadioButtons();
}
```

```
    private void displayText() {
        String inputText = editTextName.getText().toString();
        Spannable spanText = new SpannableString(inputText);

        if (checkBoxBold.isChecked()) {
            spanText.setSpan(new StyleSpan(Typeface.BOLD), 0, spanText.length(),
Spannable.SPAN_INCLUSIVE_INCLUSIVE);
        }
        if (checkBoxItalic.isChecked()) {
            spanText.setSpan(new StyleSpan(Typeface.ITALIC), 0, spanText.length(),
Spannable.SPAN_INCLUSIVE_INCLUSIVE);
        }
        if (checkBoxUnderline.isChecked()) {
            spanText.setSpan(new UnderlineSpan(), 0, spanText.length(), 0);
        }

        textViewDisplay.setText(spanText);
    }

    private void setupRadioButtons() {
        radioGroupColor.setOnCheckedChangeListener((group, checkedId) -> {
            switch (checkedId) {
                case R.id.radioButtonRed:
```

```
                textViewDisplay.setTextColor(ContextCompat.getColor(this, R.color.red));
                break;
            case R.id.radioButtonGreen:
                textViewDisplay.setTextColor(ContextCompat.getColor(this, R.color.green));
                break;
            case R.id.radioButtonBlue:
                textViewDisplay.setTextColor(ContextCompat.getColor(this, R.color.blue));
                break;
        }
    });
  }
}
```

Step 4: Define Color Resources

Make sure you have defined color values in your `res/values/colors.xml`:

XML Code:

```
<resources>
    <color name="red">#FF0000</color>
    <color name="green">#00FF00</color>
    <color name="blue">#0000FF</color>
</resources>
```

Slip 24

1. **Write an application to accept a string from the user. With two buttons todisplay the string in Uppercase and Lowercase using the toast message.**

Soln Step 1: Set Up the Android Project

221. **Open Android Studio** and start a new project.
222. Choose **"Empty Activity"**.
223. Name your project, such as **`StringCaseApp`**.
224. Choose Java as the programming language.
225. Finish the setup.

Step 2: Design the User Interface

Modify **`activity_main.xml`** to include an **`EditText`** for input, and two **`Button`** views for triggering the uppercase and lowercase operations:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity">



    <EditText

        android:id="@+id/editTextInput"

        android:layout_width="0dp"

        android:layout_height="wrap_content"

        android:hint="Enter Text"

        android:inputType="text"

        app:layout_constraintTop_toTopOf="parent"

        app:layout_constraintStart_toStartOf="parent"

        app:layout_constraintEnd_toEndOf="parent"
```

```
    android:layout_marginTop="100dp"
    android:layout_marginStart="32dp"
    android:layout_marginEnd="32dp"/>

<Button
    android:id="@+id/buttonUppercase"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Uppercase"
    app:layout_constraintTop_toBottomOf="@id/editTextInput"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="20dp"/>

<Button
    android:id="@+id/buttonLowercase"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Lowercase"
    app:layout_constraintTop_toBottomOf="@id/buttonUppercase"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="16dp"/>
```

</androidx.constraintlayout.widget.ConstraintLayout>

Step 3: Implement the Logic in `MainActivity.java`

Add the code to handle the button clicks and display the string in uppercase or lowercase using a toast:

JAVA Code:

```
import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;



import androidx.appcompat.app.AppCompatActivity;



public class MainActivity extends AppCompatActivity {



    private EditText editTextInput;

    private Button buttonUppercase;

    private Button buttonLowercase;



    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
```

```
        editTextInput = findViewById(R.id.editTextInput);
        buttonUppercase = findViewById(R.id.buttonUppercase);
        buttonLowercase = findViewById(R.id.buttonLowercase);

        buttonUppercase.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String inputText = editTextInput.getText().toString();
                String upperText = inputText.toUpperCase();
                Toast.makeText(MainActivity.this, upperText, Toast.LENGTH_SHORT).show();
            }
        });

        buttonLowercase.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String inputText = editTextInput.getText().toString();
                String lowerText = inputText.toLowerCase();
                Toast.makeText(MainActivity.this, lowerText, Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

2. Create table Car (id, name, type, color). Create Java Android Application forperforming the

following operation on the table. (Using SQLite database)

i) Insert 5 New Car Details.

ii) Show All the Car Details

Soln **Step 1: Set Up the Android Project**

226. **Open Android Studio** and start a new project.
227. Choose **"Empty Activity"**.
228. Name your project, such as **`CarDatabaseApp`**.
229. Choose Java as the programming language.
230. Finish the setup.

Step 2: Add SQLite Database Helper Class

Create a new Java class named **`DatabaseHelper`** that extends **`SQLiteOpenHelper`**. This class will handle database creation and version management

JAVA Code:

```
import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;



public class DatabaseHelper extends SQLiteOpenHelper {



    private static final String DATABASE_NAME = "car.db";

    private static final String TABLE_NAME = "Car";

    private static final String COL_1 = "ID";

    private static final String COL_2 = "NAME";

    private static final String COL_3 = "TYPE";

    private static final String COL_4 = "COLOR";
```

```
    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE " + TABLE_NAME + " (ID INTEGER PRIMARY KEY
AUTOINCREMENT, NAME TEXT, TYPE TEXT, COLOR TEXT)");
    }

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

    public boolean insertData(String name, String type, String color) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COL_2, name);
        contentValues.put(COL_3, type);
        contentValues.put(COL_4, color);
        long result = db.insert(TABLE_NAME, null, contentValues);
        return result != -1; // return true if data is inserted correctly
```

