Index

Prog No	Problem statement	Page No
1	Creating “Hello world” Application	4
2	Creating an application that displays message based on screen orientation.	6
3	Create an application to develop Login windows using UI controls.	9
4	Create an application to implement new activities using explicit intent, implicit intent and content provider.	11
5	Create an application that displays custom designed Opening Screen.	15
6	Create an UI with all views.	17
7	Create menu in Application	
8	Read/ write the Local data.	
9	Create / Read / Write data with database (SQLite).	
10	Create an application to send SMS and receive SMS	
11	Create an application to send an e-mail.	
12	Display Map based on the Current/given location.	
13	Create a sample application with login module (check username and password) On successful login change Textview “Login Successful”. On login fail alert using Toast “login fail”	
14	Learn to deploy Android applications	

Step-by-step guide to create your first application in Android Studio

Step 1: Install Android Studio

1. Download **Android Studio** from the official site: https://developer.android.com/studio
2. Install it by following the on-screen instructions.

Step 2: Start a New Android Project

1. Open **Android Studio**.
2. Click on **"New Project"**.
3. Choose a project template: Select **Empty Activity** (simplest for beginners).
4. Click **Next**.

Step 3: Configure the Project

1. **Name** your application (for example, "MyFirstApp").
2. **Package Name** (usually in the format com.example.myfirstapp).
3. Select **Save Location** on your system.
4. Choose **Programming Language**: **Java** or **Kotlin**.
5. Select **Minimum SDK** (for example, API 21 for Android 5.0 Lollipop).
6. Click **Finish**.

Step 4: Understanding the Project Structure

- **MainActivity.java (or .kt)** → Main logic of the app.
- **activity_main.xml** → UI layout of the app.
- **AndroidManifest.xml** → Defines app permissions and activities.
- **Gradle Files** → Manage dependencies and app settings.

Step 5: Design the User Interface (UI)

1. Open **activity_main.xml** under res/layout.
2. Switch to **Design Mode** or **Code Mode**.
3. Drag and drop a **Button** and **TextView** from the Palette.
4. Set the Button's text to **"Click Me"**.
5. Set an **ID** for the Button (btnClick) and TextView (txtMessage).

Step 6: Add Functionality (Java/Kotlin Code)

1. Open **MainActivity.java** (app/java/com.example.myfirstapp).
2. Modify the code to handle button click (refer any sample code for the same).

Step 7: Run the App

1. Connect an **Android Device** (Enable **USB Debugging**) OR use the **Emulator**:
    - Go to **Tools > Device Manager**
    - Click **Create Virtual Device**, select a model, and install an Android System Image.
2. Click **Run** (▶ button) or press **Shift + F10**.

That's all!!

Program Number 1:

Creating "Hello world" Application

Java Code

```
package com.example.bca_program1;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        Button b;
        b = findViewById(R.id.b1);
        b.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(MainActivity.this, "Hey Created My first
Android Application", Toast.LENGTH_LONG).show();
            }
        });
    }

    }
```


Code explanation:

1. **Package Declaration:** Defines the package name for the app.
2. **Import Statements:** Import necessary classes for UI handling, button interaction, and displaying messages (Toast).
3. **Main Activity Class:** Defines MainActivity as the entry point of the application, extending AppCompatActivity for backward compatibility.
4. **onCreate Method:** Initializes the activity when it is launched.
5. **Enabling Edge-to-Edge Display (Optional):** Allows full-screen mode with edge-to-edge content display.
6. **Setting the Layout:** Connects the activity to the XML layout file (activity_main.xml).
7. **Button Initialization and Click Event Handling:**
    a. Finds the button from the XML layout.
    b. Sets an OnClickListener to detect button clicks.

c. Displays a **Toast message** when the button is clicked.

8. **Summary:** This app displays a button, and when clicked, it shows a toast message saying, **"Hey Created My first Android Application"**.

XML code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:layout_gravity="center_vertical">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/b1"
        android:text="Click Me"
        android:layout_marginTop="100dp"
        android:layout_marginLeft="100dp"
        />
</LinearLayout>
```


Code explanation:

1. **XML Declaration:** Specifies the version and encoding of the XML file.
2. **LinearLayout:**
   - The root layout container that arranges child views in a linear direction (vertical or horizontal).
   - Uses xmlns attributes to define Android, app, and tools namespaces.
   - android:id="@+id/main" assigns a unique ID to the layout.
   - android:layout_width="match_parent" and android:layout_height="match_parent" make it occupy the full screen.
   - tools:context=".MainActivity" links this layout to MainActivity for previewing in the design editor.
   - android:layout_gravity="center_vertical" aligns its content vertically in the center.
3. **Button:**
   - A clickable UI component inside LinearLayout.
   - android:id="@+id/b1" assigns a unique ID to the button.
   - android:text="Click Me" sets the button label.
   - android:layout_width="wrap_content" and android:layout_height="wrap_content" make the button size fit its content.
   - android:layout_marginTop="100dp" and android:layout_marginLeft="100dp" add spacing from the top and left.

Program Number 2:

Creating an application that displays message based on screen orientation.

Java Code

