IV/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION – Solutions

November, 2016 **Electronics & Communication Engg.**

Sixth Semester **Mobile Application Development**

Time: Three Hours **Maximum :** 60 Marks

Answer Question No.1 compulsorily. (1 X 12 = 12 Marks)

Answer ONE question from each unit. (4 X 12 = 48 Marks)

(12 X 1 =12 Marks)

1. a) Define scope and life time of a variable.
b) What is meant by a constructor?
c) Define an abstract class.
d) Explain package. Give the examples of different packages in Java.
e) Write a Java program to display the date.
f) Explain the difference between throw and throws keywords.
g) What is an Android?
h) Define Android Views.
i) Screen Orientation.
j) Explain Shared Preferences.
k) What is the role of XML in Android application?
l) APK file.

UNIT-1

2 a) Define polymorphism. Write a Java program to illustrate method overloading. [4M]
2 b) Define Inheritance. Explain how multiple inheritance can be implemented in Java. [8M]

(OR)

3 a) Explain final keyword with suitable examples in Java. [4M]
3 b) Write a Java program to illustrate different types of parameter passing mechanisms. [8M]

UNIT-II

4 a) Explain File streams. [4M]
4 b) Define exception handling. Write a Java program to explain user defined exceptions. [8M]

(OR)

5 a) Explain about packages in Java. [4M]
5 b) Write a Java program to explain different String class methods. [8M]

UNIT-III

6 a) Describe Android application architecture. [4M]
6 b) Explain the components of Android application. [8M]

(OR)

7 a) Explain briefly Android development environment. [4M]
7 b) Explain different Android user interface controls. [8M]

UNIT-IV

8 a) What are the methods of Camera class and explain them. [4M]
8 b) Explain the fundamental steps involved in developing an application to use database. [8M]

(OR)

9 a) Develop an Android application for creating and using Service. [6M]
9 b) Discuss about Sensors in Android. [6M]

IV/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION – Solutions

November, 2016 **Electronics & Communication Engg.**

Sixth Semester **Mobile Application Development**

Time: Three Hours **Maximum :** 60 Marks

Answer Question No.1 compulsorily. (1 X 12 = 12 Marks)

Answer ONE question from each unit. (4 X 12 = 48 Marks)

(12 X 1 =12 Marks)

12 X 1M =12Marks

1.

a) Define scope and life time of a variable.

Ans) A scope determines what objects are visible to other parts of your program.
The lifetime of a variable or object is the time period in which the variable/object has valid memory.

b) What is meant by a constructor?

Ans) Constructor in java is a *special type of method* that is used to initialize the object. Java constructor is *invoked at the time of object creation*. It constructs the values i.e. provides data for the object that is why it is known as constructor.

c) Define an abstract class.

Ans) A class that is declared with abstract keyword, is known as abstract class in java. It can have abstract and non-abstract methods (method with body).

d) Explain package. Give the examples of different packages in Java.

Ans) A java package is a group of similar types of classes, interfaces and sub-packages. Package in java can be categorized in two form, built-in package and user-defined package. There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

e) Write a Java program to display the date.

Ans)
```
class DateDispay{
    public static void main(String args[])
    {
        Date date=new Date();
        System.out.println(date);
    }
```

f) Explain the difference between throw and throws keywords.

Ans) throw keyword is used to throw an exception manually. A throws clause lists the types of exceptions that a method might throw and it is not handling them.

g) What is an Android?

Ans) Android is a mobile operating system that is based on a modified version of Linux.

h) Define Android Views.

Ans) An Android is a User Interface component like button, Textbox, menu etc…

i) Screen Orientation.

Ans**)** Android supports two screen orientations: *portrait* and *landscape*. By default, when you change the display orientation of your Android device, the current activity that is displayed will automatically redraw its content in the new orientation.

j) Explain Shared Preferences.

Ans) Shared Preferences are used to store small amount of application data such as user preferences.

k) What is the role of XML in Android application?

Ans) layouts of various UI screen are stored in XML files. The Android configuration information is also stored in the .xml file.

l) APK file.