```
    }


    public Cursor getAllData() {
        SQLiteDatabase db = this.getReadableDatabase();
        return db.rawQuery("SELECT * FROM " + TABLE_NAME, null);
    }
}
```

Step 3: Modify the MainActivity

Implement functionality in **`MainActivity.java`** to add records to the database and display them.

JAVA Code:

```
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;


import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {


    DatabaseHelper myDb;
    TextView resultView;
    Button btnAddData, btnViewAll;
```

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


    myDb = new DatabaseHelper(this);
    resultView = findViewById(R.id.result_text);
    btnAddData = findViewById(R.id.add_data_button);
    btnViewAll = findViewById(R.id.view_all_button);


    addData();
    viewAll();
}


public void addData() {
    btnAddData.setOnClickListener(v -> {
        boolean isInserted = myDb.insertData("Honda", "Sedan", "Black")
                & myDb.insertData("Toyota", "SUV", "White")
                & myDb.insertData("Ford", "Truck", "Blue")
                & myDb.insertData("Nissan", "Coupe", "Red")
                & myDb.insertData("Chevrolet", "Convertible", "Yellow");
        if(isInserted)
            Toast.makeText(MainActivity.this, "Data Inserted", Toast.LENGTH_LONG).show();
```

```
        else
            Toast.makeText(MainActivity.this, "Data not Inserted", Toast.LENGTH_LONG).show();
    });
}

public void viewAll() {
    btnViewAll.setOnClickListener(v -> {
        Cursor res = myDb.getAllData();
        if(res.getCount() == 0) {
            Toast.makeText(MainActivity.this, "No Data Found", Toast.LENGTH_LONG).show();
            return;
        }

        StringBuilder buffer = new StringBuilder();
        while (res.moveToNext()) {
            buffer.append("Id :").append(res.getString(0)).append("\n");
            buffer.append("Name :").append(res.getString(1)).append("\n");
            buffer.append("Type :").append(res.getString(2)).append("\n");
            buffer.append("Color :").append(res.getString(3)).append("\n\n");
        }

        resultView.setText(buffer.toString());
    });
```

```
    }
}
```

Step 4: Define Layout in `activity_main.xml`

Create buttons and text view to trigger actions and display data.

XML Code:

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

    <Button
        android:id="@+id/add_data_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Add Data"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp"/>

    <Button
        android:id="@+id/view_all_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="View All"
```

```
        android:layout_below="@id/add_data_button"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"/>

    <TextView
        android:id="@+id/result_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/view_all_button"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"/>
</RelativeLayout>
```

Slip 25

1. **Create an android application for SMS activity.**

Soln Step 1: Set Up the Android Project

231. **Open Android Studio** and start a new project.
232. Choose **"Empty Activity"**.
233. Name your project, for example, "SMSApp".
234. Set the language to **Java**.
235. Finish the setup.

Step 2: Add Permissions

To send SMS messages, your app needs to request the necessary permissions in your `AndroidManifest.xml`:

XML Code:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.smsapp">
```

```
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>

    <application
        ...
        android:label="@string/app_name">
        ...
    </application>
</manifest>
```

Step 3: Design the User Interface

Modify the `activity_main.xml` to include two `EditText` views for the phone number and message, and a `Button` to send the SMS:

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editTextPhoneNumber"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Phone Number"
        android:inputType="phone"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="64dp"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"/>

    <EditText
        android:id="@+id/editTextSMS"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Message"
        android:inputType="textMultiLine"
        app:layout_constraintTop_toBottomOf="@id/editTextPhoneNumber"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="16dp"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"/>

    <Button
        android:id="@+id/buttonSendSMS"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send SMS"
        app:layout_constraintTop_toBottomOf="@id/editTextSMS"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="24dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 4: Implement the SMS Sending Logic in `MainActivity.java`

Add the functionality to send an SMS using the `SmsManager` API:

JAVA Code:

```
import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

public class MainActivity extends AppCompatActivity {

    private EditText editTextPhoneNumber, editTextSMS;
    private Button buttonSendSMS;

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

        editTextPhoneNumber = findViewById(R.id.editTextPhoneNumber);
        editTextSMS = findViewById(R.id.editTextSMS);
```

```
    buttonSendSMS = findViewById(R.id.buttonSendSMS);


    buttonSendSMS.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (ContextCompat.checkSelfPermission(MainActivity.this,
                    Manifest.permission.SEND_SMS) != PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(MainActivity.this,
                        new String[]{Manifest.permission.SEND_SMS}, 101);
            } else {
                sendSMS();
            }
        }
    });
  }


  private void sendSMS() {
    String phoneNo = editTextPhoneNumber.getText().toString();
    String sms = editTextSMS.getText().toString();


    try {
        SmsManager smsManager = SmsManager.getDefault();
        smsManager.sendTextMessage(phoneNo, null, sms, null, null);
        Toast.makeText(getApplicationContext(), "SMS Sent Successfully!",
Toast.LENGTH_LONG).show();
    } catch (Exception e) {
        Toast.makeText(getApplicationContext(), "SMS Failed to Send, Please try again",
Toast.LENGTH_LONG).show();
    }
  }
```

```
    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == 101 && grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            sendSMS();
        } else {
            Toast.makeText(getApplicationContext(), "SMS Permission Denied",
Toast.LENGTH_LONG).show();
        }
    }
}
```

2. **Create an Android application, which show Login Form in table layout. After clicking LOGIN button display the "Login Successful…" message if username and password is same else display "Invalid Login" message in Toast Control.**

Soln Slip 14 Question 2 Solution