SUMMER – 2024 EXAMINATION
Model Answer – Only for the Use of RAC Assessors

Subject Name: Mobile Application Development **Subject Code:** 22617

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.
8) As per the policy decision of Maharashtra State Government, teaching in English/Marathi and Bilingual (English + Marathi) medium is introduced at first year of AICTE diploma Programme from academic year 2021-2022. Hence if the students in first year (first and second semesters) write answers in Marathi or bilingual language (English +Marathi), the Examiner shall consider the same and assess the answer based on matching of concepts with model answer.

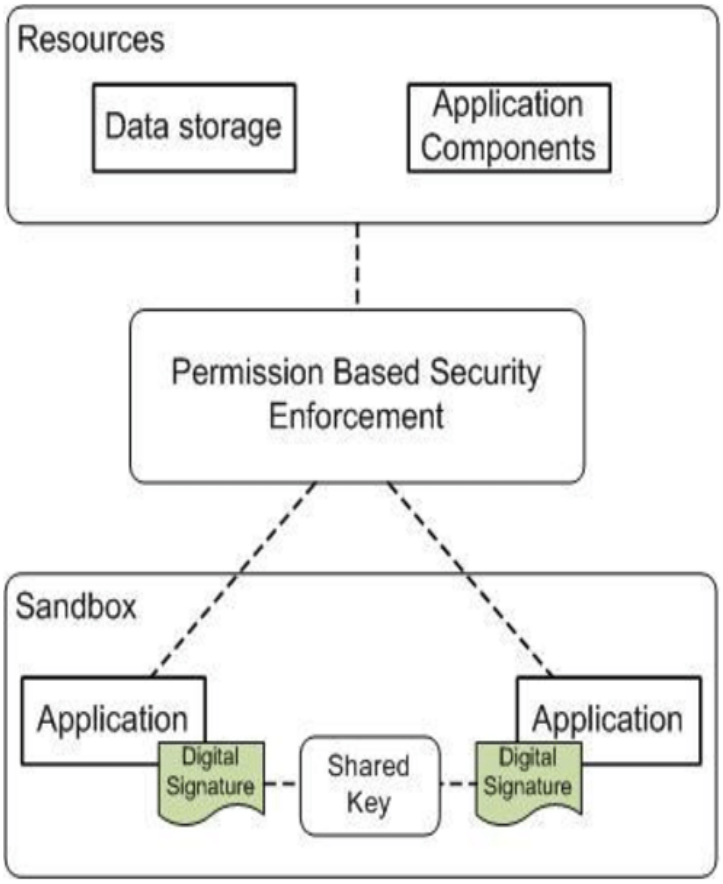
Q. No.	Sub Q. N.	Answer	Marking Scheme
1		**Attempt any FIVE of the following:**	**10 M**
	a)	**List any four features of android operating system.**	**2 M**
	Ans	**Features of Android Operating System:** 1)Storage 2)Multitasking 3)Web Browser 4)Open Source 5)Accessibility 6)Media Support 7)Streaming Media Support 8)Voice Based Features 9)Multitouch 10)External Storage 11)Video Calling 12)Handset Layout 13)Google cloud Messaging 14)WiFi Direct	Any 4 one for ½ M
	b)	**Describe role of Emulator.**	**2 M**

