_____

SUMMER – 2022 EXAMINATION

Subject Name: Mobile Application Development   **Model Answer**   **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
		Note: As Android programs contain many of the generated code statements from IDE, while assessing such answers correct logical steps taken to obtain the required output can be considered.	
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 four, ½ M for one feature)
	b)	**Define Dalvik Virtual Machine (DVM).**	2 M

	Ans	Dalvik Virtual Machine is a register-based machine that compiles byte code to get dex code and that ensures that a device can run multiple instances efficiently.	(Correct definition 2 M)
	c)	**List any four folders from directory structure of Android project and elaborate in one line.**	**2 M**
	Ans	Folders from directory structure: **1)app**: The App folder contains three subfolders (manifests, java and res) that make up our application. They are divided so that it should be fairly easy to determine which resources go in which folder. **2)Manifest**: This is where we would put our manifest files. Most Android apps have single manifest file. But an app may have several manifest files due to application versioning, or for supporting specific hardware. **3)Java**: This is the folder in our project where we will be storing all of the source code files written in Java programming language. **4)res**: It contains folders that help us to separate and sort the resources of our application. Resources basically mean all the needed files except the source code. **5)drawable**: The drawable folder contains graphics that can be drawn to the screen. **6)layout**: The layout folder contains XML files used for your layouts. These file are used to set up the layout for your Activity and is used for basic alignment of your layouts, components, widgets, and similar resources that are used for the UI of your application. **7)mipmap** : The mipmap folder contains the launcher icon files for the app. A launcher icon is a graphic that represents your app to users. **8)values**: The values folder contains XML files that contain simple values, such as strings, integers, and colors. The values folder is used to keep track of the values we will be using in our application.	(List of names of any four folders : 1 M and elaboration in one line :1 M)
	d)	**List any four attributes of check box.**	**2 M**
	Ans	1)id 2)checked 3)gravity 4)text 5)text color 6)text size 7)text style 8)background 9)padding	(Any four attribute, ½ M for each)
	e)	**Draw diagram of activity life cycle.**	**2 M**

	Ans		(2 M for correct diagram)
	f)	**State syntax to display built in zoom control.**	**2 M**
	Ans	a) Built in Zoom control in Google map can be displayed with : UiSettings.setZoomControlsEnabled(true); OR b) In case, while display Google map by intent, default zoom level can be given in the form of data as geo:latitude,longitude?z=zoom where lat and lag are for location and zoom level is to set initial zoom level which	Correct Syntax : 2 M)

_____

|			
		ranges from 0 to 21. OR c) In any normal activity, ZoomControl can be displayed as component by following syntax : ZoomControl zoomControls = (ZoomControls) findViewById(R.id.simpleZoomControl); zoomControls.show()	
	g)	**Name two classes used to play audio and video in Android.**	**2 M**
	Ans	1) MediaPlayer 2) MediaController 3) AudioManager	(Any two class names 1 M each)
2.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Describe Android architecture with diagram.**	**4 M**

Ans

Diagram: 2M
Explanation: 2M
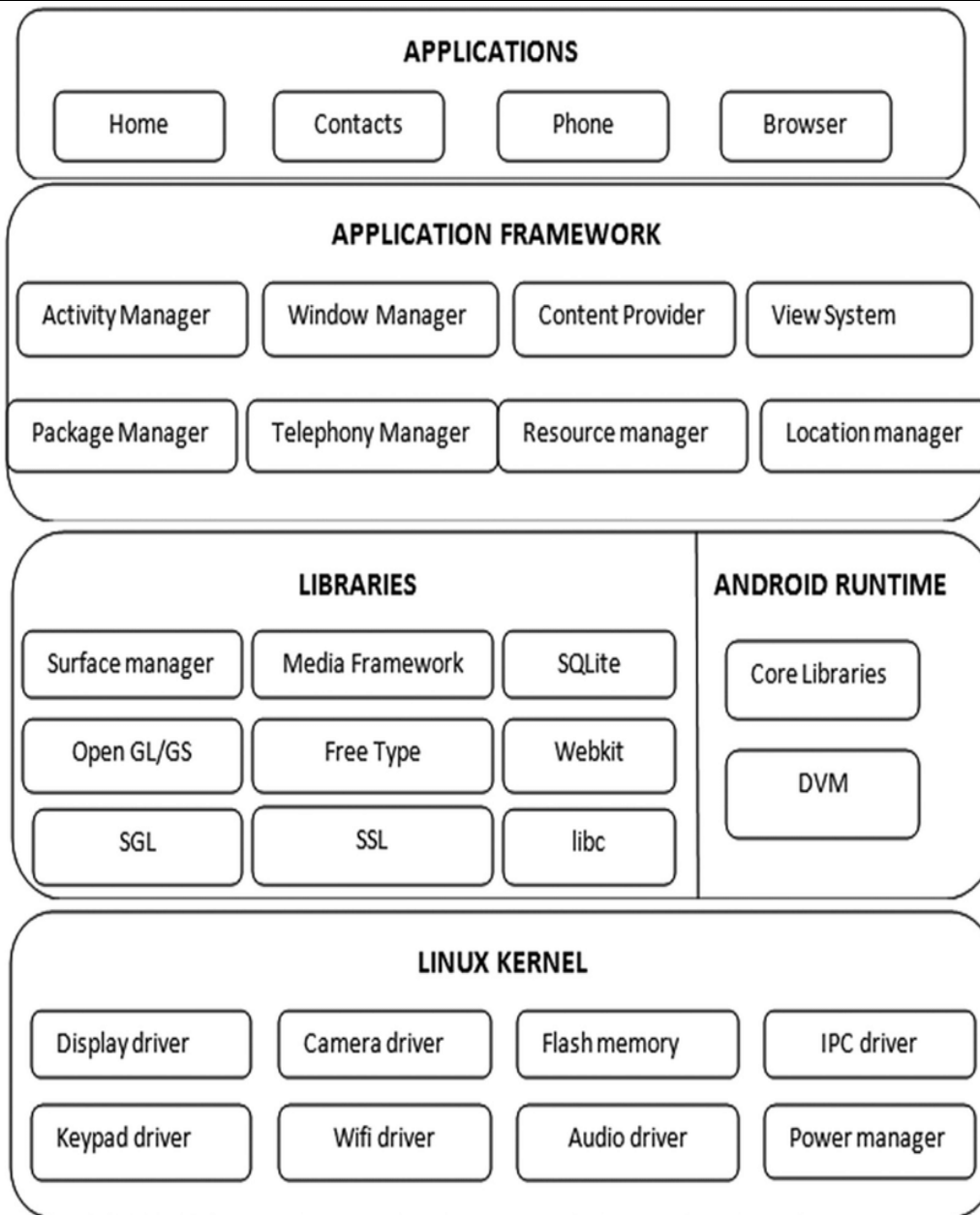
1. Applications:

