CMR
INSTITUTE OF
TECHNOLOGY

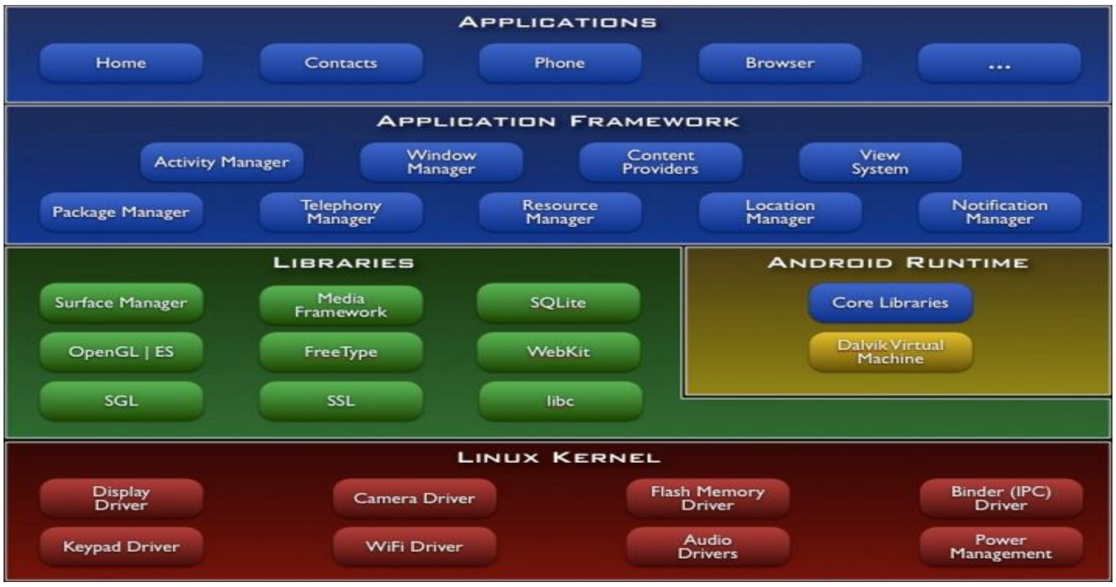
USN

Internal Assessment Test 2 Answer Key– December. 2021									
Sub:	Mobile Applications					Sub Code:	18MCA52	Branch:	MCA
Date:	16/12/2021	Duration:	90 min's	Max Marks:	50	Sem	IV		

Q1)What is android? Explain the android architecture with its features and diagram.

- Android is a mobile operating system that is based on a modified version of Linux.
- It was originally developed by a start-up of the same name, Android, Inc.
- Android is open and free; most of the Android code was released under the open-source Apache License, which means that anyone who wants to use Android can do so by downloading the full Android source code.
- The main advantage of adopting Android is that it offers a unified approach to application development.
- Android OS is a Linux-based open source platform for mobile, cellular handsets developed by Google and the Open Handset Alliance

The Android OS is roughly divided into five sections in four main layers as shown in Figure



Linux kernel: This is the kernel on which Android is based. This layer contains all the low level device drivers for the various hardware components of an Android device.

Libraries:

- These contain all the code that provides the main features of an Android OS. For example, the SQLite library provides database support so that an application can use it for data storage.
- The WebKit library provides functionalities for web browsing.

Android runtime :

- At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language.
- The Android runtime also includes the Dalvik virtual machine, which enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine (Android applications are compiled into the Dalvik executables).
- Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU.

Application Framework: Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications.

Applications:

- At this top layer, you will find applications that ship with the Android device (such as Phone, Contacts, Browser, etc.), as well as applications that you download and install from the Android Market.
- Any applications that you write are located at this layer.

