(a) Assume you have a running application that is already running an Activity called Activity 1. Activity I starts another Activity called Activity2. Name one Activity lifecycle method that will be called on Activityl after this point, but before Activity2 starts. (2 Marks) (b) Suppose you have an application that is running an Activity called Activity1. Suppose that Activityl executes and starts other Activities, but that the user never quits or backs out of the Activity. How many times can Activity I's onCreate() method get called? (2 Marks)

(c) Suppose that there' are two activities in an application named ActivityOne and Activity Two. You want to invoke Activity Two from ActivityOne. What code you will write? (3 Marks)

(d) How will you reference a textbox control in java file, that is available in XML file and the ID is txtName. (4 Marks)

(e) Suppose that there are two activities in an application named FirstActivity and SecondActivity. You want to send website name from ActivityOne to ActivityTwo. What code you will write? Suppose that website name is www.udom.ac.tz. (5 Marks) (f) How will you get the data in secondActivity? Refer to part (e). (4 Marks)

a) One Activity lifecycle method that will be called on Activity1 after it starts Activity2 but before Activity2 starts is `onPause()`.

(b) In this scenario, Activity1's `onCreate()` method can be called only once, when the Activity is initially created. Subsequent activities started by Activity1 will not trigger the `onCreate()` method of Activity1 again.

(c) To start ActivityTwo from ActivityOne, you would typically use an `Intent`. Here's an example of the code:

```
Intent intent = new Intent(ActivityOne.this, ActivityTwo.class);

startActivity(intent);
```

(d) To reference a TextBox control in a Java file that is available in an XML layout file with the ID `txtName`, you would use the `findViewById` method. Here's an example:

```
EditText txtName = findViewById(R.id.txtName);
```

(e) To send the website name from ActivityOne to ActivityTwo, you can use an Intent to pass data. Here's how you can do it in ActivityOne:

```
String websiteName = "www.udom.ac.tz";
```

```
Intent intent = new Intent(ActivityOne.this, ActivityTwo.class);

intent.putExtra("websiteName", websiteName);

startActivity(intent);
```

(f) To retrieve the website name in ActivityTwo, you would use the `getIntent()` method and then retrieve the data from the Intent. Here's how you can do it in ActivityTwo:

```
Intent intent = getIntent();

if (intent != null) {

    String websiteName = intent.getStringExtra("websiteName");

    // Now you have the websiteName value from ActivityOne

    // You can use it as needed.

}
```

Currently, UDOM SACCOS have no tool to automate its operations as a result there is a delay in providing services to its customers. You have been asked by the management of UDOOM SACCOS to develop a mobile application that will store the information of its customers. The app should store name, college, age, gender and monthly contribution of its customers. The management want to show the reports of how customers are contributing monthly. The management also wants to show the dates of different events within SACCOS such as the date of annual meeting etc. The management suggests the following features to be used: radio button for gender and spinner for college. Use progress bar to show the monthly contributions and calendar to show the dates of the events. Based on the given requirements, you found that the application has only two activities i.e. ActivityOne and Activity Two. (a) Using the methods of Intent class show how you will send the customer's data to the second activity. (5 Marks) (b) Using the methods of Intent class show how you will receive the customer's data within the second activity. (5 Marks) (c) By the help of SQLiteOpenHelper class, show how you can save the customer's details in a database. Use user and customer as the name of the database and table respectively.

(a) To send the customer's data to the second activity using the Intent class, you can put the data as extras in the Intent object. Here's how you can do it:

In ActivityOne:

```
// Assuming you have collected customer data in variables like name, college, age, gender, and monthlyContribution

Intent intent = new Intent(ActivityOne.this, ActivityTwo.class);

intent.putExtra("name", name);

intent.putExtra("college", college);

intent.putExtra("age", age);

intent.putExtra("gender", gender);

intent.putExtra("monthlyContribution", monthlyContribution);

startActivity(intent);
```

(b) To receive the customer's data within the second activity, you need to retrieve the data from the Intent object. Here's how you can do it in ActivityTwo:

In ActivityTwo:

```
// Retrieve customer data from the Intent

Intent intent = getIntent();

if (intent != null) {

    String name = intent.getStringExtra("name");

    String college = intent.getStringExtra("college");

    int age = intent.getIntExtra("age", 0); // 0 is the default value if age is not provided

    String gender = intent.getStringExtra("gender");

    double monthlyContribution = intent.getDoubleExtra("monthlyContribution", 0.0); // 0.0 is the default value if monthlyContribution is not provided
```

```
    // Now you can use the retrieved data as needed in ActivityTwo

}
```

(c) To save the customer's details in a SQLite database using the `SQLiteOpenHelper` class, you would typically create a custom helper class that extends `SQLiteOpenHelper`. Here's an example of how to do it:

```
import android.content.Context;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;



public class MyDatabaseHelper extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "user.db"; // Database name

    private static final int DATABASE_VERSION = 1; // Database version



    // Table creation SQL statement

    private static final String CREATE_CUSTOMER_TABLE = "CREATE TABLE customer (" +

            "id INTEGER PRIMARY KEY AUTOINCREMENT," +

            "name TEXT," +

            "college TEXT," +

            "age INTEGER," +

            "gender TEXT," +

            "monthlyContribution REAL)";
```

```
    public MyDatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_CUSTOMER_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Implement if needed when the database schema changes in the future
    }
}
```