• The top layer of android architecture is Applications. The native and third party applications like Contacts, Email, Music, Gallery, Clock, Games, etc. whatever we will build those will be installed on this layer only.

• The application layer runs within the Android run time using the classes and services made available from the application framework.

2. Application Framework:

• The Application Framework provides the classes used to create an Android application. It also provides a generic abstraction for hardware access and manages the user interface

and application resources.

• It basically provides the services through which we can create the particular class and make that class helpful for the Applications creation.

• The application framework includes services like telephony service, location services, notification. manager, NFC service, view system, etc. which we can use for application development as per our requirements.

3. Android Runtime:

• Android Runtime environment is an important part of Android rather than an internal part and it contains a components like core libraries and the Dalvik virtual machine.

• The Android run time is the engine that powers our applications along with the libraries and it forms the basis for the application framework.

(i) Dalvik Virtual Machine (DVM) is a register-based virtual machine like Java Virtual Machine (JVM).

It is specially designed and optimized for android to ensure that a device can run multiple

instances efficiently. It relies on the Linux kernel for threading and low-level memory

management.

(ii) The core libraries in android runtime will enable us to implement an android applications using standard JAVA programming language.

4. Platform Libraries:

• The Platform Libraries includes various C/C++ core libraries and Java based libraries such as SSL, libc, Graphics, SQLite, Webkit, Media, Surface Manger, OpenGL etc. to provide a support for android development.

• Following are the summary details of some core android libraries available for android development.

• Media library for playing and recording an audio and video formats

(i) The Surface manager library to provide a display management

(ii) SGL and OpenGL Graphics libraries for 2D and 3D graphics

(iii) SQLite is for database support and FreeType for font support

(iv) Web-Kit for web browser support and SSL for Internet security.

5. Linux Kernel:

		• Linux Kernel is a bottom layer and heart of the android architecture. It is heart of Android architecture that exists at the root of android architecture and contains all the low-level device drivers for the various hardware components of an Android device. • Linux Kernel is responsible fro device drivers, power management, memory management, device management and resource access. It manage all the drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are mainly required for the android device during the runtime. • The Linux Kernel will provides an abstraction layer between the device hardware and the remainder of the stack. It is responsible for memory management, power management, device management, resource access, etc.	
	b)	**Differentiate between DVM and JVM.**	**4 M**
	Ans		Any 4 points of differences : 1 M each

DVM (Dalvik Virtual Machine)	JVM (Java Virtual Machine)
It is Register based which is designed to run on low memory.	It is Stack based.
DVM uses its own byte code and runs the ".Dex" file. From Android 2.2 SDK Dalvik has got a Just in Time compiler	JVM uses java byte code and runs ".class" file having JIT (Just In Time).
DVM has been designed so that a device can run multiple instances of the VM efficiently. Applications are given their own instance.	A single instance of JVM is shared with multiple applications.
DVM supports the Android operating system only.	JVM supports multiple operating systems.
There is a constant pool for every application.	It has a constant pool for every class.
Here the executable is APK.	Here the executable is JAR.

	c)	**List and elaborate steps to deploy an Android application on Google play store.**	**4M**
	Ans	Steps to deploy and Android Application on Google Play Store: Step 1: Make a Developer Account Step 2: Plan to Sell? Link Your Merchant Account Step 3: Create an App Step 4: Prepare Store Listing	(List 1 M and Elaboration 3 M)

Step 5: Upload APK to an App Release
Step 6: Provide an Appropriate Content Rating
Step 7: Set Up Pricing & Distribution
Step 8: Rollout Release to Publish Your App

Step 1: Create a Developer Account

Before you can publish any app on Google Play, you need to create a Developer Account. You can easily sign up for one using your existing Google Account. You'll need to pay a one-time registration fee of $25 using your international credit or debit card. It can take up to 48 hours for your registration to be fully processed.

Step 2: Plan to Sell? Link Your Merchant Account

If you want to publish a paid app or plan to sell in-app purchases, you need to create a payments center profile, i.e. a merchant account. A merchant account will let you manage your app sales and monthly payouts, as well as analyze your sales reports right in your Play Console.

Step 3: Create an App

Now you have create an application by clicking on 'Create Application'. Here you have to select your app's default language from the drop-down menu and then type in a title for your app. The title of your app will show on Google Play after you've published.

Step 4: Prepare Store Listing

Before you can publish your app, you need to prepare its store listing. These are all the details that will show up to customers on your app's listing on Google Play. You not necessarily complete it at once , you can always save a draft and revisit it later when you're ready to publish.

The information required for your store listing is divided into several categories such as **Product Details** containing title, short and full description of the app, Your app's title and description should be written with a great user experience in mind. Use the right keywords, but don't overdo it. Make sure your app doesn't come across as spam-y or promotional, or it will risk getting suspended on the Play Store.

Graphic Assets where you can add screenshots, images, videos, promotional graphics, and icons that showcase your app's features and functionality.

Languages & Translations, Categorization where in category can be selected to which your app belong to. **Contact Details , Privacy Policy** for apps that request access to sensitive user data or permissions, you need to enter a comprehensive privacy policy that effectively discloses how your app collects, uses, and shares that data.