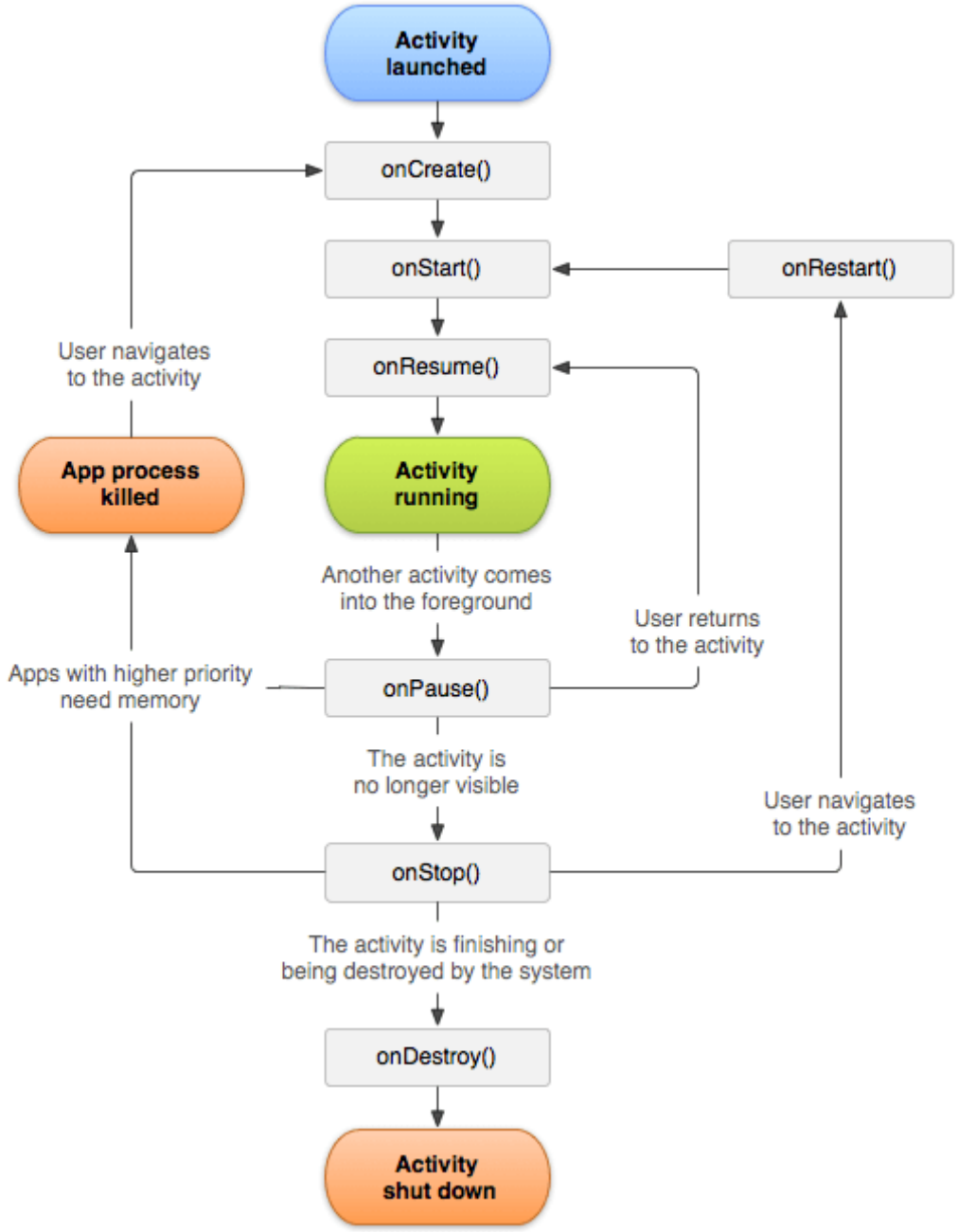
	Ans	Android emulator is a tool that creates virtual Android devices on your computer. The emulator lets you prototype, develop and test Android applications without using a physical device	Correct definition 2 M
	c)	**List various components of android UI design.**	**2 M**
	Ans	Components of android UI design: 1)views 2)viewgroups 3)fragments 4)activity	One for ½ M
	d)	**List any two attributes of Toggle Button.**	**2 M**
	Ans	1. android:textOff 2. android:textOn 3. android:id 4. android:checked 5. android:gravity 6. android:textColor 7. android:textSize 8. android: textStyle	any 2 attributes , one attribute for one M
	e)	**Define service in android operating system.**	**2 M**
	Ans	A service is an application component which runs without direct interaction with the user in the background. Services are used for repetitive and potentially long running operations, i.e., Internet downloads, checking for new data, data processing, updating content providers and the like.	Correct definition 2 M
	f)	**Explain two methods of Google Map.**	**2 M**
	Ans	getMyLocation(): This method returns the currently displayed user location. moveCamera(CameraUpdate update): This method reposition the camera according to the instructions defined in the update.	One method for 1 M
	g)	**Write down syntax to create an intent and start another activity.**	**2 M**

	Ans	Intent i = new Intent(this, ActivityTwo.class); //create intent
		startActivity(i); //start activity
			create intent for 1 M and start activity 1 M

2.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Explain need of android.**	**4 M**
	Ans	Android is a new generation mobile OS which runs on Linux kernel. There are some following points which describe why we use Android OS: **1. Desktop:** The Android phone adds widgets to the desktop. The purpose for the widget, for example, the Facebook widget allows us to update our facebook desktop. The people widget allows us to make possible different actions for different contacts right from our desktop. The message widget allows us to immediately see our e-mail from the desktop. **2. Connectivity:** On one page/desktop we could be able to have four connecting device tool button like, Turn ON/OFF Bluetooth, Turn ON/OFF WiFi, Turn ON/OFF mobile network, Turn On/Off GPS and so on. These buttons let us switch ON/OFF instantly which will help us to conserve battery life. **3. Browser:** The Android OS browser is one of the best browsers on the mobile market. It generally loads pages faster than Safari or any other browser, has Flash support and simply does everything a browser should do. For example, iPhone has Safari browser. It is stable, has no Flash support, so we cannot watch Youtube videos or any related contents, it is not flexible but monopolistic. **4. Open to Carrier:** If we know Java programming language then we are open to Android world. **5. Market**: Android OS has an android market. The android apps are free and work as well. **6. Future:** The future mobile phones are basically going to be smart phones. **7. Muti-Notification**: Android phones have multi-notification system. With android the app have access to the notification system and call all report. **8. Google Integration**: The Android has inbuilt google support. For examle, Google Map, G-Mail etc. **9. Open Source**: The code of android OS as well as the apps is available. **10. Endless Personalization**: The Android cell phone allows to configuration their mobile to look and behave exactly like they want.	Any four points, 4 M
	b)	**Compare JVM and DVM. (any four points)**	**4 M**
	Ans	<table><tr><td>JVM</td><td>DVM</td></tr></table>	Any four points 4 M, 1 M for one point)