Ans) It is an Android Packaging file which is created after building your Android application.

UNIT-1

2 a) Define polymorphism. Write a Java program to illustrate method overloading. [4M]

definition→ 1Mark
Program → 3 Marks
Any relevant program may also be written

Polymorphism (from Greek, meaning "many forms") is a feature that allows one interface to be used for a general class of actions. The specific action is determined by the exact nature of the situation.

Program:

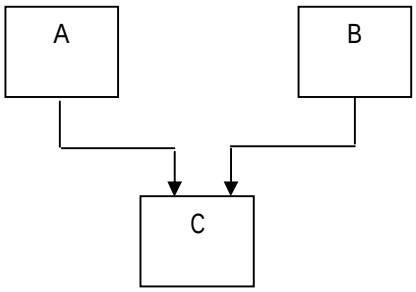
```
class overload
{
        int square(int x)
        {
        return x*x;
        }
        double square(double x)
        {
        return x*x;
        }
}
class check
{
        public static void main(String[] args)
        {
                overload r1=new overload();
                int k=r1.square(3);
                double p=r1.square(4.0);
                System.out.println("k value="+k);
                System.out.println("p value="+p);
        }
}
```

2. b) Define Inheritance. Explain how multiple inheritance can be implemented in Java. [8M]

definition→ 1Mark
explanation with example → 3+4 Marks
Any relevant program may also be written

Ans) The mechanism of deriving a new class from an old one is called inheritance. A class that is inherited (old class) is called a super class or base class or parent class. The class that does the inheriting (new class) is called a subclass or derived class or child class.

Multiple Inheritance: Derivation of one class from two or more super classes is called Multiple Inheritance. But Java does not support Multiple Inheritance directly. It can be implemented by using Interface concept.



Program:

```
class student  //program in java to show multiple inheritance
{
int rollNumber;
void getNumber(int n)
{
        rollNumber=n;
}
void printNumber()
{
        System.out.println("RollNo is " +rollNumber);
}
}

class test extends student
{
        float part1,part2;
        void getMarks(float a, float b)
        {
                part1=a;
                part2=b;
        }
        void putMarks()
        {
                System.out.println("Marks Part1 "+part1);
                System.out.println("Marks Part2 "+part2);
        }
}
```

```
interface sports
{
        float sportwt=6.0F;
        void putwt();
}
class results extends test implements sports
{
        float total;
        public void putwt()
        {
                System.out.println("Sports Marks "+ sportwt);
        }
void display()
{
        total=part1+part2+sportwt;
        System.out.println("Total marks of " +rollNumber+" is "+total);
}
}
class mainClass
{
public static void main(String srgs[])
{
        results a=new results();
        a.getNumber(10);
                a.printNumber();
        a.getMarks(10.0F,25.5F);
        a.putMarks();
        a.putwt();
        a.display();
}
}
```

In the above program, The results class using both the members of Test class as well as it is implementing Sports interface members. In that way we can achieve multiple inheritance in Java.

(OR)

3 a) Explain final keyword with suitable examples in Java. [4M]

final keyword explanation → 2Marks
examples → 2Marks
Any Relevant Examples may also be written

Ans) A final is a keyword used for three purposes. They are

a) **final as constant:** A variable can be declared as constant with the keyword final. It must be initialized while declaring. One we initialized the variable with final keyword it cannot be modified in the program. This is similar to the const in C/C++.

```
Example:    final int x = 10;
            x = 45 //error
```

b) final to prevent overriding: A method that is declared as final cannot be overridden in a subclass method. It the methods that are declared as private are implicitly final.

Example:
```
class A
{
        final void show( )
        {
        System.out.println("Hello");
        }
}
class B extends A
{
        void show( )
        {                                   //error
        System.out.println("Hai");
        }
}
```

c) final to prevent inheritance: The class that is declared as final implicitly declares all of its methods as final. The class cannot be extended to its subclass.