Step 5: Upload APK to an App Release

Finally upload your app, by uploading APK file. Before you upload APK, you need to create an app release. You need to select the type of release you want to upload your first app version to. You can choose between an internal test, a closed test, an open test, and a production release. The first three releases allow you to test out your app among a select

_____

group of users before you make it go live for everyone to access.

This is a safer option because you can analyze the test results and optimize or fix your app accordingly if you need to before rolling it out to all users.

Once you create a production release, your uploaded app version will become accessible to everyone in the countries you choose to distribute it in and click on 'Create release.'

Step 6: Provide an Appropriate Content Rating

If you don't assign a rating to your app, it will be listed as 'Unrated'. Apps that are 'Unrated' may get removed from Google Play.

To rate your app, you need to fill out a content rating questionnaire An appropriate content rating will also help you get to the right audience, which will eventually improve your engagement rates.

Step 7: Set Up Pricing & Distribution

Before you can fill out the details required in this step, you need to determine your app's monetization strategy. Once you know how your app is going to make money, you can go ahead and set up your app as free or paid.

You can always change your app from paid to free later, but you cannot change a free app to paid. For that, you'll need to create a new app and set its price.

Step 8: Rollout Release to Publish Your App

The final step involves reviewing and rolling out your release after making sure you've taken care of everything else.

Before you review and rollout your release, make sure the store listing, content rating, and pricing and distribution sections of your app each have a green check mark next to them.

Once you're sure about the correctness of the details, select your app and navigate to 'Release management' – 'App releases.' You can always opt for reviews by clicking on 'Review' to be taken to the 'Review and rollout release' screen. Here, you can see if there are any issues or warnings you might have missed out on.

Finally, select 'Confirm rollout.' This will also publish your app to all users in your target countries on Google Play.

	d)	**Describe with example, how to create a simple database in SQLite (Assume suitable data).** *(Note : Any other method such as creating subclass of SQLiteOpenHelper class and overriding and using required methods with relevant example can also be considered)*	**4M**
	Ans	**This procedure is by openOrCreateDatabase()** 1. The package imported into the application is android.database.sqlite.SQLiteDatabase. 2. Here the class used is SQLiteDatabase. 3. The method used to create the database or connect to the database is openOrCreateDatabse() method. **Program:** **activity_main.xml** *<?*xml version="1.0" encoding="utf-8"*?>* <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"	Description 1 M, 1 M for XML file , 2 M for Java File)

		xmlns:tools="http://schemas.android.com/tools"
		android:layout_width="match_parent"
		android:layout_height="match_parent"
		tools:context=".MainActivity">
		<Button
		    android:text="Create SQLite Database"
		    android:layout_width="fill_parent"
		    android:layout_height="wrap_content"
		    android:layout_centerHorizontal="true"
		    android:layout_marginTop="46dp"
		    android:id="@+id/button" />
		</RelativeLayout>
		
		MainActivity.java
		
		package in.edu.vpt.insertusingasync;
		import android.app.ProgressDialog;
		import android.content.Context;
		import android.database.sqlite.SQLiteDatabase;
		import android.os.Bundle;
		import android.support.v7.app.AppCompatActivity;
		import android.view.View;
		import android.widget.Button;;
		
		public class MainActivity extends AppCompatActivity {
		   SQLiteDatabase sqLiteDatabaseObj;
		   Button EnterData;
		
		   @Override
		   protected void onCreate(Bundle savedInstanceState) {
		     super.onCreate(savedInstanceState);
		     setContentView(R.layout.*activity_main*);
		
		     createData = (Button)findViewById(R.id.*button*);
		       createData.setOnClickListener(new View.OnClickListener() {
		     @Override
		     public void onClick(View view) {
		      sqLiteDatabaseObj = openOrCreateDatabase("AndroidJSonDataBase",
		Context.*MODE_PRIVATE*, null);
		    } });
		  }	

3.		**Attempt any __THREE__ of the following:**	**12 M**
	a)	**Write down the steps to install and configure Android studio.**	**4 M**
	Ans	**Step 1:** Go to https://developer.android.com/android-studio/download to get the Android Studio executable or zip file.	(4 M for appropriate

_____

		Step 2: ● Click on the **Download Android Studio** Button. ● Click on the "I have read and agree with the above terms and conditions" checkbox followed by the download button ● Click on the Save file button in the appeared prompt box and the file will start downloading. **Step 3:** After the downloading has finished, open the file from downloads and will prompt the following dialog box. Click on next. In the next prompt, it'll ask for a path for installation. Choose a path and hit next. **Step 4:** It will start the installation, and once it is completed, it will be like the image shown below. **Step 5:** Once "**Finish**" is clicked, it will ask whether the previous settings need to be imported [if the android studio had been installed earlier], or not. It is better to choose the 'Don't import Settings option'. Click the **OK** button. **Step 6:** This will start the Android Studio. Meanwhile, it will be finding the available SDK components. **Step 7:** After it has found the SDK components, it will redirect to the Welcome dialog box. Choose Standard and click on Next. Now choose the theme, whether the **Light** theme or the **Dark** one. The light one is called the **IntelliJ** theme whereas the dark theme is called **Darcula**. Choose as required. Click on the **Next** button. **Step 8:** Now it is time to download the SDK components. Click on Finish. Components begin to download let it complete. The Android Studio has been successfully configured. Now it's time to launch and build apps. Click on the Finish button to launch it. **Step 9:** Click on **Start a new Android Studio project** to build a new app.	steps)
	b)	**State syntax to create Text View and Image button with any two attributes of each.**	**4 M**

	Ans	**Text View:**
		Syntax :
		<TextView
		     android:id="@+id/textView1"
		     android:layout_width="<width value>"
		     android:layout_height="<height_value>"
		     android:text="<text to be displayed>"/>
		Attributes/Properties of TextView:
		● **id:** Supply an identifier name of this view, to later retrieve it with View.findViewByID() or Activity.findViewById()
		● **alpha:** alpha property of the view as a value between 0 (entirely transparent) and 1(Completely Opaque). [flag]
		● **auto link:** Controls whether links such as urls and email addresses are automatically found and converted to clickable links.[flag]
		● **gravity:** The gravity attribute is an optional attribute which is used to control the alignment of the text like left, right, center, top, bottom, center_vertical, center_horizontal etc
		● **text:** text attribute is used to set the text in a text view. We can set the text in xml as
			(Syntax of TextView and ImageButton : 1 M each Any two appropriate attributes of each : 1/2 M each)

well as in the java class.