```
package com.example.bca_program2;

import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        Button l,p;
        l=findViewById(R.id.lan);
        p=findViewById(R.id.por);
        l.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
                Toast.makeText(MainActivity.this,"Hey!We are in
Landscape Mode",Toast.LENGTH_LONG).show();
            }
                });
        p.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
                Toast.makeText(MainActivity.this,"Hey!We are in
Portrait",Toast.LENGTH_LONG).show();
            }
        });
    }
}
```


Code explanation:

1. **Package Declaration:** Defines the package name for the app.

2. **Import Statements:** Includes required classes for UI handling, screen orientation, button interaction, and displaying messages.
3. **Main Activity Class:** MainActivity extends AppCompatActivity, making it the main entry point.
4. **onCreate Method:** Initializes the activity when launched.
5. **Enabling Edge-to-Edge Display:** Allows full-screen content display.
6. **Setting the Layout:** Connects MainActivity to activity_main.xml.
7. **Button Initialization:** Finds two buttons (lan for Landscape mode and por for Portrait mode) using findViewById().
8. **Click Event Handling:**
    a. Clicking **Landscape Button**: Changes screen orientation to **landscape** and displays a **Toast message**.
    b. Clicking **Portrait Button**: Changes screen orientation to **portrait** and displays a **Toast message**.

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:id="@+id/main"
    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/por"
        android:text="Portrait"
        android:layout_centerInParent="true"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/lan"
        android:text="landscape"
        android:layout_below="@id/por"
        android:layout_centerInParent="true"/>

</RelativeLayout>
```


Code explanation:

Like Lab program 1 – XML code. However, the changes could be in the ID assignment.

Program Number 3:

Create an application to develop Login windows using UI controls.

Java Code

```
package com.example.bca_program3;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {
EditText a,b;
Button c;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        a=findViewById(R.id.ed1);
        b= findViewById(R.id.ed2);
        c=findViewById(R.id.b1);
        c.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
              String username=a.getText().toString().trim();
              String password=b.getText().toString().trim();
              if(username.equals("admin")&&password.equals("pass")) {
                  Toast.makeText(MainActivity.this, "Login Successful",
Toast.LENGTH_SHORT).show();
              }else {
                  Toast.makeText(MainActivity.this,"Invalid UserName or
password",Toast.LENGTH_SHORT).show();
              }
            }
        });
    }
}
```

Code explanation:

This Android application defines a simple login interface using Java. The MainActivity class extends AppCompatActivity, making it the main screen of the app. The onCreate method initializes the UI components, including two EditText fields (a and b) for username and password input and a Button (c) to trigger login validation.

When the button is clicked, an OnClickListener captures the entered username and password, trims extra spaces, and checks if they match predefined values ("admin" for username and "pass" for password). If correct, a success message is displayed using Toast; otherwise, an error message appears. The EdgeToEdge.enable(this); ensures full-screen UI compatibility.

XML code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:id="@+id/main"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity"
  android:orientation="vertical">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="LogIn"
    android:gravity="center"
    android:id="@+id/t1"
    />
  <EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="text"
    android:layout_marginTop="100dp"
    android:hint="Name"
    android:id="@+id/ed1"/>
  <EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/ed2"
    android:inputType="text"
    android:layout_marginTop="160dp"
    android:hint="password"/>
  <Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/b1"
    android:text="Login"/>

</LinearLayout>
```


Code explanation:

1. **XML Declaration**: The file starts with an XML declaration specifying version 1.0 and UTF-8 encoding.

2. **Root Layout (LinearLayout)**: Defines a vertical **LinearLayout** as the main container, spanning the full width and height of the screen.
3. **Namespace Declarations**: Includes Android, app, and tools namespaces for attributes and preview support.
4. **TextView for Title**: A **TextView** displays "LogIn" at the top, centered within its space.
5. **EditText for Name**: A text input field is provided for the user's name, taking full width, with a hint "Name" and a top margin of 100dp.
6. **EditText for Password**: Another text input field for the password, styled similarly but with a top margin of 160dp and the hint "password".
7. **Button for Login**: A full-width **Button** labeled "Login" is placed at the bottom, intended to trigger authentication logic.

Program Number 4:

Create an application to implement new activities using explicit intent, implicit intent and content provider.

Java Code – MainActivity.java

```
package com.example.bca1_program4;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {
    Button btnexplicit;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        btnexplicit=findViewById(R.id.btnexplicitContent);
        btnexplicit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent=new Intent(MainActivity.this,
activity_new.class);
                startActivity(intent);
            }
        });
    }

}
```

Code explanation:

1. **Package Declaration** – The code belongs to the package com.example.bca1_program4.
2. **Imports Required Libraries** – Essential Android libraries for UI components, activity management, and edge-to-edge display adjustments are imported.
3. **Class Definition** – MainActivity extends AppCompatActivity, making it a primary activity with support for modern Android features.
4. **Variable Declaration** – A Button named btnexplicit is declared to handle user interactions.
5. **onCreate Method** – The method initializes the activity when it is launched.
6. **Edge-to-Edge Display** – The EdgeToEdge.enable(this); method ensures the UI extends to the edges of the screen.
7. **Set Layout** – The activity's UI is set using setContentView(R.layout.activity_main), linking it to activity_main.xml.

8. **Button Initialization** – btnexplicit is linked to a button in the XML layout with the ID btnexplicitContent.
9. **Click Listener** – A setOnClickListener is attached to the button to detect user clicks.
10. **Explicit Intent** – When the button is clicked, an Intent is created to navigate from MainActivity to activity_new.
11. **Start New Activity** – startActivity(intent); launches activity_new, transitioning the user to a new screen.

Java Code – activity_new.java