		JVM supports multiple operating systems.	DVM supports only Android Operating System.	
		JVM forms separate classes in separate .class byte code files.	DVM forms multiple class in .dex byte code file.	
		It is based on stack based virtual machine architecture.	It is based on Register based virtual machine architecture.	
		JVM runs on more memory.	DVM runs on less memory.	
		The executable format of JVM is JAR.	The executable format of DVM is APK.	
		JVM has different constant pools.	DVM has common constant pool.	
		It runs .class byte code directly.	The .class byte codes are optimized to .odex format before executing in DVM.	
	c)	**Explain android security model.**		**4 M**
	Ans	Android is a multi-process system, in which each application (and parts of the system) runs in its own process. Most security between applications and the system is enforced at the process level through standard Linux facilities, such as user and group IDs that are assigned to applications. Additional finer-grained security features are provided through a "permission" mechanism that enforces restrictions on the specific operations that a particular process can perform, and per-URI permissions for granting ad-hoc access to specific pieces of data. The Android security model is primarily based on a sandbox and permission mechanism. Each application is running in a specific Dalvik virtual machine with a unique user ID assigned to it, which means the application code runs in isolation from the code of all others applications. As a consequence, one application has not granted access to other applications' files. Android application has been signed with a certificate with a private key Know the owner of the application is unique. This allows the author of The application will be identified if needed. When an application is installed in The phone is assigned a user ID, thus avoiding it from affecting it Other applications by creating a sandbox for it. This user ID is permanent on which devices and applications with the same user ID are allowed to run in a single process. This is a way to ensure that a malicious application has Can not access / compromise the data of the genuine application.		correct explanation 4 M

		It is mandatory for an application to list all the resources it will Access during installation. 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). Permissions are divided into several protection levels. The protection level affects whether runtime permission requests are required.  Android introduced shared user ID & permission to allow application components talk to each other & enable application to access to critical system in Android devices.	
	d)	**Draw and explain activity life cycle.**	**4 M**

	Ans	 onCreate (): Called then the activity is created. Used to initialize the activity, for example create the user interface. onStart ():called when activity is becoming visible to the user. onResume (): Called if the activity get visible again and the user starts interacting with the activity again. Used to initialize fields, register listeners, bind to services, etc. onPause (): Called once another activity gets into the foreground. Always called before the activity is not visible anymore. Used to release resources or save application data. For	Diagram 2 M , Explanation 2 M

		example you unregister listeners, intent receivers, unbind from services or remove system service listeners. onStop (): Called once the activity is no longer visible. Time or CPU intensive shutdown operations, such as writing information to a database should be down in the onStop() method. This method is guaranteed to be called as of API 11. onDestroy (): called before the activity is destroyed. **1. Activity States:** The Android OS uses a priority queue to assist in managing activities running on the device. Based on the state a particular Android activity is in, it will be assigned a certain priority within the OS. This priority system helps Android identify activities that are no longer in use, allowing the OS to reclaim memory and resources. Fig. illustrates the states an activity can go through, during its lifetime: These states are often broken into three main teams as follows: **1. Active or Running:** Activities are thought of active or running if they're within the foreground, additionally referred to as the top of the activity stack. this can be thought of the highets priority activity within the Android Activity stack, and as such only be killed by the OS in extreme things, like if the activity tries to use more memory than is available on the device as this might cause the UI to become unresponsive. **2. Paused:** When the device goes to sleep, or an activity continues to be visible but partially hidden by a new, non-full-sized or clear activity, the activity is taken into account paused. Paused activities are still alive, that is, they maintain all state and member information, and stay attached to the window manager. This can be thought of to be the second highest priority activity within the android Activity stack and, as such, can solely be killed by the OS if killing this activity can satisfy the resource requirement needed to keep the Active/Running Activity stable and responsive. **3. Stopped:** Activities that are utterly obscured by another activity are thought of stopped or within the background. Stopped activities still try and retain their state and member info for as long as possible but stopped activities are thought of to be loweat priority of the three states and, as such, the OS can kill activities during this state initial to satisfy the resource needs of higher priority activities.	
3.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Describe various installation steps of android studio and its environment.**	**4 M**
	Ans	**Steps to install Android studio and SDK** **Pre-Installation Check List** 1. Before installing Android SDK, there is need to install Java Development Kit (JDK). Ensure that JDK is at or above 1.8. 2. Uninstall older version(s) of "Android Studio" and "Android SDK", if any.	Android studio installation steps 3 M SDK 1 M

