Q1. Define services in Android operating system.

A service is an application component that performs long-running operations, usually in the background. A service doesn't provide a User Interface (UI). A service runs in the main thread of its hosting process: the service doesn't create its own thread and doesn't run in a separate process unless you specify that it should.

1. A started service is a service that an application component starts by calling startService()
2. A bound service is a service that an application component binds to itself by calling bindService()

This service used for another app component interacts with it to perform Inter-Process Communication (IPC).

Q2. Discuss the need of permissions in Android. Describe the permissions to set system functionalities like Bluetooth and Camera.

The purpose of a permission is to protect the privacy of an Android user. Android apps must request permission to access sensitive user data (such as contacts and SMS), as well as certain system features (such as camera and internet). Depending on the feature, the system might grant the permission automatically or might prompt the user to approve the request.

1. You should declare the Camera requirement in your manifest file. Appropriate declaration will allow you to use Camera feature in your application.
   <uses-permission android:name = "android.permission.CAMERA"/>

2. To use Bluetooth feature in android applications, we need to add multiple permission in manifest file.
   <uses-permission android:name = "android.permission.BLUETOOTH"/>
   <uses-permission android:name = android.permission.BLUETOOTH_ADMIN"/>

Q3. Describe the significance of SQLite database in Android.

SQLite is a software library that implements SQL database engine that is:

1. Self-contained (requires no other components)
2. Server-less (requires no server backend)
3. Zero-configuration (Doesn't need to be configured for your application)
4. Transactional (changes within a single transaction in SQLite either occur completely or not at all)
5. SQLite is the most widely deployed database engine in the world.

Q4. Explain zoom control (IN / OUT) with the help of an example.

In Android, Zoom Controls class is used for zooming and provides callback to register for events. Zoom controls has two buttons ZoomIn and ZoomOut which are used to control the zooming functionality.

Example:

```
<LinearLayout
<ZoomControls
Android:id = "@+id/simpleZoomControl"
Android:layout_width = "wrap_content"
Android:layout_height = "wrap_content"
/> />
```

Q5. List different types of views? Explain Scroll view or Grid View or List View.

Types of views are:

1. ListView is a view, which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database.
2. GridView arranges items in two-dimensional scrolling grid and the grid items are not necessarily predetermined but they are automatically inserted to the layout using a ListAdapter.
3. ImageView class is used to display an image file in the application.
4. ScrollView can hold only one direct child. This means that, if you have complex layout with more views then you must enclose them inside another standard layout. After enclosing them in a standard layout, enclose the whole layout in ScrollView to make all the element or views scrollable.

Q6. State difference between Bound Service and UnBound Service.

Bound Service	UnBound Service
Basically used for long repetitive task but bound with the component.	Basically used for long repetitive task.
starts by bindService().	startService() is the method use to start unbound service.
unbindService() is the method use to stop explicitly.	stopService() is the method use to stop explicitly.
It is dependent of the component from which it starts	It is independent of the component from which it starts

Q7. Program to describe implicit intent or Explicit Intent.

Implicit Intent:

```
package example.javatpoint.com.implicitintent;
import android.content.Intent;
import android.net.Uri;
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 button;
   EditText editText;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      button = findViewById(R.id.button);
      editText =  findViewById(R.id.editText);
      button.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            String url=editText.getText().toString();
            Intent intent=new Intent(Intent.ACTION_VIEW, Uri.parse(url));
            startActivity(intent);
         }
      });
   }
}
```

Explicit Intent:

MainActivityOne.java

```
package example.javatpoint.com.explicitintent;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
public class FirstActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_first);
   }
   public void callSecondActivity(View view){
```

```
        Intent i = new Intent(getApplicationContext(), SecondActivity.class);
        i.putExtra("Value1", "Android By Javatpoint");
        i.putExtra("Value2", "Simple Tutorial");
        startActivity(i);
    }
}
```

MainActivityTwo.java

```
package example.javatpoint.com.explicitintent;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class SecondActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        Bundle extras = getIntent().getExtras();
        String value1 = extras.getString("Value1");
        String value2 = extras.getString("Value2");
        Toast.makeText(getApplicationContext(),"Values are:\n First value: "+value1+
                "\n Second Value: "+value2, Toast.LENGTH_LONG).show();
    }
    public void callFirstActivity(View view){
        Intent i = new Intent(getApplicationContext(), FirstActivity.class);
        startActivity(i);
    }
}
```

Q8. Difference between Simple Toast and Custom Toast.

