National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	**Software For Mobile Devices**	**Course Code:**	**CS440**
Program:	**BS(CS)**	**Semester:**	**Spring 2019**
Duration:	**180 Minutes**	**Total Marks:**	**50**
Paper Date:	**16-05-2019**	**Weight**	**40**
Section:	**ALL**	**Page(s):**	**5**
Exam Type:	**Final**		

Student : Name:______________________________ Roll No.________________ Section:_______

Instruction/Notes: This is an Open book/notes exam. However the notes and books should only be in the hard form.

Question No. 1 [Marks: 15]

Attempt Question 1 on question paper and attach it to the answer sheet

Consider the code given in Figure 1 and Figure 2 for following two classes

1. MessengerService: A remote service, working in a separate process
2. MessengerActivity: An activity displaying 2 buttons, titled as "Bind Service" and "Unbind Service"
    a. "Bind Service" button's onClick is hooked to doBindService method
    b. "Unbind Service" button's onClick is hooked to doUnbindService method

What will be the output of Log.d commands when the following events happen(in sequence)

1. "Bind Service" button clicked
   Output:

2. "Unbind Service" button clicked
   Output:

```
public class MessengerActivity extends AppCompatActivity {

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

    Messenger mService = null;

        class IncomingHandler extends Handler {
            @Override
            public void handleMessage(Message msg) {
                switch (msg.what) {
                    case 3:
                        Log.d("Received From Service", String.valueOf(msg.arg1));
                            break;
                    default:
                        super.handleMessage(msg);
                }
            }
        }
        final Messenger mMessenger = new Messenger(new IncomingHandler());
        private ServiceConnection mConnection = new ServiceConnection() {
            public void onServiceConnected(ComponentName className,
                                           IBinder service) {
                mService = new Messenger(service);
                Log.d("Service connected", "successfully");
         try {
                    Message msg = Message.obtain(null,1);
                    msg.replyTo = mMessenger;
                    Log.d("Registering Activity to", "MessengerService");
                    mService.send(msg);
                    msg = Message.obtain(null,3, 5, 0);
                    Log.d("Sending data to ", "MessengerService");
                    mService.send(msg);
                } catch (RemoteException e) {
                }
            }

            public void onServiceDisconnected(ComponentName className) {
                mService = null;
            }
        };

        public void doBindService(View v) {
            Log.d("Binding Activity to", "MessengerService");
            bindService(new Intent(this,
                    MessengerService.class), mConnection, Context.BIND_AUTO_CREATE);
        }

        public void doUnbindService(View v) {
                if (mService != null) {
                    try {
                        Log.d("UnReg Activity from", "MessengerService");
                        Message msg = Message.obtain(null, 2);
                        msg.replyTo = mMessenger;
                        mService.send(msg);
                    } catch (RemoteException e) {

                    }
                }
                Log.d("UnBinding Activity to", "MessengerService");
                unbindService(mConnection);
            }
}
```

Figure: 1

```java
public class MessengerService extends Service {
    /** Keeps track of all current registered clients. */
    ArrayList<Messenger> mClients = new ArrayList<Messenger>();
    /** Holds last value set by a client. */
    int mValue = 0;
class IncomingHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
                case 1:
                    Log.d("MS", "registering Client");
                    mClients.add(msg.replyTo);
                    break;
                case 2:
                    Log.d("MS", "unregistering Client");
                    mClients.remove(msg.replyTo);
                    break;
                case 3:
                    Log.d("MS", "Setting Value");
                    mValue = msg.arg1;
                    for (int i=mClients.size()-1; i>=0; i--) {
                        try {
                            mClients.get(i).send(Message.obtain(null,
                                    3, mValue, 0));
                        } catch (RemoteException e) {
                            mClients.remove(i);
                        }
                    }
                    break;
                default:
                    super.handleMessage(msg);
            }
        }
    }
    final Messenger mMessenger = new Messenger(new IncomingHandler());
    @Override
    public void onCreate() {
        super.onCreate();
    }
    @Override
    public IBinder onBind(Intent intent) {
        Log.d("MessengerService", "onBind called");
        return mMessenger.getBinder();
    }

    @Override
    public boolean onUnbind(Intent intent) {
        Log.d("MessengerService", "onUnBind called");
        return super.onUnbind(intent);
    }
}
```

Figure: 2

For the following questions, carefully note the following instructions

a. You are not required to provide XML Layout files. Only the Java files/code is required.