```
package com.example.bca1_program4;

import static android.content.Intent.ACTION_VIEW;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class activity_new extends AppCompatActivity {
    Button btnImplicit;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new);
        btnImplicit=findViewById(R.id.btnImplicitContent);
        btnImplicit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Uri webpage=Uri.parse("http://www.google.com");
                Intent intent=new Intent(ACTION_VIEW,webpage);
                startActivity(intent);
            }
        });
    }
}
```


Code explanation:

1. **Package Declaration:** The code is part of the com.example.bca1_program4 package.
2. **Imports Required Classes:** It imports necessary Android classes, including Intent, Uri, Button, and others for handling UI and implicit intents.
3. **Class Definition:** Defines activity_new, which extends AppCompatActivity, meaning it represents an Android activity with app compatibility features.
4. **Declares Button Variable:** A Button named btnImplicit is declared.

5. **onCreate() Method Execution:** When the activity starts, the onCreate() method is called to initialize the UI.
6. **Setting Content View:** The activity layout is set using setContentView(R.layout.activity_new), linking it to an XML layout file.
7. **Button Initialization:** The btnImplicit button is linked to the corresponding button in the XML layout using findViewById().
8. **Button Click Listener:** A click listener is assigned to btnImplicit.
9. **Implicit Intent Creation:** When clicked, a Uri object is created for "http://www.google.com".
10. **Launching Web Browser:** An Intent with ACTION_VIEW and the Uri is created, triggering the default web browser to open the specified URL.
11. **Executing Intent:** The startActivity(intent) method launches the intent, opening the browser.

XML code – activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btnexplicitContent"
        android:textSize="30sp"
        android:layout_marginBottom="100dp"
        android:text="Explicit Content"
        android:layout_marginTop="30dp"/>

</LinearLayout>
```

Code explanation:

1. **XML Declaration** – The file starts with an XML declaration specifying version 1.0 and UTF-8 encoding.
2. **Root Layout (LinearLayout)** – Defines a vertical layout that stacks elements from top to bottom.
3. **Namespace Declarations** – Includes standard Android, auto-generated, and tools namespaces for attributes.
4. **Layout Properties** – The LinearLayout takes up the full width and height of the parent container.
5. **Context Reference** – The tools:context attribute links this layout to MainActivity for preview purposes in the design editor.
6. **Button Element** – A button is placed inside the LinearLayout.

7. **Button Properties** –

   - Occupies full width while wrapping its height.
   - Has a unique ID for reference in Java/Kotlin code.
   - Displays text "Explicit Content" with a font size of 30sp.
   - Includes vertical margins: 30dp at the top and 100dp at the bottom.

XML code – activity_new.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activity_new">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btnImplicitContent"
        android:text="ImplicitContent"
        android:textSize="30sp"
        android:layout_marginTop="30dp"/>
</LinearLayout>
```

Code explanation:

1. **XML Declaration** – The file starts with an XML declaration specifying version 1.0 and UTF-8 encoding.
2. **Root Layout** – A LinearLayout is used as the root container, which arranges child views in a linear fashion (either vertically or horizontally).
3. **Namespace Declarations** – It includes standard Android, auto, and tools namespaces to define attributes and provide development-time hints.
4. **Layout Properties** – The LinearLayout takes up the entire screen width and height (match_parent).
5. **Context Reference** – The tools:context attribute links the layout to an activity for design-time preview.
6. **Button Definition** – A Button is placed inside the layout with a unique ID for reference in code.
7. **Button Appearance** – The button's text is set to "ImplicitContent" with a text size of 30sp.
8. **Button Layout** – The button spans the full width of the screen (match_parent) and adjusts its height to fit its content (wrap_content).
9. **Margin Addition** – A top margin of 30dp is applied for spacing from the top of the layout.

Program Number 5:

Create an application that displays custom designed Opening Screen.

Java Code

```
package com.example.bca_program5;

import android.os.Bundle;
import android.widget.RelativeLayout;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.content.res.ResourcesCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {
RelativeLayout rl;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        rl=findViewById(R.id.main);
        rl.setBackground(ResourcesCompat.getDrawable(getResources(),
R.drawable.back_drawable, null));
    }
}
```

Code explanation:

1. **Package Declaration**: The code belongs to the package com.example.bca_program5, indicating an Android application module.
2. **Imports**: Necessary Android and Jetpack libraries are imported, including RelativeLayout, EdgeToEdge, and various resource utilities for UI customization.
3. **Class Definition**: MainActivity extends AppCompatActivity, making it the main entry point for the app's user interface.
4. **Variable Declaration**: A RelativeLayout named rl is declared to reference the layout component.
5. **onCreate() Method**: The lifecycle method onCreate() initializes the activity when it is launched.
6. **Edge-to-Edge Display**: The EdgeToEdge.enable(this); method ensures the app uses the full-screen layout without default padding or margins.
7. **Layout Inflation**: The XML layout file activity_main.xml is set as the content view.
8. **Finding View by ID**: The RelativeLayout is linked to the UI component with the ID main from the layout file.
9. **Background Customization**: The background of the RelativeLayout is set using a drawable resource (back_drawable) retrieved using ResourcesCompat.getDrawable().

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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/TVHead"
        android:layout_centerInParent="true"
        android:layout_margin="20dp"
        android:gravity="center"
        android:padding="10dp"
        android:text="Background drawable in Android"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:textStyle="bold"/>
        />

</RelativeLayout>
```

Create back_drawable.xml :

1. **Right Click drawable folder,new →Drawable Resource File→New→back_drawable.**

2.Under Drawable a new file back_drawable is created. Type the following xml code in it.