Toast	Custom Toast
Toast is used to display information for a period of time.	Custom toast is used to display information which include images.
Doesn’t uses LayoutInflater	Uses LayoutInflater
We can create simple toast by using makeText() method.	We can create custom toast by using custom layout (xml file).
Use to show alert message to user.	Use to customize toast

Q9. Describe different types of Fragment.

Fragments are divided into three stages:

1. Single frame fragment is used in hand hold devices like mobiles above android 3.0 version. Here, we can only show one fragment as a view.
2. List fragment displays a list of items by binding to a data source such as an array or Cursor, and exposes event handlers when the user selects an item.
3. Fragments Transaction is used to move one fragment to another fragment.

Q10. Write the steps to get location in Android.

To get the current location,

1. Create a location client which is LocationClient object.
2. Connect it to Location Services using connect() method.
3. Then, call its getLastLocation() method.

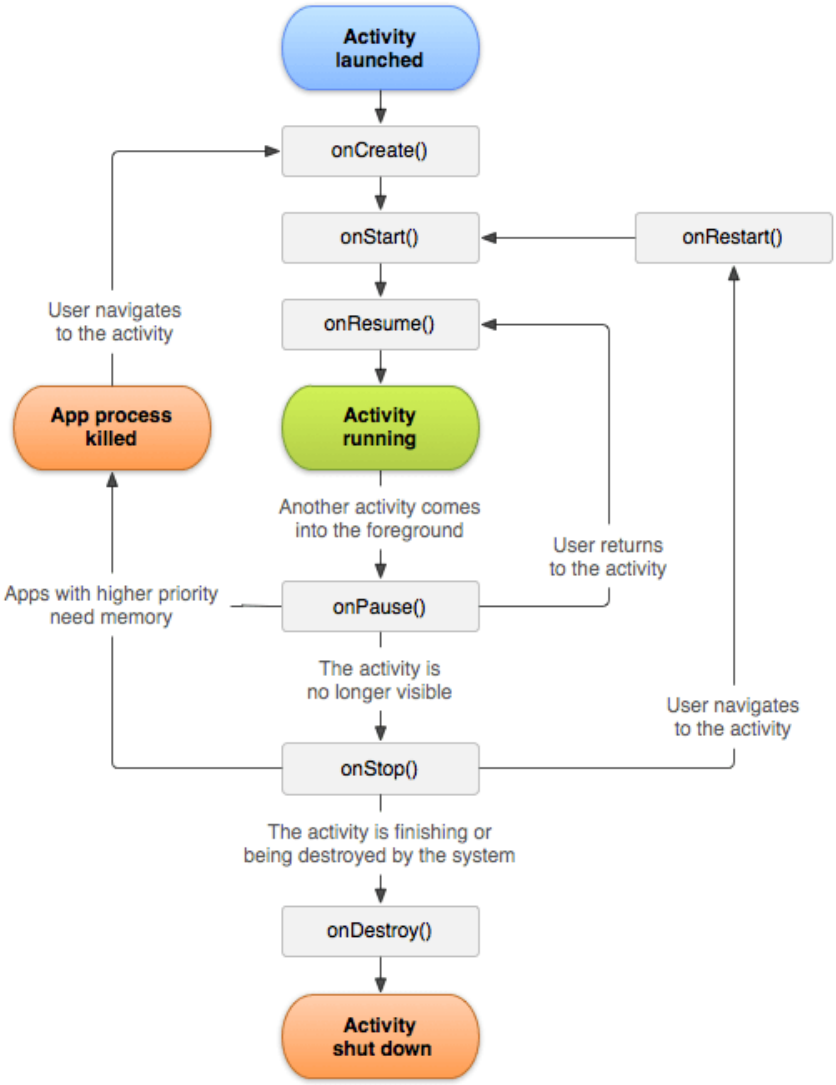
OR

1. Provide permission in the manifest file to get the location.
2. Create an instance of Location Manager as a reference of a Location Service.
3. Request location from the Location Manager.
4. Receive location update from the location Listener on the change of the location.