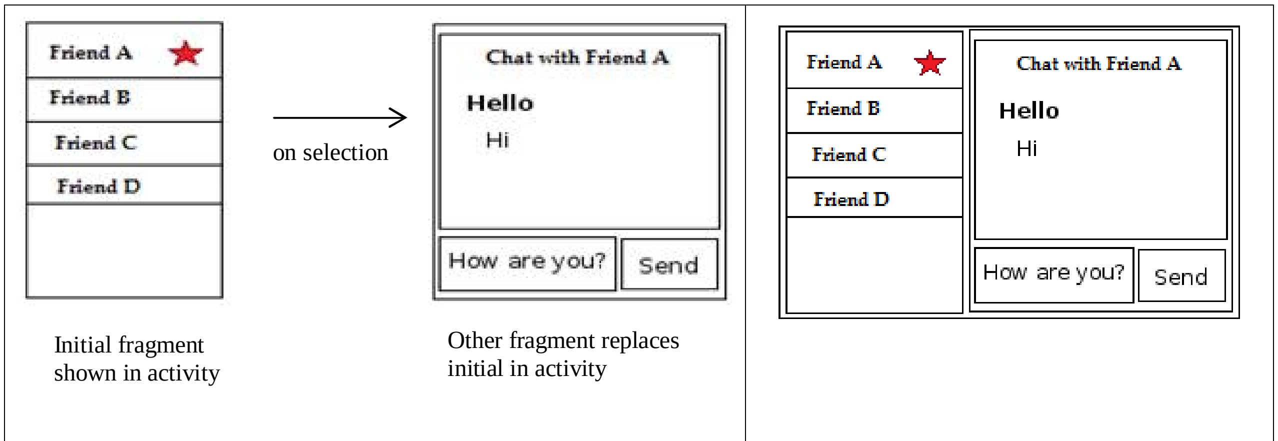
b. It must be noted that application code must function properly; scattered pieces of code that are not properly integrated may result in no credit.

c. No need to write setters/getters for any class. You can assume their availability and use them where required

d. Avoid writing code again and again for the same tasks. Use the usability features provided by android as much as possible. Concentrate on this part of the evaluation as it has very high weight-age.

Question No. 2 [Marks: 20]

Consider a Chat application. The UI needs to show a list of contacts. Against each contact, user can view history of chat messages exchanged as well as an option to send messages. To support responsive layouts, fragments are used. List of contacts is shown in one fragment whereas chat history and message exchange option in the other fragment. In the landscape mode, both fragments are shown side-by-side in the activity. In the portrait mode, contacts list fragment is shown first. Selecting a specific contact replaces the contacts list fragment with message history and exchange fragment. This is illustrated in the figure below:



(a) Portrait mode Both fragments are shown independently	(b) Landscape mode Both fragments are combined in activity

Provide Java code for:

- Each of the fragments, with proper UI handling
- Activity, while handling both portrait and landscape mode, and fragment transitions
- Communication between fragments

Question No. 3 [Marks: 15]

Hydrate is an app that reminds you to drink a glass of water around every 45 minutes for keeping you hydrated and healthy. It uses a notification to let you know when 45 minutes have passed.

Consider the details as follows:

1. The app has only one screen which allows user to turn on or off the reminders. The screen shows a toggle button that can turn the alarm on and off.
2. If user has kept the toggle button on, The app will periodically deliver the reminder to drink in the form of a notification.
3. A user can turn notifications off in following two ways
    a. As mentioned earlier, application's first screen will provide toggle button which can be switched off to stop getting reminders.
    b. The notification, shown as reminder, has a button "Stop" which opens the main activity of the application, sets the toggle button to off state and shows a toast displaying "You water intake notifications have been turned off"