Features of Android

As Android is open source and freely available to manufacturers for customization, there are no fixed hardware and software configurations. However, Android itself supports the following features:

- Storage : Uses SQLite, a lightweight relational database, for data storage.
- Connectivity : Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), WiFi, LTE, and WiMAX.
- Messaging: Supports both SMS and MMS.
- Web browser: Based on the open-source WebKit, together with Chrome's V8 JavaScript engine
- **Media support:** Includes support for the following media: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP
- **Hardware support:** Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS
- **Multi-touch:** Supports multi-touch screens
- **Multi-tasking:** Supports multi-tasking applications
- **Flash support:** Android 2.3 supports Flash 10.1.
- **Tethering:** Supports sharing of Internet connections as a wired/wireless hotspot

Q2)Describe the anatomy of the android application

The various folders and their files are as follows:

- **src** — Contains the file, ***MainActivity.java***. It is the source file for your activity. You will write the code for your application in this file.
- **Android 4.4.2** — This item contains one file, ***android.jar***, which contains all the class libraries needed for an Android application.
- **gen** — Contains the ***R.java*** file, a compiler-generated file that references all the resources found in your project. ***You should not modify this file.***
- **assets** — This folder contains all the assets used by your application, such as HTML, text files, databases, etc.
- **res** — This folder contains all the resources used in your application. It also contains a few other subfolders:
  - **drawable** - **<resolution>**: All the image files to be used by the Android application must

be stored here.

    - **layout -** contains **activity_main.xml** file, which the is GUI of the application.
    - **values -** contains files like **strings.xml, styles.xml** that are need for storing the string variables used in the applications, creating style-sheets etc.
- **AndroidManifest.xml** — This is the manifest file for your Android application. Here you specify the permissions needed by your application, as well as other features (such as intent-filters, receivers, etc.).

Details of some of the important files are given hereunder:

- **strings.xml File:** The activity_main.xml file defines the user interface for your activity. Observe the following in bold:

```
<TextView android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”@string/hello” />
```

  The ***@string*** in this case refers to the strings.xml file located in the res/values folder. Hence, ***@string/hello*** refers to the hello string defined in the ***strings.xml*** file, which is “Hello World!”:

```
<?xml version=”1.0” encoding=”utf-8”?>
<resources>
<string name=”hello”>Hello World!</string>
<string name=”app_name”>HelloWorld</string>
</resources>
```

  It is recommended that you store all the string constants in your application in this ***strings.xml*** file and reference these strings using the ***@string*** identifier. That way, if you ever need to localize your application to another language, all you need to do is replace the strings stored in the ***strings.xml*** file with the targeted language and recompile your application.

- **AndroidManifest.xml File:** This file contains detailed information about the application. Observe the code in this file:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.HelloWorld"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="19" />

    <application
        android:allowBackup="true"

        android:icon="@drawable/ic_launcher" android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
```

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

Key points about this file are as below :

- It defines the package name of the application as ***net.learn2develop.HelloWorld***.
- The version code of the application is 1. This value is used to identify the version number of your application. It can be used to programmatically determine whether an application needs to be upgraded.
- The version name of the application is 1.0. This string value is mainly used for display to the user.
- The application uses the image named ***ic_launcher.png*** located in the ***drawable*** folder.
- The name of this application is the string named **app_name** defined in the ***strings.xml*** file.
- There is one activity in the application represented by the ***MainActivity.java*** file. The label displayed for this activity is the same as the application name.
- Within the definition for this activity, there is an element named <intent-filter>:
    - The action for the intent filter is named ***android.intent.action.MAIN*** to indicate that this activity serves as the entry point for the application.
    - The category for the intent-filter is named ***android.intent.category.LAUNCHER*** to indicate that the application can be launched from the device's Launcher icon.
- Finally, the ***android:minSdkVersion*** attribute of the <uses-sdk> element specifies the minimum version of the OS on which the application will run.

- **R.java File:** As you add more files and folders to your project, Eclipse will automatically generate the content of **R.java**, which at the moment contains the following:

```
package net.learn2develop.HelloWorld;

public final class R {
public static final class attr {
}

        public static final class drawable {
                public static final int icon=0x7f020000;
}
public static final class layout {
        public static final int main=0x7f030000;
}
public static final class string {
        public static final int app_name=0x7f040001; public static final int
        hello=0x7f040000;
}
}
```

You are not supposed to modify the content of the R.java file; Eclipse automatically generates the content for you when you modify your project.

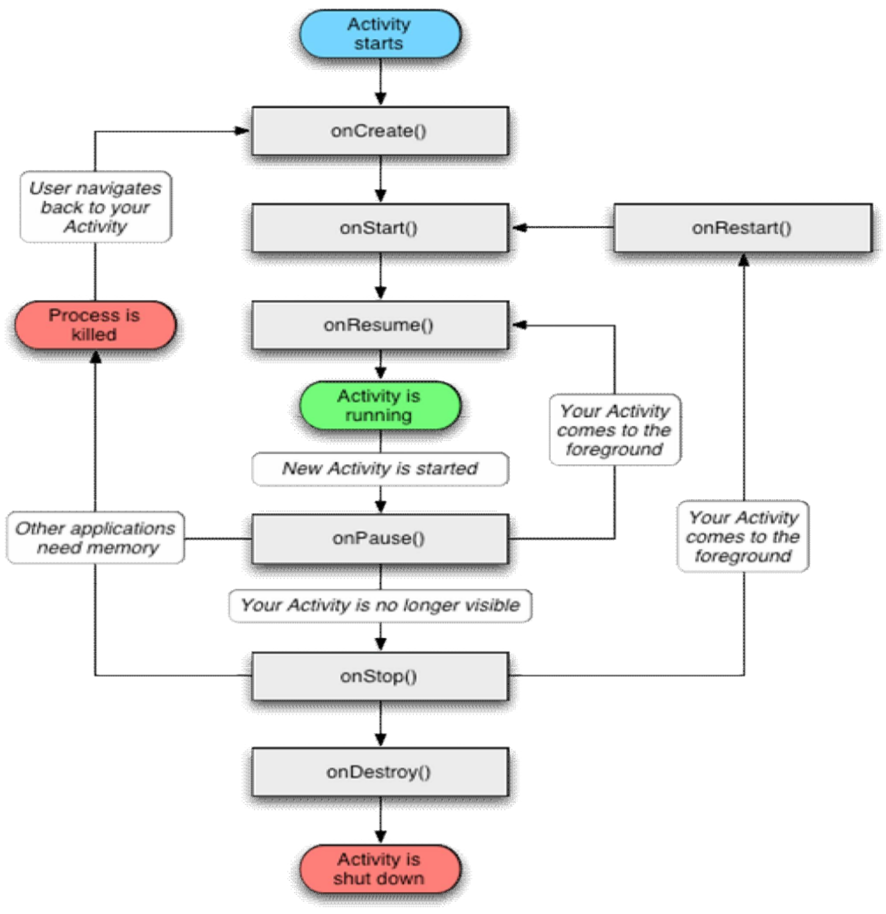
- **MainActivity.java File:** The code that connects the activity to the UI (activity_main.xml) is the setContentView() method, which is in the MainActivity.java file:

```
package net.learn2develop.HelloWorld; import
android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity
{
        /** Called when the activity is first created. */ @Override
        public void onCreate(Bundle savedInstanceState)
        {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
        }
}
```

Here, **R.layout.main** refers to the **activity_main.xml** file located in the **res/layout** folder. As you add additional XML files to the **res/layout** folder, the filenames will automatically be generated in the **R.java** file. The **onCreate()** method is one of many methods that are fired when an activity is loaded.

Q3)Explain briefly life cycle of an activity with diagram and code snippets

The Activity base class defines a series of events that governs the life cycle of an activity. Figure 2.2 shows the life cycle of an activity and the various stages it goes through — from when the activity is started until it ends.



Activity starts
onCreate()
User navigates back to your Activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
Your Activity comes to the foreground
New Activity is started
Other applications need memory
onPause()
Your Activity comes to the foreground
Your Activity is no longer visible
onStop()
onDestroy()
Activity is shut down

The Activity class defines the following events:

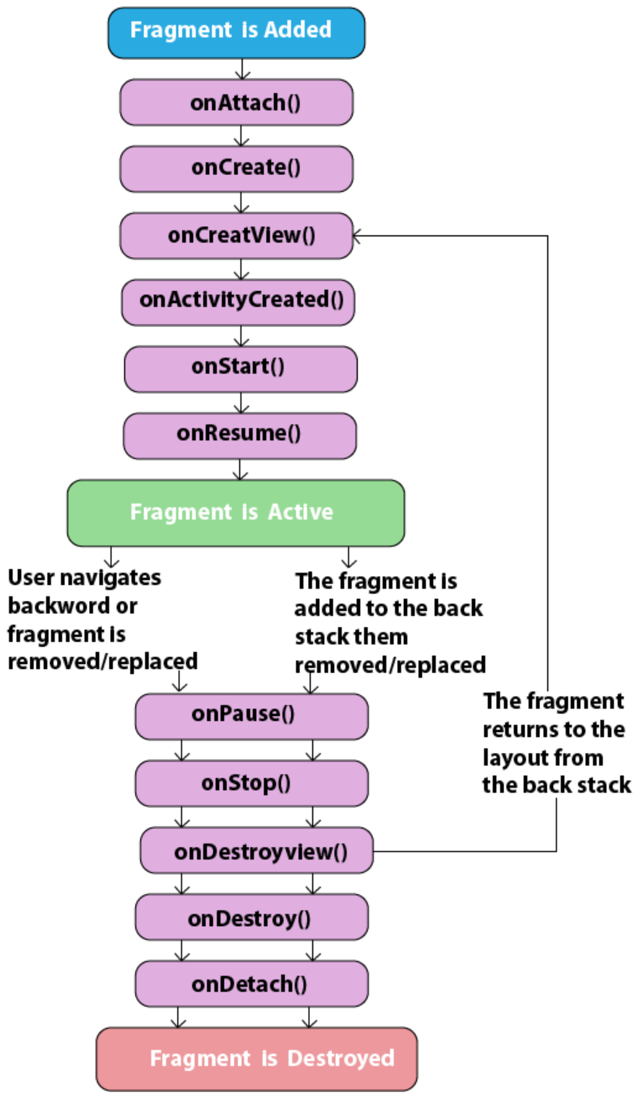
- onCreate() — Called when the activity is first created
- onStart() — Called when the activity becomes visible to the user
- onResume() — Called when the activity starts interacting with the user
- onPause() — Called when the current activity is being paused and the previous activity is being resumed
- onStop() — Called when the activity is no longer visible to the user
- onDestroy() — Called before the activity is destroyed by the system (either manually or by the system to conserve memory)
- onRestart() — Called when the activity has been stopped and is restarting again

By default, the activity created for you contains the onCreate() event. This event handler contains the code that helps to display the UI elements of your screen.

```
import android.app.Activity; import
android.os.Bundle; import
android.util.Log;

public class MainActivity extends Activity
{
        String tag = “Events”;
        /** Called when the activity is first created. */ @Override
        public void onCreate(Bundle savedInstanceState)
        {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main); Log.d(tag, “In the
                onCreate() event”);
        }

        public void onStart()
      {
              super.onStart();
              Log.d(tag, "In the onStart() event");
      }
      public void onRestart()
      {
                super.onRestart();
              Log.d(tag, "In the onRestart() event");
      }

      public void onResume()
      {
                super.onResume();
                Log.d(tag, "In the onResume() event");
      }
      public void onPause()
      {
              super.onPause();
              Log.d(tag, "In the onPause() event");
      }
      public void onStop()
      {
            super.onStop();
            Log.d(tag, "In the onStop() event");
        }
        public void onDestroy()
        {
            super.onDestroy();
            Log.d(tag, "In the onDestroy() event");
        }

    }
```

Q4)Explain briefly life cycle of an Fragment with diagram and code snippets

The lifecycle events of a Fragment reflect those of its parent Activity. But, when the container Activity is in its active and resumed state by adding or removing a fragment, it will affect the lifecycle independently. Figure shows various events in lifecycle of fragments.