4-Mark Questions:

Q1. Explain the activity life cycle.

The Activity is in a set of states during its lifetime, from when it is created until it is destroyed. Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class.



- onCreate() is called when the activity is first created, for example when user taps launcher icon.
- onStart() is called when the activity is becoming visible to user.
- onRestart() is called after activity has been stopped, immediately before it is started again.
- onResume() is called when activity will start interacting with user.
- onPause() is called when system is about to resume a previous activity.
- onStop() is called when the activity is no longer visible to the user.
- onDestroy() is called before activity is destroyed.

Q2. Explain Date and Time picker with its methods.

Android Date Picker allows you to select the date consisting of day, month and year in your custom user interface. For this functionality android provides DatePicker and DatePickerDialog components.

1	**getDayOfMonth()** This method gets the selected day of month
2	**getMonth()** This method gets the selected month
3	**getYear()** This method gets the selected year
4	**getCalendarView()** This method returns calendar view
5	**getFirstDayOfWeek()** This Method returns first day of the week

Android Time Picker allows you to select the time of day in either 24 hour or AM/PM mode. The time consists of hours, minutes and clock format. Android provides this functionality through TimePicker class.

1	**is24HourView()** This method returns true if this is in 24 hour view else false
2	**isEnabled()** This method returns the enabled status for this view
3	**setCurrentHour(Integer currentHour)** This method sets the current hour

4	**setCurrentMinute(Integer currentMinute)** This method sets the current minute
5	**setEnabled(boolean enabled)** This method set the enabled state of this view

Q3. List sensors in Android and explain any one in detail.

1. Motion sensors

   These sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors. Motion sensors are useful for monitoring device movement, such as tilt, shake, rotation, or swing. The movement is usually a reflection of direct user input (for example, a user steering a car in a game or a user controlling a ball in a game), but it can also be a reflection of the physical environment in which the device is sitting (for example, moving with you while you drive your car).

2. Environmental sensors.
3. Position sensors.

Q4. Develop an application to send SMS. (Write ONLY .java)

```
package com.example.sendsms;
import android.os.Bundle;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.Intent;
import android.telephony.SmsManager;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity {
   EditText mobileno,message;
   Button sendsms;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        mobileno=(EditText)findViewById(R.id.editText1);
        message=(EditText)findViewById(R.id.editText2);
        sendsms=(Button)findViewById(R.id.button1);
        sendsms.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View arg()) {
                String no=mobileno.getText().toString();
                String msg=message.getText().toString();
                Intent intent=new Intent(getApplicationContext(),MainActivity.class);
                PendingIntent pi=PendingIntent.getActivity(getApplicationContext(), 0, intent,0);
                SmsManager sms=SmsManager.getDefault();
                sms.sendTextMessage(no, null, msg, pi,null);
                Toast.makeText(getApplicationContext(), "Message Sent successfully!",
                    Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

Q5. Develop a simple calculator using table layout.

XML:

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
    <TableRow
        android:id="@+id/tr_first">
        <EditText
            android:id="@+id/et_result"
            android:ems="10"
            android:layout_span="4" />
    </TableRow>
    <TableRow
        android:id="@+id/tr_second" >
        <Button
            android:id="@+id/btn_one"
            android:text="1" />
        <Button
            android:id="@+id/btn_two"
            android:text="2" />
        <Button
            android:id="@+id/btn_three"
            android:text="3" />
        <Button
            android:id="@+id/btn_plus"
```

```
        android:text="+" />
    </TableRow>
     <TableRow
        android:id="@+id/tr_third" >
        <Button
            android:id="@+id/btn_four"
            android:text="4" />
        <Button
            android:id="@+id/btn_five"
            android:text="5" />
        <Button
            android:id="@+id/btn_six"
            android:text="6" />
        <Button
            android:id="@+id/btn_minus"
            android:text="-" />
    </TableRow>
      <TableRow
        android:id="@+id/tr_fourth" >
        <Button
            android:id="@+id/btn_seven"
            android:text="7" />
        <Button
            android:id="@+id/btn_eight"
            android:text="8" />
        <Button
            android:id="@+id/btn_nine"
            android:text="9" />
        <Button
            android:id="@+id/btn_times"
            android:text="*" />
    </TableRow>
      <TableRow
        android:id="@+id/tr_fifth" >
        <Button
            android:id="@+id/btn_zero"
            android:text="0" />
        <Button
            android:id="@+id/btn_clear"
            android:text="C" />
    <Button
            android:id="@+id/btn_equals"
            android:text="=" />
        <Button
            android:id="@+id/btn_divide"
            android:text="/" />
```

```
    </TableRow>
</TableLayout>
```