- **textColor:** textColor attribute is used to set the text color of a text view. Color value is in the form of "#argb", "#rgb", "#rrggbb", or "#aarrggbb".
- **textSize:** textSize attribute is used to set the size of text of a text view. We can set the text size in sp(scale independent pixel) or dp(density pixel).
- **textStyle:** textStyle attribute is used to set the text style of a text view. The possible text styles are bold, italic and normal. If we need to use two or more styles for a text view then "|" operator is used for that.
- **background:** background attribute is used to set the background of a text view. We can set a color or a drawable in the background of a text view.
- **padding:** padding attribute is used to set the padding from left, right, top or bottom. In above example code of background we also set the 10dp padding from all the sides of text view.

ImageButton:
Syntax :
```
<ImageButton
    android:id="@+id/imageButton"
    android:layout_width="<width value>"
    android:layout_height="<height value>"
    app:srcCompat="<image source from drawable folder "/>
```
Attributes/Properties of ImageButton:

- **id:** id is an attribute used to uniquely identify a image button. Below is the example code in which we set the id of a image button.
- **src:** src is an attribute used to set a source file of image or you can say image in your image button to make your layout look attractive.
- **background:** background attribute is used to set the background of an image button. We can set a color or a drawable in the background of a Button.
- **padding:** padding attribute is used to set the padding from left, right, top or bottom of the ImageButton.

	c)	**Describe Android service life cycle along with diagram.**	**4 M**
	Ans	● A service is an application component which runs without direst 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. ● Service can either be started or bound we just need to call either startService() or bindService() from any of our android components. Based on how our service was started it will either be "started" or "bound"	(Explanation 2 M, Diagram 2M)
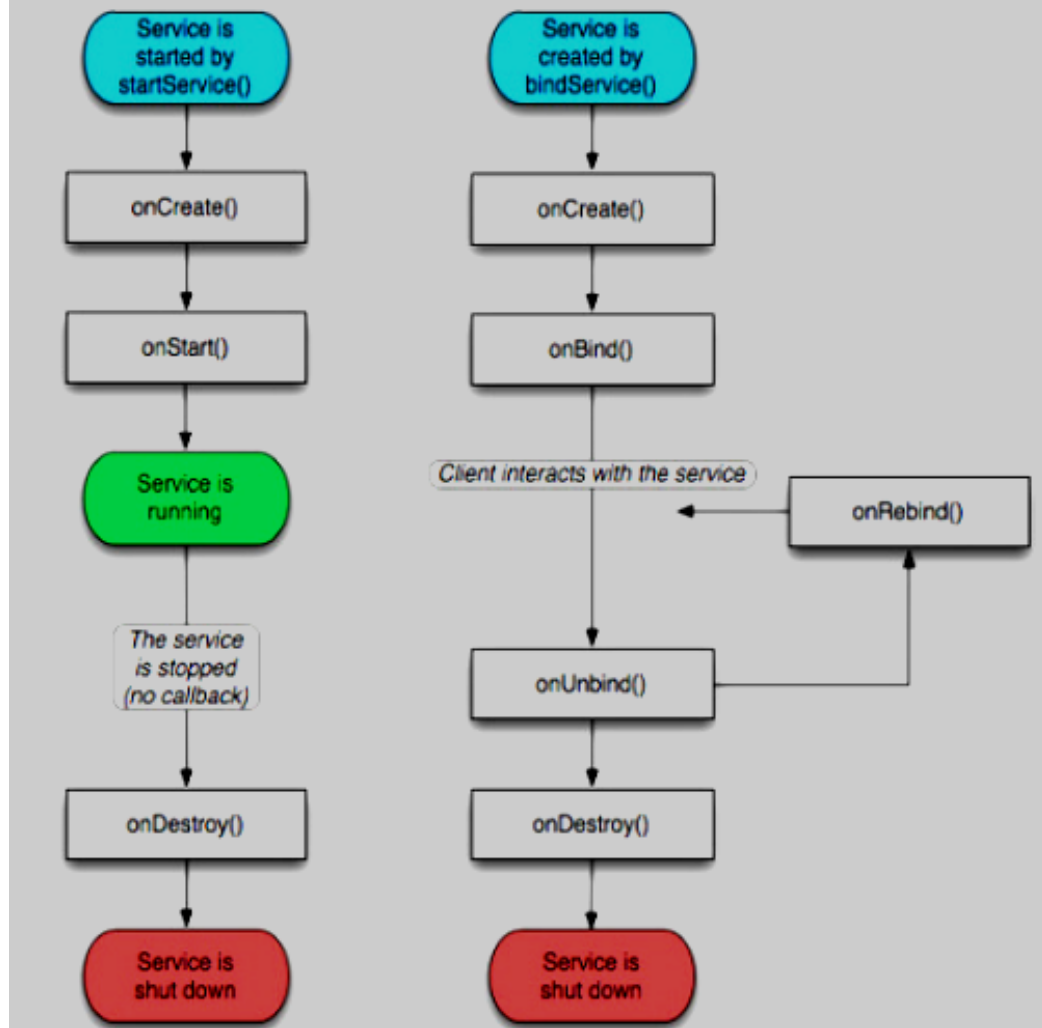
Service Lifecycle

1. **Started**
   a. A service is started when an application component, such as an activity, starts it by calling startService().
   b. Now the service can run in the background indefinitely, even if the component that started it is destroyed.