```
import android.app.Activity;
import android.os.Bundle;
import android.app.Fragment;
import android.view.View;
import android.view.ViewGroup;
import android.view.LayoutInflater;
 import android.util.Log;

public class MainActivity extends Fragment
{
        @Override
        public void onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
        saved InstanceState)
        {
                Log.d(“Fragment 1”, “on CreateView”);
                Return
                inflater.inflate(R.layout.fragment1,container,false);
        }

        @Override

        public void onAttach(Activity activity)
        {
              super.onAttach(activity);
              Log.d(“Fragment 1”, "onAttach");
        }


        public void onCreate (Bundle savedInstanceState)
        {
                super.onCreate(savedInstanceState);
              Log.d(“Fragment 1”, "onCreate");

        }
        public void onActivityCreated (Bundle savedInstanceState)
        {
                super.onActivityCreated(savedInstanceState);
              Log.d(“Fragment 1”, "onActivityCreated");

        }


        public void onStart()
      {
              super.onStart();
              Log.d(“Fragment 1”,  "In the onStart");
      }

      public void onResume()
      {
                super.onResume();
                Log.d(“Fragment 1”, "In the onResume ");
      }
      public void onPause()
```

```
        {
                super.onPause();
                Log.d("Fragment 1", "In the onPause ");
        }
        public void onStop()
        {
                super.onStop();
                Log.d("Fragment 1", "In the onStop);
        }
        public void onDestroyView()
        {
                super.onDestroyView();
                Log.d("Fragment 1", "In the onDestroyView ");
        }
        public void onDestroy()
        {
                super.onDestroy();
                Log.d("Fragment 1", "In the onDestroy ");
        }
        public void onDetach()
        {
                super.onDetach();
                Log.d("Fragment 1", "In the onDetach ");
        }


    }
```

Like activities, fragments in android also have their own life cycle. As you have seen, when a fragment is being created, it goes through the following states:
onAttach()
onCreate()
onCreateView()
onActivityCreated()

When the fragment becomes visible, it goes through these states:
onStart()
onResume()

when the fragment goes into the background mode, it goes through these states:
onPause()
onStop()

when the fragment is destroyed (when the activity is currently hosted in is destroyed), it goes through the following states:
onPause()
onStrop()
onDestroyView()
onDestroy()
onDetach()

Like activities, you can restore an instance of fragment using a Bundle object, in the following states:
onCreate()
onCreateView()
onActivityCreated()

Most of the states experienced by a fragment are similar to those of activities. However, a few new states are specific to fragment
onAttached() – Called when fragment has been associated with the activity
onCreateView() – Called to create the view for the fragment
onActivityCreated()-Called when activit's onCreate() method has been returned.
onDestroyView() – Called when the fragment's view is being removed
onDetach() – Called when the fragment is detached from the activity

Q5) Develop a mobile application to display notification.

Code for Activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10dp"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Message"
        android:textSize="30sp" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:singleLine="true"
        android:textSize="30sp" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:layout_gravity="center"
        android:text="Notify"
        android:textSize="30sp"/>

</LinearLayout>
```

Code for MainActivity.java:

```
package com.example.ashwini.program9;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity
{
    Button notify;
    EditText e;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        notify= (Button) findViewById(R.id.button);
        e= (EditText) findViewById(R.id.editText);

        notify.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                Intent intent = new Intent(MainActivity.this, SecondActivity.class);
                PendingIntent pending = PendingIntent.getActivity(MainActivity.this, 0, intent,
0);
                Notification noti = new
Notification.Builder(MainActivity.this).setContentTitle("New
Message").setContentText(e.getText().toString()).setSmallIcon(R.mipmap.ic_launcher).setContentIn
tent(pending).build();
                NotificationManager manager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
                noti.flags |= Notification.FLAG_AUTO_CANCEL;
                manager.notify(0, noti);
            }
        });
    }
}
```

Q6) Develop a mobile application to pass the data using intent object

Activity2.xml

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

    <TextView
        android:id="@+id/textView1"
        android:text="Activity2" />

    <TextView
        android:id="@+id/tname"
        android:text="TextView" />

    <TextView
        android:id="@+id/tpass"
        android:text="TextView" />

    <TextView
        android:id="@+id/tage"
        android:text="TextView" />

</RelativeLayout>
```

Activity_main.xml

```
<RelativeLayout>

    <TextView
        android:id="@+id/textView1"
        android:text="Activity1" />

    <TextView
        android:id="@+id/textView2"
        android:text="Name" />

    <EditText
        android:id="@+id/ename"
        android:ems="10" >
    </EditText>