Java:

```
package com.example.tablelayout;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends Activity implements OnClickListener{
String res, res_plus, res_minus, res_times, res_divide;
 Integer  wswitch;
 Double answer = 0.0;
 EditText result;
 Button one, two, three, four, five, six, seven, eight, nine, zero, plus, minus, times, divide,
clear, equals;
@Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  initializeVars();
 }
public void initializeVars(){
  result = (EditText)findViewById(R.id.et_result);
  one = (Button)findViewById(R.id.btn_one);
  two = (Button)findViewById(R.id.btn_two);
  three = (Button)findViewById(R.id.btn_three);
  four = (Button)findViewById(R.id.btn_four);
  five = (Button)findViewById(R.id.btn_five);
  six = (Button)findViewById(R.id.btn_six);
  seven = (Button)findViewById(R.id.btn_seven);
  eight = (Button)findViewById(R.id.btn_eight);
  nine = (Button)findViewById(R.id.btn_nine);
  zero = (Button)findViewById(R.id.btn_zero);
  plus = (Button)findViewById(R.id.btn_plus);
  minus = (Button)findViewById(R.id.btn_minus);
  times = (Button)findViewById(R.id.btn_times);
  divide = (Button)findViewById(R.id.btn_divide);
  equals = (Button)findViewById(R.id.btn_equals);
  clear = (Button)findViewById(R.id.btn_clear);
  one.setOnClickListener(this);
  two.setOnClickListener(this);
  three.setOnClickListener(this);
```

```
  four.setOnClickListener(this);
  five.setOnClickListener(this);
  six.setOnClickListener(this);
  seven.setOnClickListener(this);
  eight.setOnClickListener(this);
  nine.setOnClickListener(this);
  zero.setOnClickListener(this);
  plus.setOnClickListener(this);
  minus.setOnClickListener(this);
  clear.setOnClickListener(this);
  times.setOnClickListener(this);
  divide.setOnClickListener(this);
  equals.setOnClickListener(this);
}
@Override
 public boolean onCreateOptionsMenu(Menu menu) {
  getMenuInflater().inflate(R.menu.main, menu);
  return true;
 }
@Override
 public void onClick(View arg0) {
  switch (arg().getId()){
  case R.id.btn_one:
   res = result.getText().toString()+"1";
   result.setText(res);
  break;
  case R.id.btn_two:
   res = result.getText().toString()+"2";
   result.setText(res);
  break;
  case R.id.btn_three:
   res = result.getText().toString()+"3";
   result.setText(res);
  break;
  case R.id.btn_four:
   res = result.getText().toString()+"4";
   result.setText(res);
  break;
  case R.id.btn_five:
   res = result.getText().toString()+"5";
   result.setText(res);
  break;
  case R.id.btn_six:
   res = result.getText().toString()+"6";
   result.setText(res);
  break;
```

```
case R.id.btn_seven:
 res = result.getText().toString()+"7";
 result.setText(res);
break;
case R.id.btn_eight:
 res = result.getText().toString()+"8";
 result.setText(res);
break;
case R.id.btn_nine:
 res = result.getText().toString()+"9";
 result.setText(res);
break;
case R.id.btn_zero:
 res = result.getText().toString()+"0";
 result.setText(res);
break;
case R.id.btn_plus:
 res_plus = result.getText().toString();
 result.setText("");
 wswitch = 1;
break;
case R.id.btn_minus:
 res_minus = result.getText().toString();
 result.setText("");
 wswitch = 2;
break;
case R.id.btn_times:
 res_times = result.getText().toString();
 result.setText("");
 wswitch = 3;
break;
case R.id.btn_divide:
 res_divide = result.getText().toString();
 result.setText("");
 wswitch = 4;
break;
case R.id.btn_equals:
 switch(wswitch){
 case 1:
  if(res_plus.matches("") || result.getText().toString().matches("")){
   result.setText("");
  }else{
  answer = Double.parseDouble(res_plus)+Double.parseDouble(result.getText().toString());
  res = Double.toString(answer);
  result.setText(res);
  }
```

```
   break;
   case 2:
    if(res_minus.matches("") || result.getText().toString().matches("")){
     result.setText("");
    }else{
    answer = Double.parseDouble(res_minus)-
Double.parseDouble(result.getText().toString());
    res = Double.toString(answer);
    result.setText(res);
    }
   break;
   case 3:
    if(res_times.matches("") || result.getText().toString().matches("")){
     result.setText("");
    }else{
    answer =
Double.parseDouble(res_times)*Double.parseDouble(result.getText().toString());
    res = Double.toString(answer);
    result.setText(res);
    }
   break;
   case 4:
    if(res_divide.matches("") || result.getText().toString().matches("")){
     result.setText("");
    }else{
    answer =
Double.parseDouble(res_divide)/Double.parseDouble(result.getText().toString());
    res = Double.toString(answer);
    result.setText(res);
    }
   break;
   }
  break;
  case R.id.btn_clear:
   result.setText("");
  break;
  }
 }
}
```