		We need to install two packages: 1. Android Studio (IDE), which is an Integrated Development Environment (IDE) 2. Android SDK (Software Development Kit) for developing and running Android apps. **Steps to install Android studio:** Download Android Studio 1. Click Download Android Studio. The Terms and Conditions page with the Android Studio License Agreement opens. 2. Read the License Agreement. 3. At the bottom of the page, if you agree with the terms and conditions, select the I have read and agree with the above terms and conditions checkbox. 4. Click Download Android Studio to start the download. 5. When prompted, save the file to a location where you can easily locate it, such as the Downloads folder. 6. Wait for the download to complete. **Install Android Studio** a) Open the folder where you downloaded and saved the Android Studio installation file. b) Double-click the downloaded file. c) If you see a User Account Control dialog about allowing the installation to make changes to your computer, click Yes to confirm the installation. d) Click Next to start the installation. e) Accept the default installation settings for all steps. f) Click finish when installation is done. **Installing Android SDK** Within Android Studio, you can install the Android SDK as follows: 1. Click Tools > SDK Manager. 2. In the SDK Platforms tab, select Android Tiramisu Preview. 3. In the SDK Tools tab, select Android SDK Build. 4. Click OK to install the SDK.	
	b)	**Explain scrollview with its attributes and with suitable example.**	**4 M**
	Ans	**ScrollView** ScrollView is used to scroll the child elements of palette inside ScrollView. Android supports vertical scroll view as default scroll view. Vertical ScrollView scrolls elements vertically.Android uses *HorizontalScrollView* for horizontal ScrollView. **Attributes of ScrollView** • android:fillViewport - Defines whether the scrollview should stretch its content to fill the viewport. • android:scrollbars - scrollbars attribute is used to show the scrollbars in horizontal or vertical direction • android:layout_width – Define the width • android:layout_height – Define height • android:id – Define id	Explain: 1 M Attributes: 1 M Example : 2 M

Example :

```
<?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:orientation="vertical"
   android:paddingLeft="10dp"
   android:paddingRight="10dp">

<ScrollView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/scrollView">

<LinearLayout
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:orientation="vertical">
      <Button
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:text="Button 1"/>
      <Button
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:text="Button 2"/>
      <Button
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:text="Button 3"/>
      <Button
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:text="Button 4"/>
</LinearLayout>
</ScrollView>
</LinearLayout>
```

	c)	**Write significance of SQLite database in android.**	**4 M**
	Ans	• **SQLite** is an **open-source relational database** i.e. used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database. • It is embedded in android bydefault. So, there is no need to perform any database setup or administration task.	Any 4 points : 4 M

		• **SQLite** is one way of storing app data. It is very lightweight database that comes with Android OS. • By default, Android comes with built-in SQLite Database support so we don't need to do any configurations. • Android stores our database in a private disk space that's associated with our application and the data is secure, because by default this area is not accessible to other applications. • The package android.database.sqlite contains all the required APIs to use an SQLite database in our android applications. • In android, by using **SQLiteOpenHelper** class we can easily create the required database and tables for our application. To use SQLiteOpenHelper, we need to create a subclass that overrides the onCreate() and onUpgrade() call-back methods. • **SQLiteDatabase** class contains methods to be performed on sqlite database such as create, update, delete, select etc. • We can insert data into the SQLite database by passing ContentValues to **insert()** method. • In android, we can read the data from the SQLite database using the **query()** method in android applications. We can update the data in the SQLite database using an **update()** method in android applications.	
	d)	**Explain importance or use of developer console.**	**4 M**
	Ans	Google Play Developer Console is the platform that Google provides for Google Play and Android developers to publish their apps. ● The Google Play Developer console allows app developers and marketers to better understand how their apps are performing in terms of growth, technical performance such as crashes or display issues, and financials. ● The console offers acquisition reports and detailed analysis which can help app devs find out how well an app is really performing. ● The platform is important as it provides developers with access to first party data (trustworthy information collected about an app's audience that comes straight from Google Play) that highlights the real performance of an app. ● It shows the number of impressions an app listing receives and the number of Installs an app receives from different sources over time.	Any 4 points : 4 M

4.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Describe linear layout and frame layout with example.**	**4 M**
	Ans	**1. Linear Layout** Linear Layout is a ViewGroup that is responsible for holding views in it. It is a layout that arranges its children i.e the various views and layouts linearly in a single column(vertically) or a single row(horizontally). Whether all the children will be arranged horizontally or vertically depends upon the value of attribute android:orientation. By default the orientation is horizontal. <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="fill_parent" android:orientation="vertical" > <Button android:id="@+id/btnStartService" android:layout_width="270dp" android:layout_height="wrap_content" android:text="Button 1"/> <Button android:id="@+id/btnStopService" android:layout_width="270dp" android:layout_height="wrap_content" android:text="Button 2"/> </LinearLayout> **2. Frame Layout** FrameLayout is designed to block out an area on the screen to display a single item. Generally, FrameLayout should be used to hold a single child view, because it can be difficult	Linear layout : 2 M Frame layout : 2 M

to organize child views in a way that's scalable to different screen sizes without the children overlapping each other.