2. **Bound**
   a. A service is bound when an application component binds to it by calling bindService().
   b. A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with InterProcess Communication (IPC).
   c. Like any other components service also has callback methods. These will be invoked while the service is running to inform the application of its state. Implementing these in our custom service would help you in performing the right operation in the right state. •
   d. There is always only a single instance of service running in the app. If you are calling startService() for a single service multiple times in our

application it just invokes the onStartCommand() on that service. Neither is the service restarted multiple times nor are its multiple instances created

1. **onCreate():**
   This is the first callback which will be invoked when any component starts the service. If the same service is called again while it is still running this method wont be invoked. Ideally one time setup and intializing should be done in this callback.

2. **onStartCommand() /startSetvice()**
   This callback is invoked when service is started by any component by calling **startService().** It basically indicates that the service has started and can now run indefinetly.

3. **onBind()**
   To provide binding for a service, you must implement the onBind() callback method. This method returns an IBinder object that defines the programming interface that clients can use to interact with the service.

4. **onUnbind()**
   This is invoked when all the clients are disconnected from the service.

5. **onRebind()**
   This is invoked when new clients are connected to the service. It is called after onRebind

6. **onDestroy()**
   This is a final clean up call from the system. This is invoked just before the service is being destroyed.

	d)	**State and elaborate the syntax of required class and methods for Geocoding.**	**4 M**
	Ans	**Geocoder:** A class for handling geocoding and reverse geocoding. Geocoding is the process of transforming a street address or other description of a location into a (latitude, longitude) coordinate. Reverse geocoding is the process of transforming a (latitude, longitude) coordinate into a (partial) address. The amount of detail in a reverse geocoded location description may vary, for example one might contain the full street address of the closest building, while another might contain only a city name and postal code. The Geocoder class requires a backend service that is not included in the core android framework. The Geocoder query methods will return an empty list if there no backend service in the platform. Use the isPresent() method to determine whether a Geocoder implementation exists.	(2 M for explanation and syntax of class, 2 M for explanation and syntax of any two methods)

Syntax
Geocoder (Context context)

Constructs a Geocoder localized for the default locale.

Geocoder(Context context, Locale locale)

Constructs a Geocoder localized for the given locale.

Methods with Syntax
a. getFromLocation

Syntax

public List<Address> getFromLocation (double latitude, double longitude, int maxResults)

public void getFromLocation (double latitude, double longitude, int maxResults, Geocoder.GeocodeListener listener)

This method returns an array of Addresses that attempt to describe the area immediately surrounding the given latitude and longitude. The returned addresses should be localized for the locale provided to this class's constructor.

b. getFromLocationName
Syntax :
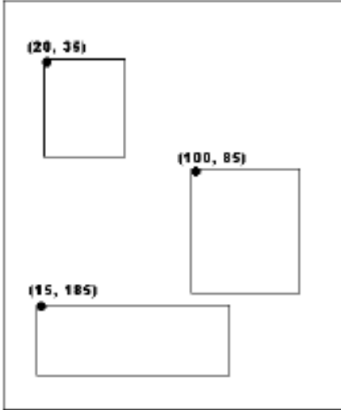
- public List<Address> getFromLocationName (String locationName, int maxResults, double lowerLeftLatitude, double lowerLeftLongitude, double upperRightLatitude, double upperRightLongitude)
- public void getFromLocationName (String locationName, int maxResults, double lowerLeftLatitude, double lowerLeftLongitude, double upperRightLatitude, double upperRightLongitude, Geocoder.GeocodeListener listener)
- public void getFromLocationName (String locationName, int maxResults, Geocoder.GeocodeListener listener)
- public List<Address> getFromLocationName (String locationName, int maxResults)

Returns an array of Addresses that attempt to describe the named location, which may be a place name such as "Dalvik, Iceland", an address such as "1600 Amphitheatre Parkway, Mountain View, CA", an airport code such as "SFO", and so forth. The returned addresses should be localized for the locale provided to this class's constructor.

c. isPresent

Syntax

		public static boolean isPresent () Returns true if there is a geocoder implementation present that may return results. If true, there is still no guarantee that any individual geocoding attempt will succeed.	
4.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Explain with example, code to create GUI using absolute layout (Assume suitable data).**	**4 M**
	Ans	● AbsoluteLayout is based on the simple idea of placing each control at an absolute position. We specify the exact x and y coordinates on the screen for each control. This is not recommended for most UI development (in fact AbsoluteLayout is currently deprecated) since absolutely positioning every element on the screen makes an inflexible UI that is much more difficult to maintain. **Absolute Layout** ● An Absolute Layout lets you specify exact locations (x/y coordinates) of its children. Absolute layouts are less flexible and harder to maintain than other types of layouts without absolute positioning. AbsoluteLayout is based on the simple idea of placing each control at an absolute position. ● We specify for the exact x and y corodinates on the screen for every control. So this recommended for most UI development (in fact Absolute Layout is currentaly deprecated)since absolute positioning of every element on the screen makes an inflexible UI that is much more difficult to maintain. ● Consider what happens if a control needs to be added to the user interface UI, we would have to change the position of every single element that is shifted by the new control. This allows child views to be positioned at specified X and Y coordinates within the containing layout view. **Example** *(Note :Any other relevant example using absoluteLayout can be considered, No java code is expected)*	(Explanation 2 M, Example 2 M)