Q6. Define content provider and explain fragments.

- A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. A content provider can use different ways to store its data and the data can be stored in a database, in files, or even over a network. A content provider

behaves very much like a database where you can query it, edit its content, as well as add or delete content using insert(), update(), delete(), and query() methods. In most cases this data is stored in an SQlite database.

- A Fragment represents a behaviour or a portion of user interface in a FragmentActivity. Fragments are the modular section of an activity design. In android, the fragment will act as a sub-activity and we can reuse it in multiple activities. The fragment has its own lifecycle call-backs and fragments can accept its own input events. We can add or remove fragments in an activity while activity is running. If we pause an activity, all the fragments related to an activity will also be stopped.

Q7. Describe Async Task methods.

In Android, AsyncTask (Asynchronous Task) allows us to run the instruction in the background and then synchronize again with our main thread. Mainly we used it for short operations that will not effect on our main thread. In Android, AsyncTask is executed and goes through four different steps or method. Here are these four methods of AsyncTasks:

1. onPreExecute() –

This method is mainly used to setup the task for instance by showing a ProgressBar or ProgressDialog in the User Interface.

2. doInBackground(Params) –

This method is invoked on the background thread immediately after onPreExecute() finishes its execution. Main purpose of this method is to perform the background operations that can take a long time.

3. onProgressUpdate(Progress…) –

This method is used to display any form of progress in the user interface while the background operations are executing. We can also update our progress status for good user experience.

4. onPostExecute(Result) –

This method is invoked on the main UI thread after the background operation finishes in the doInBackground method.

Q8. Explain Android Service in detail.

Android service is a component that is used to perform operations on the background such as playing music, handle network transactions, interacting content providers etc. It doesn't have any User Interface. The service runs in the

background indefinitely even if application is destroyed. Moreover, service can be bounded by a component to perform interactivity and inter process communication (IPC).

There can be two forms of a service. The lifecycle of service can follow two different paths: started or bound.

1) Started Service

A service is started when component calls startService() method, now it runs in the background indefinitely. It is stopped by stopService() method. The service can stop itself by calling the stopSelf() method.

2) Bound Service

A service is bound when another component calls bindService() method. The client can unbind the service by calling the unbindService() method.