We can add multiple children to a FrameLayout and control their position within the FrameLayout by assigning gravity to each child, using the android:layout_gravity attribute.

Example:

```
<FrameLayout
   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"
   android:id="@+id/table"
  android:foregroundGravity="center"
   android:foreground="#000"
    tools:context=".MainActivity">
   <ImageView
      android:layout_width="200dp"
      android:layout_height="200dp"
      android:layout_marginBottom="10dp"
      android:src="@mipmap/ic_launcher"
      android:scaleType="centerCrop" />
<TextView
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_gravity=""
   android:text="CENTER"/>
</FrameLayout>
```

	b)	**Write a program to create first display screen of any search engine using auto complete text view.**	**4 M**
	Ans	<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:layout_gravity="center"> <AutoCompleteTextView android:id="@+id/txt" android:textSize="50dp" android:layout_centerHorizontal="true" android:hint="Enter Text to search" android:layout_width="wrap_content" android:layout_height="wrap_content" /> </RelativeLayout> package com.example.al_libaansapp; import android.os.Bundle; import android.view.View; import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.EditText; import android.widget.TextView; import android.widget.Toast; import androidx.appcompat.app.AppCompatActivity;	Xml file: 1 M Java file: 3 M

```
import android.view.View;
import android.widget.AutoCompleteTextView;
import android.widget.Button;


public class MainActivity extends AppCompatActivity
 {
   String[] fruits = {"apple", "mango", "banana", "kiwi", "pineapple", "guava", "grapes", "orange", "watermelon", "papaya"};
   AutoCompleteTextView txt;


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


      txt = findViewById(R.id.txt);


      ArrayAdapter adp = new ArrayAdapter(this, android.R.layout.simple_dropdown_item_1line, fruits);
      txt.setThreshold(1);
      txt.setAdapter(adp);
   }
}
```

	c)	**What is fragment? Explain with example.**	**4 M**
	Ans	• Android Fragment is the part of activity, it is also known as sub-activity. There can be more than one fragment in an activity. • Fragments represent multiple screen inside one activity.	Define Fragment : 2 M Example: 2 M

- We can create Fragments by extending Fragment class or by inserting a Fragment into our Activity layout by declaring the Fragment in the activity's layout file, as a <fragment> element.
- We can combine multiple Fragments in a single activity to build a multi-plane UI.
- We can only show a single Activity on the screen at one given point of time so we were not able to divide the screen and control different parts separately. With the help of Fragment's we can divide the screens in different parts and controls different parts separately.

Example:

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:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:gravity="center"
   tools:context=".MainActivity">

   <fragment
      android:id="@+id/Frag1"
      android:layout_width="match_parent"
      android:layout_height="400dp"
      android:name="com.example.al_libaansapp"/>
   <fragment
      android:id="@+id/Frag2"
      android:layout_width="match_parent"
      android:layout_height="400dp"
      android:name="com.example.al_libaansapp"/>
</LinearLayout>
```

- Fragment1.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout 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:background="@color/black"
   tools:context=".Fragment1">

   <TextView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:gravity="center"
      android:text="Fragment 1"
      android:textSize="40dp"
      android:textColor="@color/white"
      android:textStyle="bold" />

</FrameLayout>
```

- Fragment2.xml:

```
<FrameLayout

   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:background="@color/purple"
   tools:context=".Fragment2">

      <TextView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:gravity="center"
      android:text="Fragment 2"
      android:textSize="40dp"
      android:textStyle="bold"
      android:textColor="@color/black"/>

</FrameLayout>
```

	d)	**Describe types of permissions used while developing android application.**	**4 M**

	Ans	**Types of permissions** **1. Install-time permissions** • Install-time permissions give your app limited access to restricted data, and they allow your app to perform restricted actions that minimally affect the system or other apps. • When you declare install-time permissions in your app, the system automatically grants your app the permissions when the user installs your app. Android includes several sub-types of install-time permissions, Normal permissions and Signature permissions. **a) Normal permissions** • These permissions allow access to data and actions that extend beyond your app's sandbox. • However, the data and actions present very little risk to the user's privacy, and the operation of other apps. **b) Signature permissions** • If the app declares a signature permission that another app has defined, and if the two apps are signed by the same certificate, then the system grants the permission to the first app at install time. Otherwise, that first app cannot be granted the permission. **2. Runtime permissions** • Runtime permissions, also known as dangerous permissions, give your app additional access to restricted data, and they allow your app to perform restricted actions that more substantially affect the system and other apps. • Many runtime permissions access *private user data*, a special type of restricted data that includes potentially sensitive information. Examples of private user data include location and contact information. • The system assigns the "dangerous" protection level to runtime permissions. **3. Special permissions** • Special permissions correspond to particular app operations.	Each type : 1 M