_____

		activity_main.xml
		
		<AbsoluteLayoutxmlns:android="http://schemas.android.com/apk/res/android"
		android:layout_width="fill_parent"
		android:layout_height="fill_parent">
		    <TextView
		    android:layout_x="110px"
		    android:layout_y="110px"
		    android:text="User Name"
		    android:layout_width="wrap_content"
		    android:layout_height="wrap_content" />
		    <EditText
		    android:layout_x="250px"
		    android:layout_y="80px"
		    android:width="100px"
		    android:layout_width="200dp"
		    android:layout_height="wrap_content" />
		    <TextView
		    android:layout_x="110px"
		    android:layout_y="200px"
		    android:text="Password"
		    android:layout_width="wrap_content"
		    android:layout_height="wrap_content" />
		    <EditText
		    android:layout_x="250px"
		    android:layout_y="150px"
		    android:width="100px"
		    android:layout_width="200dp"
		    android:layout_height="wrap_content" />
		    <Button
		    android:layout_width="wrap_content"
		    android:layout_height="wrap_content"
		    android:text="Log In"
		    android:layout_x="300px"
		    android:layout_y="300px"/>
		</AbsoluteLayout>	

	b)	**Write a program to demonstrate Date and Time picker.** *(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.)*	**4 M**

	Ans	**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"
			(2M for correct use of Date picker
			
			2M for

_____

		android:layout_height="match_parent"
		tools:context=".MainActivity">
		
		<TextView
		  android:id="@+id/tvDate"
		  android:layout_width="149dp"
		  android:layout_height="46dp"
		  android:layout_marginEnd="224dp"
		  android:layout_marginBottom="312dp"
		  android:textSize="20dp"
		  android:textStyle="bold"
		  app:layout_constraintBottom_toBottomOf="parent"
		  app:layout_constraintEnd_toEndOf="parent" **/>**
		
		<Button
		  android:id="@+id/btnDate"
		  android:layout_width="wrap_content"
		  android:layout_height="wrap_content"
		  android:layout_marginEnd="96dp"
		  android:layout_marginBottom="312dp"
		  android:text="Set Date"
		  app:layout_constraintBottom_toBottomOf="parent"
		  app:layout_constraintEnd_toEndOf="parent"
		  tools:ignore="DuplicateClickableBoundsCheck" **/>**
		
		<DatePicker
		  android:id="@+id/dtpcker"
		  android:layout_width="314dp"
		  android:layout_height="293dp"
		  android:layout_marginBottom="368dp"
		  android:datePickerMode="spinner"
		  app:layout_constraintBottom_toBottomOf="parent"
		  app:layout_constraintEnd_toEndOf="parent"
		  app:layout_constraintHorizontal_bias="0.36"
		  app:layout_constraintStart_toStartOf="parent" **/>**
		
		<TimePicker
		  android:id="@+id/timepcker"
		  android:layout_width="184dp"
		  android:layout_height="195dp"
		  android:layout_marginEnd="132dp"
		  android:layout_marginBottom="108dp"
		  android:timePickerMode="spinner"
		  app:layout_constraintBottom_toBottomOf="parent"
		  app:layout_constraintEnd_toEndOf="parent" **/>**
		
		<TextView
		  android:id="@+id/tvTime"
		  android:layout_width="130dp"
			correct use
			of
			TimePicker)

```
     android:layout_height="56dp"
     android:layout_marginEnd="232dp"
     android:layout_marginBottom="40dp"
     android:textSize="20dp"
     android:textStyle="bold"
     app:layout_constraintBottom_toBottomOf="parent"
     app:layout_constraintEnd_toEndOf="parent" />

  <Button
     android:id="@+id/btnTime"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_marginEnd="104dp"
     android:layout_marginBottom="48dp"
     android:text="Set Time"
     app:layout_constraintBottom_toBottomOf="parent"
     app:layout_constraintEnd_toEndOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java
```java
package com.example.datepickereg;
import androidx.appcompat.app.AppCompatActivity;
import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;
import android.widget.TimePicker;
import java.util.Calendar;

public class MainActivity extends AppCompatActivity {
TextView tvDate,tvTime;
DatePicker dtpcker;
TimePicker timepcker;
Button b1,b2;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    tvDate=findViewById(R.id.tvDate);
    tvTime=findViewById(R.id.tvTime);
    b1=findViewById(R.id.btnDate);
    b2=findViewById(R.id.btnTime);
    dtpcker=findViewById(R.id.dtpcker);
    timepcker=findViewById(R.id.timepcker);
    b1.setOnClickListener(new View.OnClickListener() {
      @Override
```

_____

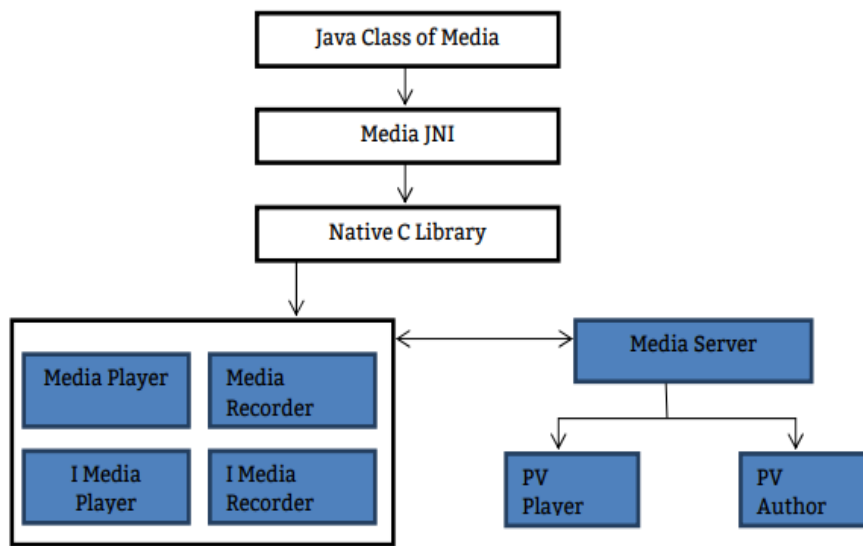|			

		public void onClick(View view) {
		    tvDate.setText("Date : "+dtpcker.getDayOfMonth()+"-
		"+dtpcker.getMonth()+"-"+dtpcker.getYear());
		    }
		  });
		    b2.setOnClickListener(new View.OnClickListener() {
		      @Override
		      public void onClick(View view) {
		
		tvTime.setText(timepcker.getCurrentHour()+":"+timepcker.getCurrentMinute());
		      }
		    });
		    }
		}	

c)		**Describe multimedia framework of Android with diagram.**	**4 M**
Ans		● The android multimedia system includes multimedia applications, multimedia frameworks, OpenCore engine and hardware abstract for audio/video input/output devices. And the goal of the android multimedia framework is to provide a reliable interface for java services. The multimedia framework consists of several core dynamic libraries such as libmediajni, libmedia, libmediaplayservice and so on. ● Java classes call the Native C library Libmedia through Java JNI(Java Native Interface). Libmedia library communications with Media Server guard process through Android's Binder IPc (inter process communication) mechanism. ● Media Server process creates the corresponding multimedia service according to the Java multimedia applications. ● The whole communication between Libmedia and Media Server forms a Client/Server model **Android Multimedia Framework Architecture**	(Explanation 2M, Diagram 2 M)

		● Typical video/audio data stream works in Android as follows. Particularly, Java applications first set the URI of the media (from file or network) to PVPlayer through Java framework, JNI and Native C. In this process, there are no data stream flows. ● Then PVPlayer processes the media data stream with the steps: demux the media data to separate video/audio data stream, decode video/audio data, sync video.audio time, send the decoded data out. ● The below is the description of media codec/format, container and network protocol supported by the Android platform. 1. **Container:** The audio file format is a file for storing digital audio data on a system. This data can be manipulated to reduce the size or change the quality of the audio. It is a kind of container to store audio information. 2. **Audio Format:** Any format or codec can be used including the ones provided by Android or those which are specific devices. However it is recommended to use the specified file formats as per devices. 3. **Network Protocol:** Protocols such as RTSP, HTTP,HTTPS are supported in audio and video playback.	
	d)	**Discuss developer console with at least four features**	**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 relevant features 1M each)
	e)	**Write a program to demonstrate declaring and using permissions with any relevant example.**	**4 M**
	Ans	**Permission declaring :** The permissions are declared in AndroidManifest.xml file under Manifest folder. Permission can be set by <uses-permission> tag in AndroidManifest.xml. **Example:** Following example is to send SMS. *(Note: Any other relevant example which uses permissions can be considered)* **AndroidManifest.xml**	4 M for any correct program demonstrating declaration and use of permissions

_____

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

activity_main.xml