XML code

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
<gradient
    android:angle="270"
    android:endColor="@color/white"
    android:startColor="#2C3A87"/>
</shape>
</RelativeLayout>
```

Code explanation:

1. **XML Declaration**: Specifies the XML version and character encoding used in the file.
2. **Root Layout**: A RelativeLayout is defined as the parent container, allowing child views to be positioned relative to each other.
3. **Namespace Declarations**: Three XML namespaces are included:
   - android for standard Android attributes.
   - app for custom attributes from libraries.
   - tools for design-time attributes in Android Studio.

4. **Layout Properties**: The RelativeLayout takes up the full width and height of the parent (match_parent).
5. **TextView Definition**: A TextView is created inside the RelativeLayout.
6. **TextView Dimensions**: The width is set to match the parent, while the height wraps around the content.
7. **TextView Positioning**: The layout_centerInParent attribute centers the TextView within the RelativeLayout.
8. **Margins and Padding**: A margin of 20dp and padding of 10dp are added for spacing.
9. **Text Appearance**:
   - Displays "Background drawable in Android".
   - Aligns text at the center.
   - Sets text color to black.
   - Uses a font size of 20sp.
   - Applies bold styling.
10. **Syntax Error**: An extra closing tag (/>) after TextView causes an error. Removing it fixes the issue.

Program Number 6:

Create an UI with all views.

Java Code

```
package com.example.bca_program6;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    EditText ed1;
    DatePicker picker;
    RadioButton male,female;
    TextView txt2,txt3;
    Button btn1;
    RadioGroup rg1;
    CheckBox chk1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ed1 = findViewById(R.id.edittext1);
        male=findViewById(R.id.rbmale);
        female=findViewById(R.id.rbfemale);
```

```
        txt2=findViewById(R.id.textview2);
        txt3=findViewById(R.id.textview3);

        btn1 = findViewById(R.id.button1);
        chk1=findViewById(R.id.check1);
        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                txt2.setText( "Name:"+ed1.getText());
                String result="";

result+=(male.isChecked())?"male":(female.isChecked())?"female":"";
                Toast.makeText(getApplicationContext(),"Successfully
submitted", Toast.LENGTH_SHORT).show();
                String str=result;
                txt3.setText("Gender:"+result);

            }


        });


    }
}
```

Code explanation:

1. **Package Declaration:** The code belongs to the package com.example.bca_program6.
2. **Imports:** It imports necessary Android classes for UI components, activity management, and edge-to-edge display handling.
3. **Class Definition:** The MainActivity class extends AppCompatActivity, making it the main entry point for the app.
4. **UI Components:** A TextView and a Button are declared as instance variables.
5. **onCreate Method:**
   - The method is called when the activity is created.
   - The layout activity_main.xml is set as the UI.
   - The TextView and Button are linked to their respective UI elements using findViewById().
6. **Button Click Listener:** When the button is clicked, the TextView displays the message **"All the Best"**.
7. **Can add RadioButtons,Checkbox and EditText if required.**

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"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="17dp"
        android:text="UI WITH ALL VIEWS"
        android:textSize="30dp" />

    <TextView
        android:id="@+id/text1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="35dp"
        android:layout_below="@+id/textView1"
        android:text="Name"
        android:textSize="30dp" />

    <EditText
        android:id="@+id/edittext1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Entyer your number"
        android:layout_alignParentLeft="true"
        android:inputType="text"
        android:textSize="20dp"
        android:layout_marginTop="68dp"
        android:layout_marginLeft="88dp"/>

     <TextView
        android:id="@+id/txt1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Gender"
         android:layout_marginTop="65dp"
        android:layout_below="@+id/edittext1"
        android:textSize="30dp" />

    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="230dp"
        android:gravity="center"
        android:orientation="horizontal"
        >
    <RadioButton
        android:id="@+id/rbmale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="30dp"
```

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

    <RadioButton
        android:id="@+id/rbfemale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="30dp"
        android:text="Female" />
    </RadioGroup>
    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/check1"
        android:text="I agree to the terms and conditions"
        android:layout_marginTop="350dp"/>
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/radioGroup"
        android:layout_marginTop="108dp"
        android:text="Update" />
    <TextView
        android:id="@+id/textview2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
       android:layout_below="@+id/button1"
        android:layout_marginTop="110dp"
        android:textSize="30dp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textview3"
        android:layout_below="@+id/textview2"
        android:layout_marginTop="30dp"
        android:textSize="30dp" />

    />


</RelativeLayout>
```


Code explanation:

1. **XML Declaration**: Defines the XML version and character encoding used in the file.
2. **Root Layout**: A LinearLayout is defined as the root container, setting its width and height to fill the screen (match_parent).
3. **Namespace Declarations**: Includes standard Android XML namespaces for Android attributes, custom attributes, and design-time tools.
4. **Layout Orientation**: Specifies a vertical orientation, meaning child elements are stacked one below the other.
5. **TextView Component**:

- Displays the text "Surana College" in bold.
- Has a width that fills the parent and height that adjusts to content.
- Positioned centrally using layout_gravity.
- Assigned a unique ID (tv).

6. **Button Component**:
   - A clickable button with the label "Button".
   - Matches the parent width and wraps its height to fit the content.
   - Given a unique ID (b1).
7. **Usage**: This layout defines a simple UI with a title text and a button, commonly used in Android applications.

Program Number 7:

Create menu in Application.

Java Code