		• Only the platform and OEMs can define special permissions. Additionally, the platform and OEMs usually define special permissions when they want to protect access to particularly powerful actions, such as drawing over other apps.	
	e)	**Develop a program to send an SMS.**	**4 M**
	Ans	AndroidManifest.xml <uses-permission android:name="android.permission.SEND_SMS"/> activity_main.xml <?xml version="1.0" encoding="utf-8"?> <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto" xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent" android:layout_height="match_parent" tools:context=".MainActivity"> <TextView android:id="@+id/textView" android:layout_width="81dp" android:layout_height="41dp" android:layout_marginEnd="268dp" android:layout_marginBottom="576dp" android:text="To :" app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintEnd_toEndOf="parent"/> <TextView android:id="@+id/textView2" android:layout_width="70dp"	Xml file: 2 M Java File : 2 M

```
android:layout_height="43dp"
android:layout_marginEnd="276dp"
android:layout_marginBottom="512dp"
android:text="Sms Text"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent" />
<EditText
android:id="@+id/etPhno"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginEnd="40dp"
android:layout_marginBottom="572dp"
android:ems="10"
android:inputType="textPersonName"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent" />
<EditText
android:id="@+id/etmsg"
android:layout_width="193dp"
android:layout_height="51dp"
android:layout_marginEnd="56dp"
android:layout_marginBottom="504dp"
android:inputType="textPersonName"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent" />
<Button
```

android:id="@+id/btnSms"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_marginEnd="156dp"

android:layout_marginBottom="400dp"

android:text="SEND SMS"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

MainActivity.java

public class MainActivity extends AppCompatActivity

{

EditText et1,et2;

Button b1;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);

et1=findViewById(R.id.etPhno);

et2=findViewById(R.id.etmsg);

b1=findViewById(R.id.btnSms);
if(ContextCompat.checkSelfPermission(MainActivity.this,Manifest.permission.SEND_SMS)!
=

PackageManager.PERMISSION_GRANTED)

{

ActivityCompat.requestPermissions(MainActivity.this,new

		String[]{Manifest.permission.SEND_SMS},100);
		}
		b1.setOnClickListener(new View.OnClickListener() {
		@Override
		public void onClick(View v) {
		try {
		String phno= et1.getText().toString();
		String msg=et2.getText().toString();
		SmsManager smsManager= SmsManager.getDefault();
		smsManager.sendTextMessage(phno,null,msg,null,null);
		Toast.makeText(MainActivity.this,"Sms sent successfully",
		Toast.LENGTH_LONG).show();
		}
		catch(Exception e)
		{
		Toast.makeText(MainActivity.this,"Sms failed to send... try again",
		Toast.LENGTH_LONG).show();
		}
		}
		});
		}
		}	

5.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Develop android application to enter one number and display factorial of a number once click on button.**	**6 M**
	Ans	**(Note: Consider the appropriate XML file. All attributes are not required. In java file all imports are not expected. Different relevant logic/code can be considered.)** **activity_main.xml**	xml file- 2 M java code- 4 M

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

   <EditText
      android:layout_width="wrap_content"
      android:layout_height=" wrap_content "
      android:hint="Enter a number"
      android:id="@+id/number"
       android:inputType="number"
       android:textSize="30dp"
       />

   <Button
       android:layout_width=" wrap_content "
       android:layout_height="wrap_content"
       android:id="@+id/btn1"
       android:layout_below="@id/number"
       android:textSize="20dp"
       android:text="Calculate Factorial"
/>

   <TextView
       android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/tv"
        android:layout_below="@id/btn1"
        android:textSize="20dp"/>
</RelativeLayout>
```

MainActivity.java

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

public class MainActivity extends AppCompatActivity {
    EditText number;
    Button btn1;
    TextView tv;
    int num;
    int factor;
String s;
     @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        number=(EditText)findViewById(R.id.number);
        btn1=(Button)findViewById(R.id.btn1);
```

```
        tv=(TextView)findViewById(R.id.tv);


        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String strnum;
                strnum = number.getText().toString();
                num = Integer.parseInt(strnum);
                factorial(num);


            }
        });
    }
    private void factorial(int num) {
        int i = 1;
        factor = i;
        while (i <= num) {
            factor = factor * i;
            i++;
        }
     s = "Factorial of Number is :   "+ factor;
   tv.setText(s);
    }
}
```

	b)	**Write a program to capture an image using camera and display it.**	**6 M**
	Ans	**(Note: Consider the appropriate XML file. All attributes are not required. In java file all imports are not expected. Different relevant logic/code can be considered.)**	XML file- 2 M Java code- 4 M

activity_main.xml

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

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

android:layout_width="match_parent"

android:layout_height="match_parent"

android:padding="40dp"

android:orientation="horizontal"

tools:context=".MainActivity">

<TextView

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:text="CAMERA"

android:id="@+id/text"

android:textSize="20dp"

android:gravity="center"/>

<ImageView

android:id="@+id/image"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_below="@+id/text"

android:layout_marginTop="81dp"