```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"
    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"
        tools:ignore="SpeakableTextPresentCheck" />

    <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

```java
package com.example.testreceivesms;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

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)	**Write a program to convert temperature from celcius to farenhite and vice versa using Toggle button. (Design UI as per your choice. Write XML and java file)** *(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.)*	**6 M**
	Ans	**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">

  <EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/edittext"
    android:hint="Enter the temp"/>

  <ToggleButton
    android:id="@+id/togglebutton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/edittext"
    android:layout_marginTop="35dp"
    android:textOff="F to C"
    android:textOn="C to F" />``` | XML file: 2M

 Java Code: 4M |

```xml
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/togglebutton"
        android:layout_marginTop="56dp" />
</RelativeLayout>
```

MainActivity.java

```java
package com.example.p1;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.ToggleButton;

    public class MainActivity extends AppCompatActivity {
        Button b1;
        EditText et;
        ToggleButton tb;
        Double a;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        et=findViewById(R.id.edittext);
        b1=findViewById(R.id.button);
        tb=findViewById(R.id.togglebutton);
        b1.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
            if(tb.isChecked())
            {
                a=Double.parseDouble(String.valueOf(et.getText()));
                Double b=a*9/5+32;
                String r=String.valueOf(b);
                Toast.makeText(MainActivity.this,r+"°F",Toast.LENGTH_SHORT).show();
            }
            else
            {
                a=Double.parseDouble(String.valueOf(et.getText()));
                Double b=a-32;
                Double c=b*5/9;
                String r=String.valueOf(c);

Toast.makeText(MainActivity.this,r+"°C",Toast.LENGTH_SHORT).show();
            }
```

_____

		``
		          }
		      });
		    }
		}
		``	

	b)	**Write a program to capture an image using camera and display it.** *(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.)*	**6 M**

	Ans	**activity_main.xml**
		
		``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>
		``
			XML file:
			2M
			
			Java Code:
			4M

MainActivity.java

```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)	**Develop and application to send and receive SMS (Design minimal UI as per your choice. Write XML, java and manifest file)** *(Note: Consider appropriate XML file. All attributes are not required. In java file all imports are not expected. Different relevant login/code can be considered. Statements showing permissions can be written under AndroidManifest.xml)*	**6 M**
Ans	**Permissions and \<receiver\> tag required in AndroidManifest.xml**	XML file: 1M

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.WRITE_SMS"/>

<receiver
   android:name=".SmsReceiver"
   android:enabled="true"
   android:exported="true">
   <intent-filter>
     <action android:name="android.provider.Telephony.SMS_RECEIVED" />
   </intent-filter>
</receiver>
```

Manifest File:1M

Java Code: 4M

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"
     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" />
```

```xml
<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"
    tools:ignore="SpeakableTextPresentCheck" />

<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

```java
package com.example.testreceivesms;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
```

```
public class MainActivity extends AppCompatActivity {
SmsReceiver sms= new SmsReceiver();
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();
            }
         }
      });
   }
   @Override
   protected void onStart() {
      super.onStart();
      IntentFilter filter=new
IntentFilter("android.provider.Telephony.SMS_RECEIVED");
      registerReceiver(sms,filter);
   }