Example:
```
final class A
{
        void show( )
        {
        System.out.println("Hello");
        }
}
class B extends A                   //error
{
        void show( )
        {
        System.out.println("Hai");
        }
}
```

3. b) Write a Java program to illustrate different types of parameter passing mechanisms. [8M]

Call by value mechanism program → 4 Marks
call by reference mechanism program → 4 Marks
Any Relevant Example program may also be written

Ans) **Argument Passing:** Arguments passed to a sub-routine (method) is falls into two categories.

1. Call by value
2. Call by reference

1. Call by Value: Call by value copies the value of an argument into the formal parameter of the sub-routine. Therefore, if changes made to the parameter of the sub-routine have no effect on the argument used to call it.

Example:

```
class Test
{
        void meth(int i, int j)
        {
        i*=2;
        j/=2;
        }
}
class cbv
{
        public static void main(String[] args)
        {
        Test obj=new Test();
        int a=4,b=7;
        System.out.println("Before calling a="+a+" b="+b);
        obj.meth(a,b);
        System.out.println("After  calling a="+a+" b="+b);
        }
}
```

O/P: Before calling a=4 b=7
After calling a=4 b=7

2. ***Call by Reference:*** In this method, a reference to an argument is passed to the parameter. Inside the parameter, this reference is used to access the actual argument specified in the call. Therefore, any changes made to the parameter will effect the argument used to call it. This one is also referred as passing an object as parameter to a method.

Example:

```
class Test
{
        int a,b;
        Test(int i,int j)
        {
        a=i;
        b=j;
        }
        void meth(Test ox)
        {
        ox.a*=2;
        ox.b/=2;
        }
}
class cbr
{
        public static void main(String[] args)
        {
        Test obj=new Test(4,5);
        System.out.println("Before calling a="+obj.a+" b="+obj.b);
```

```
        obj.meth(obj);
        System.out.println("After  calling a="+obj.a+" b="+obj.b);
        }
}
```

O/P: Before calling a=4 b=5
After calling a=8 b=2

UNIT-II

4 a) Explain File streams. [4M]

File streams explanation → 4Marks

Ans) File streams can be byte oriented or character oriented.

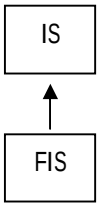
Byte oriented streams are FileInputStream which is used for reading data in terms of bytes from a file and FileOutputStream which is used for writing data in terms of bytes to a file opened for writing.

FileInputStream

FileInputStream is derived from **InputStream** and used to read bytes of data from a file. Constructors of FileInputStream are

FileInputStream (String filepath)
FileInputStream (File fileobj)

Here, filepath is the full path name of the file
fileobj is a File object that describes the file.

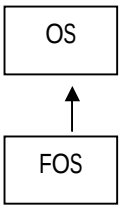


FileOutputStream

FileOutputStream is derived from **OutputStream** and used to send bytes of data into a file. Constructors of FileOutputStream are

FileOutputStream (String filepath)
FileOutputStream (File fileobj)
FileOutputStream (String filepath, boolean append)

Here, filepath is the full path name of the file
fileobj is a File object that describes the file
if append is true, then the file is in append mode.



4. b) Define exception handling. Write a Java program to explain user defined exceptions. [8M]

Definition → 1Mark
user defined exceptions program & Explanation → (4 + 3)Marks
Any Relevant Example application is correct

Ans) Exception handing: An exception is an abnormal event that rises during the execution of a program and disturbs the normal flow of instructions i.e., exception is a run-time error. Handling of raised abnormal events is called exception handling.

Example program:

```
// This program creates a custom exception type.
class MyException extends Exception {
        private int detail;
        MyException(int a) {
                detail = a;
        }
        public String toString() {
                return "MyException[" + detail + "]";
        }
        }
        class ExceptionDemo {
                static void compute(int a) throws MyException {
                        System.out.println("Called compute(" + a + ")");
                        if(a > 10) throw new MyException(a);
                        System.out.println("Normal exit");
                        }
                        public static void main(String args[]) {
                        try {
                                compute(1);
                                compute(20);
                        } catch (MyException e) {
                        System.out.println("Caught " + e);
                }
        }
}
```

This example defines a subclass of Exception called MyException. This subclass is quite simple: it has only a constructor plus an overloaded toString( ) method that displays the value of the exception. The ExceptionDemo class defines a method named compute( ) that throws a MyException object. The exception is thrown when compute( )'s integer parameter is greater than 10. The main( ) method sets up an exception handler for MyException, then calls compute( ) with a legal value (less than 10) and an illegal one to show both paths through the code.