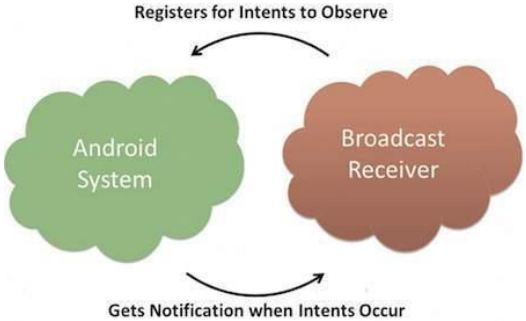
The service cannot be stopped until all clients unbind the service

Q9. Explain Broadcast Receiver in detail.

Broadcast Receivers simply respond to broadcast messages from other applications or from the system itself. These messages are sometime called events or intents. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use.

There are following two important steps to make BroadcastReceiver works for the system broadcasted intents −

1. Creating the Broadcast Receiver: A broadcast receiver is implemented as a subclass of BroadcastReceiver class and overriding the onReceive() method where each message is received as a Intent object parameter.
2. Registering Broadcast Receiver: An application listens for specific broadcast intents by registering a broadcast receiver in AndroidManifest.xml file. Consider, we are going to register MyReceiver for system generated event ACTION_BOOT_COMPLETED which is fired by the system once the Android system has completed the boot process.



Q10. Write a program to demonstrate Custom Toast.

File: activity_main.xml

```
<RelativeLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent" >
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello World!" />
</RelativeLayout>
```

File: customtoast.xml

```
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:id="@+id/custom_toast_layout"
   android:orientation="vertical" >
   <ImageView
      android:src="@drawable/logo"/>
   <TextView
      android:text="Custom Toast" />
</LinearLayout>
```

File: MainActivity.java

```
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      LayoutInflater li = getLayoutInflater();
View layout = li.inflate(R.layout.customtoast,(ViewGroup)
findViewById(R.id.custom_toast_layout));
      Toast toast = new Toast(getApplicationContext());
      toast.setDuration(Toast.LENGTH_SHORT);
      toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
      toast.setView(layout);
      toast.show();
}
}
```

Q11. Write a program to capture an image using Camera and display it.

Please add CAMERA permission inside your AndroidManifest.xml file:

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

activity_main.xml

```
<LinearLayout
android:id="@+id/activity_main"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<ImageView
android:layout_centerHorizontal="true"
android:id="@+id/imageView" />
<Button
android:text="Click here to capture image using camera"
android:id="@+id/button" />
</LinearLayout>
```

MainActivity.java file

```
import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
Button button;
ImageView imageView;
Intent intent;
public static final int RequestPermissionCode  = 1;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
button = (Button)findViewById(R.id.button);
imageView = (ImageView)findViewById(R.id.imageView);
EnableRuntimePermission();
button.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
intent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
```

```
startActivityForResult(intent, 7);
}
});
}
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
if (requestCode == 7 && resultCode == RESULT_OK) {
Bitmap bitmap = (Bitmap) data.getExtras().get("data");
imageView.setImageBitmap(bitmap);
}
}
public void EnableRuntimePermission(){
if (ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
Manifest.permission.CAMERA)) {
Toast.makeText(MainActivity.this,"CAMERA permission allows us to Access CAMERA
app", Toast.LENGTH_LONG).show();
} else {
ActivityCompat.requestPermissions(MainActivity.this,new String[]{
Manifest.permission.CAMERA}, RequestPermissionCode);
}
}
@Override
public void onRequestPermissionsResult(int RC, String per[], int[] PResult) {
switch (RC) {
case RequestPermissionCode:
if (PResult.length > 0 && PResult[0] == PackageManager.PERMISSION_GRANTED) {
Toast.makeText(MainActivity.this,"Permission Granted, Now your application can access
CAMERA.", Toast.LENGTH_LONG).show();
} else {
Toast.makeText(MainActivity.this,"Permission Canceled, Now your application cannot
access CAMERA.", Toast.LENGTH_LONG).show();
}
break;
}
}
}
```

Q12. Write a program to TURN ON and TURN OFF Bluetooth.