4.

define activities called Home.xml

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
```

```
 <EditText
 android:id="@+id/initialbalance"
 android:layout_width="wrap_content"
 android:layout_height ="wrap_content"
 android:hint="enter intial blance"/>

<EditText
 android:id="@+id/ndays"
 android:layout_width="wrap_content"
 android:layout_height ="wrap_content"
 android:hint="enter numbers of days"/>

<EditText
 android:id="@+id/spend"
 android:layout_width="wrap_content"
 android:layout_height ="wrap_content"
 android:hint="enter spend"/>

<Button
 android:id="@+id/btnsave"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text ="save"/>

</LinearLayout>


//Home.java
```

```
public class Helsb extends AppCompactActivity{

Button btnsave;
EditText iblance,ndays,mspend;
public void onCreate(Bundle flora){
super.onCreate(flora);
setContentView(R.layout.home);

 btnsave=findViewById(R.id.btnsave);
 iblance=findViewById(R.id.initialbalance);
 ndays=findViewById(R.id.ndays);
 mspend=findViewById(R.id.spend);

 btnsave.setOnClickListener((View v)->{

    //receive input and convert
    int x=Interger.parseInt(iblance.getText().toString());
    int y=Interger.parseInt(ndays.getText().toString());
    int z=Interger.parseInt(mspend.getText().toString());

    if(x>z){
    int balance=x-z;
     Toast.makeText(Home.this,"success",Toast.LENGTH_LONG ).show();
    }
    else{

      Toast.makeText(Home.this,"insufficient blance",Toast.LENGTH_LONG ).show();
```

```
    }
}
}
```

Android Fragment represents a behavior or a portion of a user interface in an activity. Multiple fragments can be combined in a single activity to build a multi-panel User Interface (UI) and reuse a fragment in multiple activities. As a mobile apps developer, you have been asked to develop an application with two fragments named FragmentOne and Fragment Two. When a user click on the specific fragment, the description of the fragment displays on the fragment container in the main activity which hosts the fragments. Assume the ids for FragmentOne and Fragment Two are btnFragment_one and btnFragment two respectively. Hint: use linear layout to create a fragment container.

fragment

define activities called Home.xml

```
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="#009EFa"
   android:orientation="vertical">
 <TextView
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:gravity="center"
       android:textStyle="bold"
       android:textSize="34sp"
       android:text="Fragment title"
  " />
   <Button
```

```
        android:id="@+id/frag_one"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="fragment one"
        android:textSize="25sp"/>

    <Button
        android:id="@+id/frag_two"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="fragment two"
        android:textSize="25sp" />

    <LinearLayout
        android:id="@+id/frament_container"
        android:layout_width="match_parent"
        android:background="#792020"
        android:orientation="vertical"
        android:layout_height="match_parent">



    </LinearLayout>
</LinearLayout>


//java
```

```
public class Basefragment extends AppCompactActivity{
Button fragone,fragtwo,;
public void onCreate(Bundle ney){
super.onCreate(ney);
setContentView(R.layout.home)
          fragone=findViewById(R.id.frag_one);
        fragtwo=findViewById(R.id.frag_two);
        moveBtn = findViewById(R.id.move);

fragone.setOnClickListener((View v)->{
//obejct fragment
Fragment frag=new Homefragment();
//object fragmentTransaction
FragmentTransaction ft=getSupportFragmentManager().beginTransaction();
ft.replace((R.id.fragment_container),frag);
ft.addTostack(null);
ft.commit();
});



}


}
```

Here are the statements with their correctness:

(a) False - Mobile apps on both Android and iOS platforms can perform long-lasting tasks, such as network operations. However, it's important to execute them in the background using threads or asynchronous mechanisms to avoid blocking the main UI thread.

(b) False - Android is not built upon the Java Micro Edition (J2ME) version of Java. Android uses a modified version of the Java programming language, and it has its own runtime environment and libraries.

(c) True - Native libraries are one of the core components of the .apk (Android Package) in Android. They contain compiled code that can be executed directly by the device's processor and are used for performance-critical operations.

(d) False - The code for an Android application is not contained within the XML layout file. XML layout files define the UI structure, while the application's logic and behavior are typically written in Java or Kotlin.

(e) False - The XML file that contains text resources used in an Android application is typically named `strings.xml`, not `text.xml`.

(f) True - There is no guarantee that an activity will be stopped (i.e., go through the `onStop()` lifecycle method) before being destroyed (i.e., go through the `onDestroy()` lifecycle method).

(g) False - In an explicit intent, the sender specifies a specific component (e.g., a class name) to which the intent is directed, but it does not specify the type of receiver. The type of receiver is determined by the component specified in the intent.

(h) False - Multiple activities can be running at the same time in an Android application, depending on the app's design and user interactions.

(i) False - While Java is a commonly used programming language for Android development, Kotlin is also a supported language for building Android applications.

(j) False - While SQLite is a popular database used in Android applications, Android apps can also work with other databases such as MySQL, Firebase Realtime Database, and more, depending on the requirements of the app.