android:src="@drawable/rose"/>

<Button

android:id="@+id/photo"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_below="@+id/image"
```

android:layout_centerHorizontal="true"

android:layout_marginTop="30dp"

android:text="TAKE PHOTO" />

</RelativeLayout>

MainActivity.java

package com.example.ifcdiv;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

import android.graphics.Bitmap; import android.os.Bundle;

import android.provider.MediaStore;

import android.view.View;

import android.widget.Button;

import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

Button b1;

ImageView imageView;

int CAMERA_REQUEST=1;

@Override protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState); setContentView(R.layout.activity_main);
b1=findViewById(R.id.photo); imageView=findViewById(R.id.image);
b1.setOnClickListener(new View.OnClickListener() {

@Override public void onClick(View v) {

Intent i=new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(i,CAMERA_REQUEST); } }); }

@Override protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) { super.onActivityResult(requestCode, resultCode, data);

if (requestCode==CAMERA_REQUEST) { Bitmap image= (Bitmap)
data.getExtras().get("data");

imageView.setImageBitmap(image); } } }

	c)	**Write a program to show users current location.**	**6 M**
	Ans	***(Note: Consider the appropriate XML file. All attributes are not required. In java file all imports are not expected.)***	XML file: 1 M Java Code: 5 M

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<fragment
android:layout_width="match_parent"
android:layout_height="match_parent"
android:id="@+id/google_map"
android:name="com.google.android.gms.maps.SupportMapFragment" />
</RelativeLayout>
```

MainActivity.Java

```
package com.example.location;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.FragmentActivity;
import android.Manifest;
import android.content.pm.PackageManager;
import android.location.Location;
import android.os.Bundle;
import android.widget.Toast;
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
public class MainActivity extends FragmentActivity implements
OnMapReadyCallback
{
Location currentlocation;
FusedLocationProviderClient fusedLocationProviderClient;
```

```
private static final int REQUEST_CODE = 101;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(this);
fetchLastLocation();
}
private void fetchLastLocation() {
if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission
(this, Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
ActivityCompat.requestPermissions(this,new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},REQUEST_CODE);
return;
}
Task<Location> task = fusedLocationProviderClient.getLastLocation();
task.addOnSuccessListener(new OnSuccessListener<Location>() {
@Override
public void onSuccess(Location location) {
if(location!=null)
{
currentlocation=location;
Toast.makeText(getApplicationContext(),currentlocation.getLatitude()+""+
currentlocation.getLongitude(), Toast.LENGTH_SHORT).show();
SupportMapFragment supportMapFragment =
(SupportMapFragment)getSupportFragmentManager().findFragmentById(R.id.google_ma
p);
supportMapFragment.getMapAsync(MainActivity.this);
}
}
});
}
@Override
public void onMapReady(@NonNull GoogleMap googleMap) {
LatLng latLng=new LatLng(currentlocation.getLatitude(),currentlocation.getLongitude());
MarkerOptions markerOptions=new MarkerOptions().position(latLng)
.title("I am Here");
googleMap.animateCamera(CameraUpdateFactory.newLatLng(latLng));
googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng,5));
googleMap.addMarker(markerOptions);
}
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
super.onRequestPermissionsResult(requestCode, permissions, grantResults);
switch (requestCode) {
```

		case REQUEST_CODE:
		if (grantResults.length > 0 && grantResults[0] ==
		PackageManager.PERMISSION_GRANTED) {
		fetchLastLocation();
		}
		break;
		}
		}
		}	

6.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Write a program to display the list of sensors supported by device.**	**6 M**
	Ans	**(Note: Consider the appropriate XML file. All attributes are not required. In java file all imports are not expected. Different relevant logic/code can be considered.)** **activity_main.xml** <?xml version="1.0" encoding="UTF-8"?> <LinearLayout android:paddingRight="10dp" android:paddingLeft="10dp" android:layout_height="match_parent" android:layout_width="match_parent" android:orientation="vertical" xmlns:android="http://schemas.android.com/apk/res/android"> <TextView android:layout_height="wrap_content" android:layout_width="wrap_content" android:visibility="gone" android:layout_gravity="center" android:textStyle="bold" android:textSize="20dp" android:text="Sensors" android:layout_marginTop="80dp" android:id="@+id/sensorslist"/> </LinearLayout> **MainActivtiy.java** Package com.example.sensordisplay; import android.support.v7.app.AppCompatActivity; import android.os.Bundle; import android.content.Context; import android.hardware.Sensor; import android.hardware.SensorManager; import android.view.View;	Xml file- 2 M Java code- 4 M

```
import android.widget.TextView;
import java.util.List;
public class MainActivity extends AppCompatActivity {
private SensorManager mgr;
private TextView txtList;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
mgr = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
txtList = (TextView)findViewById(R.id.sensorslist);
List<Sensor> sensorList = mgr.getSensorList(Sensor.TYPE_ALL);
StringBuilder strBuilder = new StringBuilder();
for(Sensor s: sensorList){
strBuilder.append(s.getName()+"\n");
}
txtList.setVisibility(View.VISIBLE);
txtList.setText(strBuilder);
}
}
```

b) Write a program to send e-mail. — **6 M**

Ans **(Note: Consider the appropriate XML file. All attributes are not required. Different relevant logic/code can be considered.)** — Xml file- 2 M; Java code- 4 M

activity_main.xml

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

android:id="@+id/editText1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_alignParentRight="true"
android:layout_marginTop="18dp"
android:layout_marginRight="22dp" />
<EditText
android:id="@+id/editText2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/editText1"
android:layout_alignLeft="@+id/editText1"
android:layout_marginTop="20dp" />
```