Set Bluetooth permissions in our android manifest file:

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

activity_main.xml

```
<RelativeLayout
android:layout_width="match_parent"
android:layout_height="match_parent" >
<Button
android:id="@+id/btnOn"
android:text="Turn On" />
<Button
android:id="@+id/btnOFF"
android:text="Turn OFF" />
</RelativeLayout>
```

MainActivity.java

```
import android.bluetooth.BluetoothAdapter;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
Button btntOn = (Button)findViewById(R.id.btnOn);
Button btntOff = (Button)findViewById(R.id.btnOFF);
final BluetoothAdapter bAdapter = BluetoothAdapter.getDefaultAdapter();
btntOn.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
if(bAdapter == null) {
Toast.makeText(getApplicationContext(),"Bluetooth Not
Supported",Toast.LENGTH_SHORT).show();
}
else{
if(!bAdapter.isEnabled()){
startActivityForResult(new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE),1);
Toast.makeText(getApplicationContext(),"Bluetooth Turned
ON",Toast.LENGTH_SHORT).show();
}
}
}
});
btntOff.setOnClickListener(new View.OnClickListener() {
@Override
```

```
public void onClick(View v) {
bAdapter.disable();
Toast.makeText(getApplicationContext(),"Bluetooth Turned OFF",
Toast.LENGTH_SHORT).show();
}
});
}
}
```

Q13. Write a program to list out country name and Toast the selected country. Make use of List View.

Java:

```
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.AdapterView;
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
final String[] mobile = {"India","Japan","USA","Australia","Sri
Lanka","Pakistan","Bhutan","South Africa","New Zealand","Russia"};
ArrayAdapter adapter = new ArrayAdapter<String>(this, R.layout.activity_listview, mobile);
ListView list1 = (ListView) findViewById(R.id.list1);
list1.setAdapter(adapter);
list1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
Toast.makeText(MainActivity.this, ((TextView) view).getText(),
Toast.LENGTH_SHORT).show();
} });}}
```

XML:

```
<LinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:id="@+id/l1"
android:orientation="vertical">
<ListView
android:id="@+id/list1" />
</LinearLayout>
```

Listview:

```
<TextView
android:id="@+id/label"
android:layout_width="fill_parent"
android:layout_height="fill_parent" >
</TextView>
```

Q14. Write a program to create button “Start Dialer”. When u clicks it should open the phone dialer.

Java:

```
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class MainActivity extends AppCompatActivity
{
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
Button b1 = (Button) findViewById(R.id.b1);
b1.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Intent i1 = new Intent(Intent.ACTION_DIAL); startActivity(i1);
}
});
}
}
```

XML:

```
<LinearLayout
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical">
<Button
android:id="@+id/b1"
android:text="Start Dialer" />
</LinearLayout>
```

Q15. Write a program to Start and Stop music on button click event in the background.

MainActivity.java:

```
import  androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
   private Button buttonStart;
   private Button buttonStop;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      buttonStart = (Button) findViewById(R.id.buttonStart);
      buttonStop = (Button) findViewById(R.id.buttonStop);
      buttonStart.setOnClickListener(this);
      buttonStop.setOnClickListener(this);
   }
   @Override
   public void onClick(View view) {
      if (view == buttonStart) {
         startService(new Intent(this, MyService.class));
         Toast.makeText(this, "Activity Started", Toast.LENGTH_SHORT).show();
      } else if (view == buttonStop) {
         stopService(new Intent(this, MyService.class));
         Toast.makeText(this, "Activity Stopped", Toast.LENGTH_SHORT).show();
      }
   }
}
```

MyService.java:

```
import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;
import android.provider.Settings;
public class MyService extends Service {
   private MediaPlayer player;
   @Override
   public IBinder onBind(Intent intent) {
```

```
        return null;
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        player = MediaPlayer.create(this, Settings.System.DEFAULT_RINGTONE_URI);
        player.setLooping(true);
        player.start();
        return START_STICKY;
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        //stopping the player when service is destroyed
        player.stop();
    }
}
```

Adding Service in Manifest:

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.intent.category.LAUNCHER" />
```

XML:

```
<LinearLayout
android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:orientation="vertical"
>
<Button
android:id="@+id/buttonStart"
android:text="Start Activity"
/>
<Button
android:id="@+id/buttonStop"
android:text="Stop Activity"/>
</LinearLayout>
```