Here is the **result:**

```
Called compute(1)
Normal exit
Called compute(20)
Caught MyException[20]
```

(OR)

5 a) Explain about packages in Java. [4M]

Definition & creation of packages → 2Marks
Importing packages → 2 Marks

Ans)

Package is a collection of related classes.

Defining a Package:

To define a package, place "package" command as the first statement in the Java source file. So, that any class declared within that file will belong to the specified package. The syntax of package creation is as follows.

Syntax: package pack-name;

where pack-name is the name of the package.

Example:

```
package mypack;
public class number
{
        public void add (int a,int b)
        {
        System.out.println("Sum="+(a+b));
        }
}
```

- The class that is defined in the package must be start with the public access modifier. So, that it can be accessible by any another of them. If it is not public, it is possible to access only in that package.
- Java uses file system directories to store packages. We save the program with number.java and compile the package is as

  javac –d number.java

  Due to this compilation mypack directory is automatically created and .class file is stored in that directory.
- Package creation has completed. The package information is now including in our actual program by means of "import" statement. "import" is a keyword that links the package with our program. It is placed before the class definitions.

```
import mypack.*;
        or
import mypack.number;
```

5. b) Write a Java program to explain different String class methods. [8M]

Java program demonstrating String class	**→ 6Marks**
Explanation of methods used	**→ 2Marks**
Any Relevant program may also be written	

Ans)

```
class IndexOfDemo {
        public static void main(String args[]) {
                String s = "Now is the time for all good men " +
                "to come to the aid of their country.";
                System.out.println(s);
                System.out.println("length="+s.length());
                System.out.println("indexOf(t) = " +s.indexOf('t'));
                System.out.println("lastIndexOf(t) = " +s.lastIndexOf('t'));
                System.out.println("indexOf(the) = " +s.indexOf("the"));
                System.out.println("lastIndexOf(the) = " +s.lastIndexOf("the"));
                System.out.println("indexOf(t, 10) = " +s.indexOf('t', 10));
                System.out.println("lastIndexOf(t, 60) = " +s.lastIndexOf('t', 60));
                System.out.println("indexOf(the, 10) = " +s.indexOf("the", 10));
                System.out.println("lastIndexOf(the, 60) = " +s.lastIndexOf("the", 60));
        }
}
```

o/p:

```
E:\MAD 2016_17\java2016exs\String>java IndexOfDemo
Now is the time for all good men to come to the aid of their country.
length=69
indexOf(t) = 7
lastIndexOf(t) = 65
indexOf(the) = 7
lastIndexOf(the) = 55
indexOf(t, 10) = 11
lastIndexOf(t, 60) = 55
indexOf(the, 10) = 44
lastIndexOf(the, 60) = 55

E:\MAD 2016_17\java2016exs\String>
```

String class methods:

Method	Description
int indexOf(int ch)	This method returns the index within this string of the first occurrence of the specified character.
int indexOf(String str)	This method returns the index within this string of the first occurrence of the specified substring.
int lastIndexOf(int ch)	This method returns the index within this string of the last occurrence of the specified character.
int length()	returns number of characters in the string.

UNIT-III

6 a) Describe Android application architecture. [4M]

Application architecture → 4 Marks

Ans)

Every Android Application contains the following folders. Each folder contains standard file(s) which is used for a specific purpose. Description regarding each folder and its contents is given below:

src folder : Contains the .java source files for your project. Inthis example, there is one file, MainActivity.java. TheMainActivity.java file is the source file for your activity.You will write the code for your application in this file.

Android 4.2.2 library : Contains android.jar, which contains all the class libraries needed for your appl.

gen folder: contains R.java → a compiler-generated file which references all the resources found in your project. You should not modify this file.

assets : Contains all the assets (html, text files and databases)used by your project.

res: Folder which contains all the resources used in your prj. Contains drawable, values, layout folders etc..