   @Override
   protected void onStop() {
      super.onStop();
      unregisterReceiver(sms);
```

_____

		``
		      }
		    }
		
		SmsReceiver.java
		package com.example.testreceivesms;
		import android.content.BroadcastReceiver;
		import android.content.Context;
		import android.content.Intent;
		import android.os.Bundle;
		import android.telephony.SmsMessage;
		import android.widget.Toast;
		
		public class SmsReceiver extends BroadcastReceiver {
		   SmsReceiver(){}
		   @Override
		   public void onReceive(Context context, Intent intent) {
		      Bundle bundle = intent.getExtras();
		
		      if (bundle != null) {
		         // Retrieve the SMS Messages received
		         Object[] sms = (Object[]) bundle.get("pdus");
		
		         // For every SMS message received
		         for (int i=0; i < sms.length; i++) {
		            // Convert Object array
		            SmsMessage smsMessage = SmsMessage.createFromPdu((byte[]) sms[i]);
		
		            String phone = smsMessage.getOriginatingAddress();
		            String message = smsMessage.getMessageBody().toString();
		
		            Toast.makeText(context, "Received from "+ phone + ": " + message,
		Toast.LENGTH_SHORT).show();
		         }
		      }
		   }
		}
		``	

6.		**Attempt any __TWO__ of the following:**	**12 M**
	a)	**Write a program to implement Android Activity Life Cycle. Use toast messages to display message through life cycle.** *(Note: No XML code is required. In java file all imports are not expected.)*	**6 M**

	Ans	```
		package com.example.p1;
		import androidx.appcompat.app.AppCompatActivity;
		import android.os.Bundle;
		import android.widget.Toast;
		   public class MainActivity extends AppCompatActivity {
		      @Override
		``
			Use of any 6 methods of Activity life cycle : 1 M each

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toast.makeText(getApplicationContext(),"Activity
created",Toast.LENGTH_LONG).show();
}

    @Override
    protected void onStart() {
        super.onStart();
        Toast.makeText(getApplicationContext(),"Activity
Started",Toast.LENGTH_LONG).show();
    }

    @Override
    protected void onStop() {
        super.onStop();
        Toast.makeText(getApplicationContext(),"Activity
Stop",Toast.LENGTH_LONG).show();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Toast.makeText(getApplicationContext(),"Activity
Destroy",Toast.LENGTH_LONG).show();
    }

    @Override
    protected void onPause() {
        super.onPause();
Toast.makeText(getApplicationContext(),"Activity
Pause",Toast.LENGTH_LONG).show();
    }
@Override
    protected void onRestart() {
        super.onResume();
        Toast.makeText(getApplicationContext(),"Activity
Restart",Toast.LENGTH_LONG).show();

    @Override
    protected void onResume() {
        super.onResume();
        Toast.makeText(getApplicationContext(),"Activity
Resume",Toast.LENGTH_LONG).show();
    }
}
```

| | b) | **Develop an application to display Google map with user's current location.** *(Note : Consider the appropriate XML file. All attributes are not required.* | **6 M** |

		In java file all imports are not expected. Different relevant logic/code can be considered.)	
	Ans	**act ivity_main.xml**	XML file: 1M

act ivity_main.xml

```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

```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;
```

Java Code: 5M

```java
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()+""+current
location.getLongitude(), Toast.LENGTH_SHORT).show();
                SupportMapFragment supportMapFragment =
(SupportMapFragment)getSupportFragmentManager().findFragmentById(R.id.go
ogle_map);
                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;
    }
  }
}
``` | |
| | c) | **Design UI using table layout to display buttons with 0 9 numbers on it. Even display submit and clear button. When user clicks on particular buttons and later when clicks on submit button, it should display the numbers clicked.**
(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.) | **6 M** |
| | Ans | **activity_main.xml**
```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout 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:stretchColumns="*"
    tools:context=".MainActivity">
    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <Button
``` | XML file: 3M

Java Code: 3M |

```
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:id="@+id/button0"
        android:text="0"/>
    <Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:id="@+id/button1"
        android:text="1"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button2"
        android:text="2" />
</TableRow>
<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:id="@+id/button3"
        android:text="3"/>
    <Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:id="@+id/button4"
        android:text="4"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="5"
        android:id="@+id/button5"/>
</TableRow>
<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="6"
        android:id="@+id/button6"/>
    <Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="7"
```

_____

```xml
              android:id="@+id/button7"/>

          <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="8"
            android:id="@+id/button8"/>

    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <Button
          android:layout_height="wrap_content"
          android:layout_width="wrap_content"
          android:text="9"
          android:id="@+id/button9"/>
        <Button
          android:layout_height="wrap_content"
          android:layout_width="wrap_content"
          android:text="Submit"
          android:id="@+id/submit"/>
        <Button
          android:layout_height="wrap_content"
          android:layout_width="wrap_content"
          android:text="Clear"
          android:id="@+id/clear"/>
    </TableRow>
</TableLayout>
```

(Java File)
```java
package com.example.p1;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

  public class MainActivity extends AppCompatActivity {
      Button button0, button1, button2, button3, button4, button5, button6,button7,
button8, button9,submit,clear;
    String a=null;
      @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      button0 = (Button) findViewById(R.id.button0);
      button1 = (Button) findViewById(R.id.button1);
      button2 = (Button) findViewById(R.id.button2);
```

_____

```java
button3 = (Button) findViewById(R.id.button3);
button4 = (Button) findViewById(R.id.button4);
button5 = (Button) findViewById(R.id.button5);
button6 = (Button) findViewById(R.id.button6);
button7 = (Button) findViewById(R.id.button7);
button8 = (Button) findViewById(R.id.button8);
button9 = (Button) findViewById(R.id.button9);
submit=(Button) findViewById(R.id.submit);
clear=(Button) findViewById(R.id.clear);
button0.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        a=button0.getText().toString();
    }
});
button1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        a=button1.getText().toString();
    }
});
    button2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

            a=button2.getText().toString();
        }
    });
    button3.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

            a=button3.getText().toString();
        }
    });
    button4.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

            a=button4.getText().toString();
        }
    });
    button5.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

            a=button5.getText().toString();
```

```
                    }
                });
    button6.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {

          a=button6.getText().toString();
      }
    });
    button7.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
          a=button7.getText().toString();
      }
    });
    button8.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {

          a=button8.getText().toString();
      }
    });
    button9.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {

          a=button9.getText().toString();
      }
    });
  submit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

Toast.makeText(getApplicationContext(),a,Toast.LENGTH_LONG).show();         }
    });
  }
  }
```