```
<EditText
android:id="@+id/editText3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/editText2"
android:layout_alignLeft="@+id/editText2"
android:layout_marginTop="30dp" />
<TextView
android:id="@+id/textView1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignBaseline="@+id/editText1"
android:layout_alignBottom="@+id/editText1"
android:layout_alignParentLeft="true"
android:text="Send To:"
android:textColor="#0F9D58" />
<TextView
android:id="@+id/textView2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignBaseline="@+id/editText2"
android:layout_alignBottom="@+id/editText2"
android:layout_alignParentLeft="true"
android:text="Email Subject:"
android:textColor="#0F9D58" />
<TextView
android:id="@+id/textView3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignBaseline="@+id/editText3"
android:layout_alignBottom="@+id/editText3"
android:text="Email Body:"
android:textColor="#0F9D58" />
<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/editText3"
android:layout_alignLeft="@+id/editText3"

android:layout_marginLeft="76dp"
android:layout_marginTop="20dp"
android:text="Send email!!" />
</RelativeLayout>
```

MainActivity.java

```
package com.example.email;
import android.content.Intent;
```

```
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
// define objects for edit text and button
Button button;
EditText sendto, subject, body;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
// Getting instance of edittext and button
sendto = findViewById(R.id.editText1);
subject = findViewById(R.id.editText2);
body = findViewById(R.id.editText3);
button = findViewById(R.id.button);
// attach setOnClickListener to button with Intent object define in it
button.setOnClickListener(view -> {
String emailsend = sendto.getText().toString();
String emailsubject = subject.getText().toString();
String emailbody = body.getText().toString();
// define Intent object with action attribute as ACTION_SEND
Intent intent = new Intent(Intent.ACTION_SEND);
// add three fields to intent using putExtra function
intent.putExtra(Intent.EXTRA_EMAIL, new String[]{emailsend});
intent.putExtra(Intent.EXTRA_SUBJECT, emailsubject);
intent.putExtra(Intent.EXTRA_TEXT, emailbody);
// set type of intent
intent.setType("message/rfc822");
// startActivity with intent with chooser as Email client using createChooser function
startActivity(Intent.createChooser(intent, "Choose an Email client :"));
});
}
}
```

	c)	**Write a program to show five checkboxes and total selected checkboxes using linear layout.**	**6 M**
	Ans	**(Note: Consider the appropriate XML file. All attributes are not required. Different relevant logic/code can be considered.)**	XML file- 2 M Java code- 4 M

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:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<TextView
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Hobbies" android:id="@+id/t1"
/>
<CheckBox
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/c1"
android:text="Swimmning"
/>
<CheckBox
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/c2"
android:text="Running "
/>
<CheckBox
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/c3"
android:text="Cycling "
/>
<CheckBox android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/c4"
android:text="Reading "
/>
<CheckBox
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/c5"
android:text="Football "
/>
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Submit"
android:id="@+id/b1"
/>
</LinearLayout>
```

MainActivity.java

```
package com.example.checkbox;
import androidx.appcompat.app.AppCompatActivity;
import java.lang.StringBuffer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```
import android.widget.CheckBox;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
Button b1;
CheckBox c1,c2,c3,c4,c5;
String s;
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
s ="Your Hobbies are:- ";
b1=(Button) findViewById(R.id.b1);
c1=(CheckBox) findViewById(R.id.c1);
c2=(CheckBox) findViewById(R.id.c2);
c3=(CheckBox) findViewById(R.id.c3);
c4=(CheckBox) findViewById(R.id.c4);
c5=(CheckBox) findViewById(R.id.c5);
b1.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) { if(c1.isChecked())
{
s = s + c1.getText()+" ";
Toast.makeText(getApplicationContext(),s,Toast.LENGTH_SHORT).show();
}
if(c2.isChecked())
{
s = s + c2.getText()+" ";
Toast.makeText(getApplicationContext(),s,Toast.LENGTH_SHORT).show();
}
if(c3.isChecked())
{
s = s + c3.getText()+" ";
Toast.makeText(getApplicationContext(),s,Toast.LENGTH_SHORT).show();
}
if(c4.isChecked())
{
s = s + c4.getText()+" ";
Toast.makeText(getApplicationContext(),s,Toast.LENGTH_SHORT).show();
}
if(c5.isChecked())
{
s = s + c5.getText()+" ";
Toast.makeText(getApplicationContext(),s,Toast.LENGTH_SHORT).show();
}
}
});
}
}
```