AndroidManifest.xml : Specifies permissions for your application and other features such as intent-filters and receivers.

6 b) Explain the components of Android application. [8M]

4 components, for each component 2 Marks → 8 Marks

Ans) Components of Android application are Activities, Intents, Services, Broadcast Receivers, Content Providers.
Activities → They dictate the UI and handle the user interaction to the smart phone screen.
Services → They handle background processing associated with an application.
Broadcast Receivers → They handle communication between Android OS and applications.
Content Providers → They handle data and database management issues.

Activities :

An activity represents a single screen with a user interface,in-short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of **Activity** class as follows –

```
public class MainActivity extends Activity{

}
```

Services:

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.

A service is implemented as a subclass of Service class as follows –

```
public class MyService extends Service{

}
```

Broadcast Receivers:

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

A broadcast receiver is implemented as a subclass of BroadcastReceiver class and each message is broadcaster as an Intent object.

```
public class MyReceiver extends BroadcastReceiver{

  public void onReceive(context, intent){ }

}
```

Content Providers:

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver*class. The data may be stored in the file system, the database or somewhere else entirely.

A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends ContentProvider{
        public void onCreate(){ }
}
```

Intents:

Intents are used to

- ✓ call built in applications
- ✓ Pass data to an activity
- ✓ To return data from an activity

An intent contains action and data. The **action** describes what is to be performed and **data** specifies what is affected such as a person in the contacts database. The data is specified as an Uri object.

Some examples of action are as follows:

- ✓ ACTION_VIEW
- ✓ ACTION_DIAL
- ✓ ACTION_PICK
- ✓ ACTION_CALL

Some examples of data include the following:

- ✓ http://www.google.com
- ✓ tel:+651234567
- ✓ geo:37.827500,-122.481670
- ✓ content://contacts

An Android application may also contains fragments, layouts, resources, manifest.

(OR)

7 a) Explain briefly Android development environment. [4M]

Application development using eclipse → 4 Marks

Ans)

The first step towards developing any applications is obtaining the integrated development environment (IDE) such as Eclipse or Android Studio.

Next Install Android Studio which contains a debugger, libraries, an emulator, documentation, sample code, and tutorials. Next install The Android Development Tools (ADT) plug-in for Eclipse is an extension to the Eclipse IDE that supports the creation and debugging of Android applications. Using the ADT, you will be able to do the following in Eclipse:

- ➢ Create new Android application projects.
- ➢ Access the tools for accessing your Android emulators and devices.
- ➢ Compile and debug Android applications.
- ➢ Export Android applications into Android Packages (APK).
- ➢ Create digital certificates for code-signing your APK.

In Eclipse you can find Android Package Manager to install necessary packages. Emulator in Eclipse is used for running application on the virtual device instead on the physical device. There are several features in the Eclipse for android development like intelli sense, debugging features like single stepping, run to cursor etc..

7 b) Explain different Android user interface controls. [8M]

basic views → 4 Marks
picker views → 2 Marks
List views → 2 Marks

Ans) Android supports 3 types of view groups:

- **Basic views** — Commonly used views such as the TextView, EditText, and Button Views
- **Picker views** — Views that enable users to select from a list, such as the TimePicker And DatePicker views
- **List views** — Views that display a long list of items, such as the ListView and the SpinnerView views

Basic Views:

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

- **Button** → Represents a push-button widget
- **ImageButton** → Similar to the Button view, except that it also displays an image
- **EditText** → A subclass of the TextView view, except that it allows users to edit its text content
- **CheckBox** → A special type of button that has two states: checked or unchecked
- **RadioGroup** and **RadioButton** → The RadioButton has two states: either checked or unchecked. Once a RadioButton is checked, it cannot be unchecked. A RadioGroup is used to group together one or more RadioButton views, thereby allowing only one RadioButton to be checked within the RadioGroup.
- **ToggleButton** → Displays checked/unchecked states using a light indicator

Processing Button events:

Step 1: In activity_main.xml file, add the following code:

```
<Button
 android:id="@+id/button1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/button_str"
 android:textColor="@android:color/background_dark"
 android:textSize="15sp" />
```

Step 2: in MainActivity.java file onCreate( ) method, add the following code :

```
Button btn=(Button)findViewById(R.id.button1);
btn.setOnClickListener(new View.OnClickListener() {
public void onClick(View arg0) {
Toast.makeText(getBaseContext(),"Clicked Button",  Toast.LENGTH_SHORT).show();
}
});
```

When you run the program, after clicking button, a toast message “Clicked Button” will be displayed.

Creating and using TimePicker view:

Step 1: in activity_main.xml file, add the following code:

```
<TimePicker
 android:id="@+id/timePicker1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentLeft="true"
 android:layout_alignParentTop="true"
 android:layout_marginTop="41dp" />
```

Step 2: In MainActivity.java file onCreate() method, add the following code:

```
tpicker=(TimePicker)findViewById(R.id.timePicker1);
      // button
       Button btn=(Button)findViewById(R.id.button1);
       btn.setOnClickListener(new View.OnClickListener() {


       @Override
 public void onClick(View arg0) {
       String str="Time Selected :"+tpicker.getCurrentHour().toString()
       +"  "+tpicker.getCurrentMinute().toString();
       Toast.makeText(getBaseContext(),str, Toast.LENGTH_SHORT).show();
       }
 });
```

When you run the program, after selecting time from the TimePicker view, a toast message “Time Selected : 09:30” will be displayed.

List Views:

List views are views that enable you to display a long list of items. In Android, there are two types of list views: ListView and SpinnerView. Both are useful for displaying long lists of items.

Example:

Step 1: Add necessary imports in imports section of MainActivity.java file:

Step 2: change the base class of MainActivity to ListActivity

Step 3: in MainActivity class, add the following:

```
String[] presidents = {
                        "Dwight D. Eisenhower",
                        "John F. Kennedy",
                        "Lyndon B. Johnson"
                        };
```

Step 4: comment out setContentView() method and add the following in onCreate()

```
setListAdapter(new ArrayAdapter<String>(this,
                        android.R.layout.simple_list_item_1, presidents));
```

Step 5: Add the following method in the MainActivity class

```
public void onListItemClick(ListView parent, View v, int position, long id)
    {
        Toast.makeText(this,"You have selected " +
        presidents[position],Toast.LENGTH_SHORT).show();
    }
```

UNIT-IV

8 a) What are the methods of Camera class and explain them. [4M]

Class methods syntax → 2 Marks
Explanation → 2 Marks
Any 4 methods may be written

Ans)

Sl. No.	Method	Description
1	static Camera open()	Creates a new Camera object to access the first back-facing camera on the device.
2	static Camera open(int cameraID)	Creates a new Camera object to access a particular hardware camera.
3	final void release()	Disconnects and releases the Camera object resources.
4	final Boolean enableShutterSound(boolean enabled)	Enable or disable the default shutter sound when taking a picture.

5	final void cancelAutoFocus()	Cancels any auto-focus function in progress.
6	final void autofocus(AutoFocusCallback cb)	Starts camera auto-focus and registers a callback function to run when the camera is focused.
7	final void startPreview()	Starts capturing and drawing preview frames to the screen.
8	final void takepicture(ShutterCallback cb, PictureCallback pcb, PictureCallback pcb2)	Triggers an asynchronous image capture.

8. b) Explain the fundamental steps involved in developing an application to use database. [8M]

Ans)

8 steps with sample example → 8 Marks

Android uses the SQLite database system. For Android, the SQLite database that you create programmatically in an application is
always stored in the /data/data/<package_name>/databases folder.

Steps in using databases in Android:

- Create a new project in your Android Studio Go to File-> New ->New project and select Empty Activity from template and leave every thing as default.
- Add Five EditText for username,college,place,userId and number, also add a button to activity_main.xml.
- Create one more activity and add Recyclerview to its xml.
- Create a new Java class and name it DbHelper.
- Extend SQLiteOpenHelper in DbHelper class.
- Create a new Java class name it UserData and implement Serializable Interface to it and declare Five String varriable name,college,place,user_id and number in this class.
- Write method in DbHelper class to insert user data in to Database and to delete from Database.
- Write one more method to fetch all data from Database to show into Recyclerview
- Create a new Java class for Recyclerview Adapter and complete all the code for Recyclerview Adapter.
- Save user data in MainActivity on Button click and call second Activity.
- Show all the inserted data in Recyclerview of Second Activity.

(OR)

9 a) Develop an Android application for creating and using Service. [6M]

Application/ steps → 6 Marks
Any Relevant application may also be written

Ans) Steps:

1. Create an Android Application.
2. Add a Class called MyService which extends from Service class. A java file named MyService.java will be created.
3. In onStartCommand method of MyService class add the following code:

```
    public int onStartCommand(Intent intent, int flags, int startId) {
    // We want this service to continue running until it is explicitly
    // stopped, so return sticky.
    Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
    return START_STICKY;
}
```

4. In the AndroidManifest.xml file, add the following code:

```
<application

---
<service android:name=".MyService" />
</application>
```

5. Add two buttons(Start & Stop) to start and stop the service in the main.xml file
6. In the Start Button 'onClick' event handler, add the following code:

```
public void onClick(View v) {
    startService(new Intent(getBaseContext(), MyService.class));
}
```

7. In the Stop Button 'onClick' event handler, add the following code:

```
public void onClick(View v) {
    stopService(new Intent(getBaseContext(), MyService.class));
}
```

8. Run the application by clicking CTRL + F11. On Clicking the Start button, service will be started. On Clicking the Stop button, service will be stopped.

9 b) Discuss about Sensors in Android. [6M]

different Sensors Explanation → 2 Marks
using sensors in Android → 4Marks

Ans)

Android contains different sensors:

1. **Accelerometer:** Measures acceleration along various axes.
2. **Gyroscope:** Measures rotational change along some axes.
3. **Magnetic field sensor:** Measures the strength of magnetic fields along a set of axes.
4. **Light sensor:** Measures amount of ambient light.
5. **Proximity sensor:** Measures external object's proximity to the device.
6. **Temperature sensor:** Measures ambient temperature.
7. **Pressure Sensor:** Acts as a barometer.

To access a sensor or set of sensors, Android provides a convenient system service called the SensorManager. This can be accessed via the getSystemService() method of the Context with the argument of Context.SENSOR_SERVICE. With the SensorManager you then can get a specific sensor via the getDefaultSensor() method.

However, a composite sensor may sometimes be returned, so if you wish to get access to the raw sensor and its associated data, you should use getSensorList():

```
SensorManager mngr =
(SensorManager) context.getSystemService(Context.SENSOR_SERVICE);
// getting the default accelerometer
Sensor accel = mngr.getDefaultSensor (Sensor.TYPE_ACCELEROMETER);
// getting the raw accelerometer
List<Sensor> list = mngr.getSensorList(Sensor.TYPE_ACCELEROMETER);
```

Once you get a sensor or set of sensors, you can actually enable them and start getting their data by registering a listener against the sensors. Data should begin to come in at the rate you give as an argument. This rate can be SENSOR_DELAY_NORMAL, SENSOR_DELAY_UI (a rate appropriate for basic UI interaction), SENSOR_DELAY_GAME (a high rate that many games would find sufficient), SENSOR_DELAY_FASTEST ("give it to me as fast as you can"), or a specified delay between events in units of milliseconds:

```
SensorEventListener listener = new SensorEventListener() {
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) { }
@Override
public void onSensorChanged(SensorEvent event) { }
};
// registering a listener

mngr.registerListener(listener, sensor, SensorManager.SENSOR_DELAY_UI);
```

The two methods in a SensorEventListener—onAccuracyChanged() and onSensorChanged()—are called when data from the sensor in question is available. onAccuracyChanged() is called whenever a change to the degree of error or accuracy with the sensor occurs.

Signature of the Internal Examiner	**Signature of the HOD**	**Signature of the External Examiner**