    <EditText
        android:id="@+id/epass"
        android:ems="10" />

    <TextView
        android:id="@+id/textView3"
        android:text="Pass" />

    <TextView
        android:id="@+id/textView4"
        android:text="Age" />

    <EditText
        android:id="@+id/eage"
        android:ems="10" />

    <Button
        android:id="@+id/button1"
        android:text="Button" />

</RelativeLayout>
```

Code for Activity2.java:

```
package com.example.bsec_intentpassdata;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Activity2 extends Activity{
     TextView tname, tage, tpass;

     @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity2);
        tname=(TextView) findViewById(R.id.tname);
        tpass=(TextView) findViewById(R.id.tpass);
        tage=(TextView) findViewById(R.id.tage);


        tname.setText(getIntent().getStringExtra("name"));
        tpass.setText(getIntent().getStringExtra("pass"));

tage.setText(Integer.toString(getIntent().getIntExtra("age",0)));

    }
}
```

Code for MainActivity.java:

```
package com.example.bsec_intentpassdata;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;


public class MainActivity extends Activity {
     EditText ename, epass, eage ;
     Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

       btn=(Button) findViewById(R.id.button1);
       ename=(EditText) findViewById(R.id.ename);
       epass=(EditText) findViewById(R.id.epass);
       eage=(EditText) findViewById(R.id.eage);

       btn.setOnClickListener(new OnClickListener() {

          @Override
          public void onClick(View v) {
               // TODO Auto-generated method stub
               String uname=ename.getText().toString();
               String upass=epass.getText().toString();
               String age=eage.getText().toString();
               int uage=Integer.parseInt(age);

               //without bundle
               Intent i = new Intent(MainActivity.this,
Activity2.class);
               i.putExtra("name", uname);
               i.putExtra("pass", upass);
               i.putExtra("age", uage);
               startActivity(i);
          }
     });
    }


}
```

Q7) Explain basic view in android with suitable code snippets

A view is a widget that has an appearance on screen. Basic views used to design the UI of your Android applications. These basic views enable you to display text information, as well as perform some basic selection.

- TextView
- EditText
- Button
- ImageButton
- CheckBox
- ToggleButton
- RadioButton
- RadioGroup

TextView

The TextView view is used to display text to the user.

```
<TextView
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”@string/hello”
/>
```

EditText
A subclass of the TextView view, except that it allows users to edit its text content

```
<EditText android:id=”@+id/txtName”
 android:layout_width=”fill_parent”
 android:layout_height=”wrap_content” />
```

Button
Represents a push-button widget

```
<Button android:id=”@+id/btnSave”
 android:layout_width=”fill_parent”
 android:layout_height=”wrap_content”
 android:text=”Save” />
```

ImageButton
Similar to the Button view, except that it also displays an image

```
<ImageButton android:id=”@+id/btnImg1”
 android:layout_width=”fill_parent”
 android:layout_height=”wrap_content”
 android:src=”@drawable/icon” />
```

CheckBox

A special type of button that has two states: checked or unchecked

```
<CheckBox android:id="@+id/chkAutosave"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="Autosave" />
```

RadioGroup and RadioButton — The RadioButton has two states: either checked or unchecked. Once a RadioButton is checked, it cannot be unchecked. A RadioGroup is used to group together one or more RadioButton views, thereby allowing only one RadioButton to be checked within the RadioGroup.

```
<RadioGroup android:id="@+id/rdbGp1"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical" >
            <RadioButton android:id="@+id/rdb1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Option 1" />
            <RadioButton android:id="@+id/rdb2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Option 2" />
 </RadioGroup>
```

ToggleButton — Displays checked/unchecked states using a light indicator

```
<ToggleButton android:id="@+id/toggle1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content" />
```

Q8) Write a note on Date Picker rand Time Picker Views.

TimePicker
The TimePicker displays a standard UI to enable users to set a time. By default, it displays the time in the AM/PM format. If you wish to display the time in the 24-hour format, you can use the setIs24Hour View() method.
To programmatically get the time set by the user, use the getCurrentHour() and getCurrentMinute() methods:

DatePicker
Another view that is similar to the TimePicker is the DatePicker. Using the DatePicker, you can enable users to select a particular date on the activity.
Like the TimePicker, you call the getMonth(), getDayOfMonth(), and getYear() methods to get the month, day, and year, respectively:

```
"Date selected:" + datePicker.getMonth() + 1 +
"/" + datePicker.getDayOfMonth() +
"/" + datePicker.getYear() + "\n" +
```

The getMonth()method returns 0 for January, 1 for February, and so on. Hence, you need to add a one to the result of this method to get the month number.

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent" >

        <DatePicker android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

        <TimePicker android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

        <Button android:id="@+id/btnSet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I am all set!" />

</LinearLayout>
```

<u>MainActivity.java Code</u>

```
        import android.app.Activity;
        import android.os.Bundle;
        import android.view.View;
        import android.widget.Button;
        import android.widget.TimePicker;
        import android.widget.DatePicker;
        import android.widget.Toast;

        public class MainActivity extends Activity {
                TimePicker timePicker;
                /** Called when the activity is first created. */
                @Override
                public void onCreate(Bundle savedInstanceState) {
                        super.onCreate(savedInstanceState);
                        setContentView(R.layout.main);
                        timePicker = (TimePicker) findViewById(R.id.timePicker);
                        timePicker.setIs24HourView(true);
                        datePicker = (DatePicker) findViewById(R.id.datePicker);
                        //---Button view---
                        Button btnOpen = (Button) findViewById(R.id.btnSet);
                        btnOpen.setOnClickListener(new View.OnClickListener() {

                                public void onClick(View v) {
                                        Toast.makeText(getBaseContext(),
                                        "Date selected:" + datePicker.getMonth() + 1 +
                                        "/" + datePicker.getDayOfMonth() +
                                        "/" + datePicker.getYear() + "\n" +
                                        "Time selected:" +
                                        timePicker.getCurrentHour() +
                                        ":" + timePicker.getCurrentMinute(),
                                        Toast.LENGTH_SHORT).show();
                                }
                        });
```

```
            }
        }
```

Q9) Develop a mobile application to list the tourist places of Karnataka using List View.

Activity_mail.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.lab5.MainActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:text="Karnataka Tourist Places" />

    <ListView
        android:id="@+id/listView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="14dp" >
    </ListView>

</RelativeLayout>
```

MainActivity.java

```
package com.example.lab5;


import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;


public class MainActivity extends Activity {

      String[] places = {
                   "Coorg",
                   "Hampi",
                   "Gokarna",
                   "Bandipur National Park",
                   "Jog Falls",
                   "Nandi Hills",
                   "Dharmashala",
                   "Mysore",
                   "Chikmagalur",
                   "Bengaluru",
                   "Badami",
                   "Nagarhole National Park"
                   };
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ArrayAdapter adapter= (new ArrayAdapter<String>(this,
                    android.R.layout.simple_list_item_1, places));
       ListView listview=(ListView) findViewById(R.id.listView1);
       listview.setAdapter(adapter);
    }



}
```

Q10) Devise an application that draws basic graphical primitives (rectangle, circle) on the screen.

Code for Activity_main.xml:

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

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/imageView" />
</RelativeLayout>
```

Code for MainActivity.java:

```
package com.example.ashwini.program4;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.drawable.BitmapDrawable;
import android.os.Bundle;
import android.widget.ImageView;

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

        //Creating a Bitmap
        Bitmap bg = Bitmap.createBitmap(720, 1280, Bitmap.Config.ARGB_8888);

        //Setting the Bitmap as background for the ImageView
        ImageView i = (ImageView) findViewById(R.id.imageView);
        i.setBackgroundDrawable(new BitmapDrawable(bg));

        //Creating the Canvas Object
        Canvas canvas = new Canvas(bg);

        //Creating the Paint Object and set its color & TextSize
        Paint paint = new Paint();
        paint.setColor(Color.BLUE);
        paint.setTextSize(50);

        //To draw a Rectangle
        canvas.drawText("Rectangle", 420, 150, paint);
        canvas.drawRect(400, 200, 650, 700, paint);

        //To draw a Circle
        canvas.drawText("Circle", 120, 150, paint);
        canvas.drawCircle(200, 350, 150, paint);

    }
}
```