```
package com.example.bca_program7;

import android.os.Bundle;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

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

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v,
insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
            return insets;
        });
    }
}
```


Code explanation:

1. **Package Declaration**: Defines the package com.example.bca_program7, organizing the code within a specific namespace.
2. **Imports**: Brings in necessary Android and Jetpack libraries for UI handling, including EdgeToEdge, AppCompatActivity, and WindowInsetsCompat.
3. **Class Declaration**: MainActivity extends AppCompatActivity, making it the main entry point of the Android app.

4. **onCreate() Method**: Executes when the activity is created, initializing the UI and setting up insets for full-screen handling.
5. **Enable Edge-to-Edge Mode**: EdgeToEdge.enable(this) ensures the activity uses the full screen, including behind system bars.
6. **Set Layout**: setContentView(R.layout.activity_main) inflates the activity_main layout, displaying the UI.
7. **Handle Window Insets**:
   - Retrieves the root view (findViewById(R.id.main)).
   - Listens for system insets (status bar, navigation bar).
   - Adjusts padding dynamically to prevent UI overlap with system bars.

XML code : 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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

2. Right click res→new→Android Resource Directory→directory name→menu
3. Right click newly created menu directory→new→menu resource file→my_menu

XML code :my_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:title=""
        app:showAsAction="always"
        android:icon="@drawable/baseline_menu_24">
        <menu>
    <item android:id="@+id/new_game"
        android:icon="@drawable/baseline"
        app:showAsAction="never"
        android:title="new_game"/>
    <item android:id="@+id/help"
        android:icon="@drawable/outline_help_outline_24"
        app:showAsAction="never"
        android:title="help" />
```

```
    </menu>
    </item>
</menu>
```

- **Ensure in themes.xml , the theme is changed as follows :**

```
<resources xmlns:tools="http://schemas.android.com/tools">
  <!-- Base application theme. -->
  <stylename="Base.Theme.BCA_Program8"
parent="Theme.AppCompat.DayNight.DarkActionBar">
    <!-- Customize your light theme here. -->
    <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
  </style>

  <style name="Theme.BCA_Program8" parent="Base.Theme.BCA_Program8" />
</resources>
```

Code explanation:

1. **XML Declaration** – Specifies that the document follows XML format with UTF-8 encoding.
2. **Root Layout (ConstraintLayout)** – The entire UI is wrapped in a ConstraintLayout, which allows flexible positioning of child elements.
3. **Namespace Declarations** – Defines standard Android, auto (app), and tools namespaces for attributes.
4. **Layout Properties** – The ConstraintLayout fills the parent container (match_parent for both width and height).
5. **TextView Definition** – A TextView is added inside the ConstraintLayout with the text "Hello World!".
6. **Size Attributes** – The TextView uses wrap_content for both width and height, making it size dynamically based on content.
7. **Constraints** – The TextView is centered by constraining all four sides (Top, Bottom, Start, End) to the parent layout.
8. **Tools Context** – Defines .MainActivity as the context for preview purposes in the Android Studio design editor.

Program Number 8:

Read / Write data the Local Data

Java Code

```
package com.example.f1;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.InputStreamReader;

public class MainActivity extends AppCompatActivity {
EditText fn,data;
Button sb,rb;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        fn = findViewById(R.id.ed1);
        data = findViewById(R.id.ed2);
        sb = findViewById(R.id.button1);
        rb = findViewById(R.id.button2);
        sb.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String filename = fn.getText().toString();
                String d1 = data.getText().toString();
                FileOutputStream fos;
                try {
                    fos = openFileOutput(filename,
Context.MODE_PRIVATE);
                    fos.write(d1.getBytes());
                    fos.close();
                    Toast.makeText(getApplicationContext(), filename +
"saved", Toast.LENGTH_LONG).show();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
        rb.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View view) {
                String filename = fn.getText().toString();
                String name =
String.valueOf(getFileStreamPath(filename));
                StringBuffer stringB = new StringBuffer();
                try {
                    BufferedReader inputReader
                            = new BufferedReader(new
InputStreamReader(openFileInput(filename)));
                    String inputString;
                    while ((inputString = inputReader.readLine()) !=
null) {
                        stringB.append(inputString + "\n");
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
                Toast.makeText(getApplicationContext(),
stringB.toString(),
                        Toast.LENGTH_LONG).show();
            }
        });
    }
        }
```

Code explanation:

1. **Package and Imports**: The necessary Android classes are imported, including file handling, UI components, and context-related functionalities.
2. **Class Declaration**: MainActivity extends AppCompatActivity, enabling the use of modern Android UI features.
3. **UI Component Initialization**:
   - Two EditText fields (fn for filename, data for content).
   - Two Button elements (sb for saving, rb for reading).
4. **onCreate() Method Execution**:
   - The UI is set using setContentView().
   - The EditText and Button elements are linked to their respective XML components using findViewById().
5. **Save Button Click Listener** (sb):
   - Retrieves text from EditText fields (filename and content).
   - Writes the content to a file in internal storage using FileOutputStream.
   - Displays a toast message indicating successful file saving.
6. **Read Button Click Listener** (rb):
   - Retrieves the filename from EditText.
   - Opens and reads the file content using BufferedReader.
   - Appends file content to a StringBuffer.
   - Displays the content in a toast message.
7. **Exception Handling**: Both file operations handle exceptions, ensuring the app does not crash due to file access issues.

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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="File Name:"
        android:textSize="20dp"
        android:id="@+id/TextView1"
        android:layout_alignBaseline="@+id/ed1"
        android:layout_alignBottom="@+id/ed1"
        android:layout_alignParentLeft="true"/>
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/ed1"
        android:layout_marginRight="20dp"
        android:layout_marginTop="25dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Data:"
        android:textSize="20dp"
        android:id="@+id/TV2"
        android:layout_below="@+id/TextView1"
        android:layout_alignParentLeft="true"/>
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/ed2"
        android:layout_marginRight="20dp"
        android:layout_marginTop="75dp"
        android:layout_below="@+id/ed1"
        android:hint="Enter data here"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button1"
        android:text="SAVE"
        android:layout_marginLeft="50dp"
        android:layout_marginTop="100dp"
```

```
        android:layout_below="@id/TV2"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button2"
        android:text="READ"
        android:layout_marginLeft="200dp"
        android:layout_marginTop="100dp"
        android:layout_below="@id/TV2"
        />
</RelativeLayout>
```

Code explanation:

1. **XML Declaration**: The file starts with an XML declaration specifying version 1.0 and UTF-8 encoding.
2. **Root Layout**: A RelativeLayout is used as the parent layout, allowing child elements to be positioned relative to each other.
3. **Text Label for File Name**: A TextView displays "File Name:", positioned to align with an EditText for file input.
4. **File Name Input**: An EditText is placed at the top right, allowing users to enter a file name. It has margins for spacing.
5. **Text Label for Data Input**: Another TextView displays "Data:", positioned below the first label.
6. **Data Input Field**: An EditText appears below the file name input, allowing users to enter data. It includes a hint to guide input.
7. **Save Button**: A Button labeled "SAVE" is placed below the "Data" label, with margins for spacing.
8. **Read Button**: Another Button labeled "READ" is positioned next to the "SAVE" button with adjusted margins for alignment.
9. **Overall Layout Structure**: Elements are arranged logically with proper alignments to create a simple user interface for file handling.

Program Number 9:

Create / Read / Write data with database (SQLite).

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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Student Details"
        android:textSize="25dp"
        android:layout_centerHorizontal="true"/>

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Rollno"
        android:id="@+id/et1"
        android:layout_marginTop="50dp"/>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Name"
        android:id="@+id/et2"
        android:layout_marginTop="35dp"
        android:layout_below="@id/et1"
        android:layout_centerHorizontal="true"/>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Department"
        android:id="@+id/et3"
        android:layout_marginTop="35dp"
        android:layout_below="@id/et2"
        android:layout_centerHorizontal="true"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/b1"
        android:layout_below="@id/et3"
        android:text="Update"
        android:onClick="onUpdate"
        android:textSize="20dp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"/>
    <Button
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/b2"
        android:layout_below="@id/b1"
        android:text="Read"
        android:onClick="onRead"
        android:textSize="20dp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/b3"
        android:layout_below="@id/b2"
        android:text="Insert"
        android:onClick="onInsert"
        android:textSize="20dp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/b4"
        android:layout_below="@id/b3"
        android:text="Delete"
        android:onClick="onDelete"
        android:textSize="20dp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"/>

</RelativeLayout>
```

Java Code(MainActivity.java)

```
package com.example.sql1;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {
SQLiteDatabase db;
```

```
EditText et1,et2,et3;
Button b1,b2,b3,b4;
String rno;
String name;
String dept;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        et1=findViewById(R.id.et1);
        et2=findViewById(R.id.et2);
        et3=findViewById(R.id.et3);
        b1=findViewById(R.id.b1);
        b2=findViewById(R.id.b2);
        b3=findViewById(R.id.b3);
        b4=findViewById(R.id.b4);
        DBhelper dBhelper=new DBhelper(this);
        db=dBhelper.getReadableDatabase();
        db=dBhelper.getWritableDatabase();

        }
    public void onUpdate(View view) {
        rno=et1.getText().toString();
        name=et2.getText().toString();
        dept=et3.getText().toString();
        if(rno.equals("")|| name.equals("")||dept.equals(""))
        {
            Toast.makeText(this, "Please enter values",
Toast.LENGTH_LONG).show();
            return;
        }else {
            ContentValues values = new ContentValues();
            values.put("rollno", rno);
            values.put("name", name);
            values.put("dept", dept);
            db.update("student", values,
                    "rollno="+rno,null);
            Toast.makeText(this, "Values Inserted" +
                    "Successfully", Toast.LENGTH_LONG).show();

        } }

    public void onRead(View view) {
        StringBuffer buffer=new StringBuffer();
        Cursor c=db.rawQuery("select * from student",
                null);
        while(c.moveToNext()) {
            buffer.append("\n" + c.getString(0));
            buffer.append("\n" + c.getString(1));
            buffer.append("\n" + c.getString(2));
        }
        Toast.makeText(this,buffer.toString(),
                Toast.LENGTH_LONG).show();
```

```
    }

    public void onInsert(View view) {
        rno=et1.getText().toString();
        name=et2.getText().toString();
        dept=et3.getText().toString();
        if(rno.equals("")|| name.equals("")||dept.equals(""))
        {
            Toast.makeText(this, "Please enter values",
Toast.LENGTH_LONG).show();
            return;
        }else {
            ContentValues values = new ContentValues();
            values.put("rollno", rno);
            values.put("name", name);
            values.put("dept", dept);
            db.insert("student", null, values);
            Toast.makeText(this, "Values Inserted" +
                    "Successfully", Toast.LENGTH_LONG).show();

        }
    }

    public void onDelete(View view) {
        rno=et1.getText().toString();
        name=et2.getText().toString();
        dept=et3.getText().toString();
        if(rno.equals(""))
        {
            Toast.makeText(this,"Please enter roll no",
                    Toast.LENGTH_LONG).show();
            return;} else{
            db.delete("student","rollno="+rno,
                    null);
            Toast.makeText(this,"Values Deleted Successfully",
                    Toast.LENGTH_LONG).show();

        }
    }
}
```

1.Go to main→java→com.example.program10,Right click →new→Java class→DBhelper, select class.

Java Code(DBhelper.java)

```
package com.example.sql1;

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

```
import androidx.annotation.Nullable;

public class DBhelper extends SQLiteOpenHelper {

   public DBhelper(@Nullable Context context) {
      super(context, "student", null, 1);
   }

   @Override
   public void onCreate(SQLiteDatabase sqLiteDatabase) {
      sqLiteDatabase.execSQL("create table " +
            "student(rollno int,name varchar(20)," +
            "dept varchar(5))");
   }

   @Override
   public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
sqLiteDatabase.execSQL("drop table if exists student");
onCreate(sqLiteDatabase);
   }
}
```

Code explanation:

1. Activity_main.xml gives the UI design .
2. Main_activity.java gives the code for creating an instance
3. DBhelper.java is required to connect to sqlite.

Program Number 10:

Create an application to Send and Receive SMS

XML code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Enter Phone Number"
        android:textSize="30dp"
        android:gravity="center"
        android:layout_marginTop="30dp"/>
    <EditText
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/ed1"
        android:phoneNumber="true"/>
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Enter Message"
        android:gravity="center"
        android:textSize="30dp"
        android:layout_marginTop="50dp"/>
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/ed2"
        android:inputType="textMultiLine"
        android:lines="5"
        android:gravity="center"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/buttonsend"
        android:gravity="center"
        android:textSize="20dp"
        android:text="SEND"/>

</LinearLayout>
```

4. Android Manifest File needs permissions to be set

AndroidManifest.XML code

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

    <uses-feature
        android:name="android.hardware.telephony"
        android:required="false" />
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission
android:name="android.permission.READ_PHONE_STATE"/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Sms_final"
        tools:targetApi="31">
```

```
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver
            android:name=".smsReceiver"
            android:exported="true"
            android:permission="BROADCAST_SMS">
            <intent-filter>
                <action
android:name="android.provider.Telephony.SMS_RECEIVED"/>
            </intent-filter>
        </receiver>
    </application>

</manifest>
```

5. Main_activity.java contains the code to send SMS

Activity_main.java

```
package com.example.sms_final;

import android.app.PendingIntent;
import android.content.Intent;
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.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        EditText phone = findViewById(R.id.ed1);
        EditText msg = findViewById(R.id.ed2);
```

```
        Button send = findViewById(R.id.buttonsend);
        send.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String phoneno = phone.getText().toString();
                String message = msg.getText().toString();
                SmsManager smsmanager = SmsManager.getDefault();
                Intent intent = new Intent(getApplicationContext(),
MainActivity.class);
                PendingIntent pi =
PendingIntent.getActivity(getApplicationContext(), 0, intent,
PendingIntent.FLAG_IMMUTABLE);
                smsmanager.sendTextMessage(phoneno, null, message, pi,
null);
                Toast.makeText(getApplicationContext(),
                        "Message sent successfully",
                        Toast.LENGTH_LONG).show();
            }
        });
    }
    }
```

6. Go to Main→java→com.example.Program11. Right click→New→Java Class and create new java file **smsReceiver**.

smsReceiver.java

```
package com.example.sms_final;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;

public class smsReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle bundle=intent.getExtras();
        SmsMessage[] msg=null;
        String str="SMS From";
        if(bundle!=null)
        {
            Object[] recv=(Object[])bundle.get("pdus");
            msg=new SmsMessage[recv.length];
            for(int i=0;i<msg.length;i++) {
                msg[i] = SmsMessage.createFromPdu((byte[]) recv[i]);
                if (i == 0) {
                    str += msg[i].getOriginatingAddress();
                    str += ":";
                }
```

```
                str += msg[i].getMessageBody().toString();
            }
            Toast.makeText(context,str,Toast.LENGTH_LONG).show();
                }

            }
        }
```

5.Ensure while executing ,SMS feature is allowed for the app.

Program Number 11:

Create an application to Send an email

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:id="@+id/main"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">
<EditText
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:hint="Enter Subject"
  android:id="@+id/sub"
  android:layout_marginTop="50dp"/>
  <EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter Content"
    android:id="@+id/con"
    android:layout_below="@+id/sub"
    android:layout_marginTop="50dp"/>
  <EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter MailAddress"
    android:id="@+id/mail"
    android:layout_below="@+id/con"
    android:layout_marginTop="50dp"/>
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/send"
    android:text="Send Email!"
```

```
    android:layout_below="@id/mail"
    android:layout_centerHorizontal="true"/>

</RelativeLayout>
```

Main_activity.java

```
   package com.example.bca_program11;

import static android.content.Intent.createChooser;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {
     Button b1;
     EditText ed1,ed2,ed3;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ed1=findViewById(R.id.sub);
        ed2=findViewById(R.id.con);
        ed3=findViewById(R.id.mail);
        b1=findViewById(R.id.send);
        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String subject,content,to_email;
                subject=ed1.getText().toString();
                content=ed2.getText().toString();
                to_email=ed3.getText().toString();
                if(subject.equals("") && content.equals("") &&
to_email.equals(""))
                {
                    Toast.makeText(MainActivity.this,"All Fields are
required",Toast.LENGTH_LONG).show();
                }
                else{
                    sendmail(subject,content,to_email);
                }
            }
```

```
        });
    }
        public void sendmail(String subject,String content,String
to_email){

        Intent intent=new Intent(Intent.ACTION_SEND);
        intent.putExtra(Intent.EXTRA_EMAIL,new String[]{to_email} );
        intent.putExtra(Intent.EXTRA_SUBJECT,subject);
        intent.putExtra(Intent.EXTRA_TEXT,content);
        intent.setType("message/rfc822");
        startActivity(Intent.createChooser(intent,"Choose email
content"));


        }

}
```

Program Number 12:

Display Map based on the Current/given location.

XML code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
  android:orientation="vertical"
  android:padding="16dp"
  tools:context=".MainActivity">
  <TextView
    android:id="@+id/locationTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="18sp" />
</LinearLayout>
```

Java Code:Main_activity.java

```
package com.example.gps;

import android.content.pm.PackageManager;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.location.Location;
import android.widget.TextView;
import android.Manifest;
import androidx.activity.EdgeToEdge;
```

```
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity implements LocationListener {
  private LocationManager locationManager;
  private TextView locationTextView;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    locationTextView = findViewById(R.id.locationTextView);
    if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
      ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);
    }
    // Get a reference to the location manager
    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
  }

  protected void onResume() {
    super.onResume();
// Request location updates from the location manager
    if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION) ==
        PackageManager.PERMISSION_GRANTED) {
      locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
this);
    }
  }
//    protected void onPause() {
//      super.onPause();
//// Sto                    p receiving location updates when the activity is paused
//      locationManager.removeUpdates(this);
//    }
  @Override
  public void onLocationChanged(@NonNull Location location) {
    locationTextView.setText("Latitude: " + location.getLatitude() + ", Longitude: " +
        location.getLongitude());
  }
  @Override
  public void onStatusChanged(String provider, int status, Bundle extras) {
    LocationListener.super.onStatusChanged(provider, status, extras);
  }
  @Override
```

```
    public void onProviderEnabled(@NonNull String provider) {
        LocationListener.super.onProviderEnabled(provider);
    }
    @Override
    public void onProviderDisabled(@NonNull String provider) {
        LocationListener.super.onProviderDisabled(provider);
    }
}
```

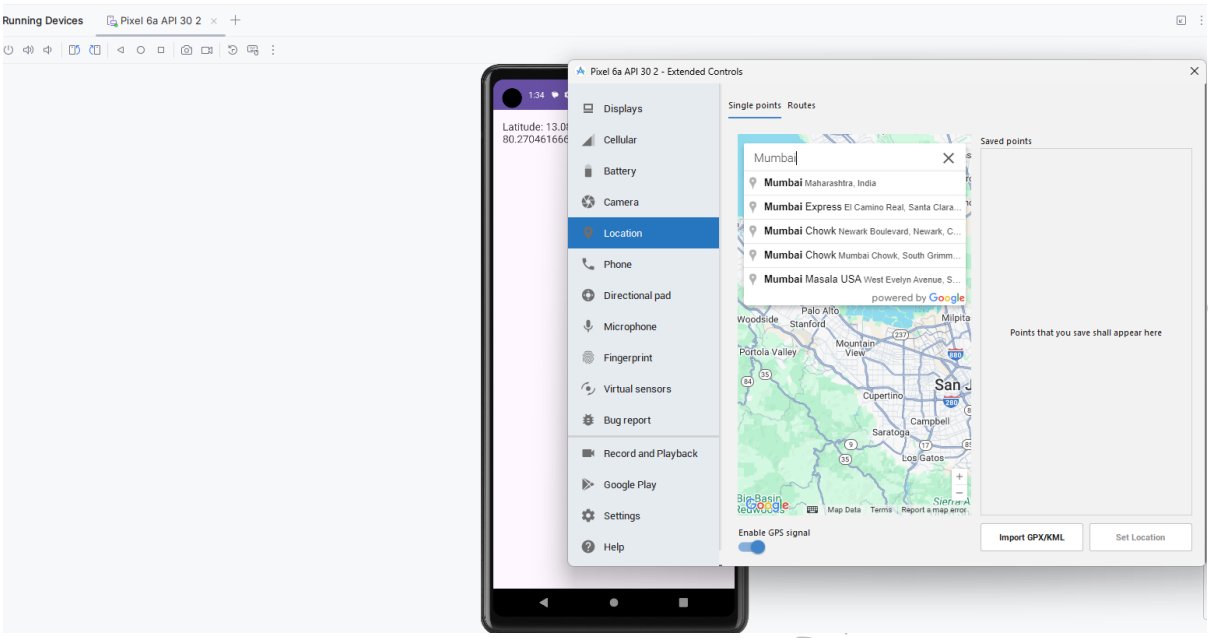
Android Manifest.xml code highlighted code to be added

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.GPS"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```


While executing change the location to change the latitude and longtitude. And click on set location and execute again.



Program 14:

Learn to deploy Android applications:

Steps to Deploy an Android Application

1. Prepare App (use Program 1 Hello world for this program) Optimize performance and test thoroughly.Ensure compatibility with various devices.

activity_main.xml Code:

```
<?xml version="1.0" encoding="utf8"?>
<androidx.constraintlayout.widget.ConstraintLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/resauto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".MainActivity">

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Hello World!"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"
 android:textSize="30sp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java

```
package com.example.helloworld;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity
{
 protected void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }
}
```

2. Generate Signed APK (Android Package Kit):
   In Android Studio, navigate to Build > Generate Signed Bundle/APK.
   Follow the prompts to create a new keystore or use an existing one. A keystore is a binary file that contains a set of private keys.
   Configure the build type (release) and signing configuration.
   Generate the signed APK file.
3. Test Your Signed APK:
 Before distributing your app, test the signed APK to ensure that the signing process didn't introduce any issues.
   Install the APK on various devices and perform thorough testing.
   Release on Google Play Console:
   Sign in to the Google Play Console (https://play.google.com/apps/publish).
   Create a new app entry if this is your first release or select an existing app.
   Complete all the required information for the app listing, including the title, description, screenshots, and categorization.
   Upload your signed APK file.
   Set pricing and distribution options.
   Optimize your store listing for search and conversion.
   Once everything is set, click the "Publish" button to release your app to the Google Play Store.
5. Other Distribution Channels (Optional):

   Besides Google Play, you can distribute your app through other channels such as Amazon Appstore, Samsung Galaxy Store, or third party app marketplaces.
   Each distribution channel may have its own requirements and submission process, so be sure to follow their guidelines.
6. Monitor and Update:
   Keep an eye on user feedback and app performance metrics through the Google Play Console.
   Regularly update your app to fix bugs, add new features, and improve user experience based on feedback