Topic questions (1/2 mark each)

Please include your answers to theses questions in a separate PDF document, and ensure you submit it at the same time as your project.

Q1. Explain the concept of *out-of-band data* in socket communication. Will out-of-band data always be delivered reliably?

Out-of-band data (called "urgent data" in TCP) looks to the application like a separate stream of data from the main data stream. This can be useful for separating two different kinds of data.

Note that just because it is called "urgent data" does not mean that it will be delivered any faster, or with higher priority than data in the in-band data stream. Also beware that unlike the main data stream, the out-of-bound data may be lost if your application can't keep up with it.

Q2. Is it possible to force a socket to empty its data in the buffer? If so, how to do that?

Basically, you can't force it. TCP makes up its own mind as to when it can send data. Normally when you call write() on a TCP socket, TCP will indeed send a segment, but there's no guarantee and no way to force this. In practice, we can also indicate that we need data to be transmitted urgently. However, there is again no guarantee that socket will absolutely follow our instructions.

Q3. Explain when SYN segment is used in TCP communication. Can any SYN segment carry data payload and why?

SYN segment is used to establish TCP connections. SYN segment by right should not carry data payload because when it is used, the TCP connection hasn't been fully established yet and therefore data communication is forbidden.

(this part is optional) However, with new improvement to TCP, it is possible for SYN segment to carry data. But the data carried cannot be delivered to applications before the establishment of a connection. Meanwhile it is also important to note that, although this is considered as an enhancement, its support is limited in the Internet.

Q4. Consider hosts A and B communicating over a TCP connection. Assume unrealistically that the initial sequence number for each of A and B is 0 (after the handover process). Assume that all segments sent between A and B have 20 byte headers. A sends B a segment with a 100 byte payload, B responds with a segment with a 100 byte payload and then another segment with a 200 byte payload, and finally A responds with a segment with a 50 byte payload. Give the value of the sequence number field and acknowledgement number field for each segment.

A -> B 20-byte header + 100-byte payload, seq no=0, ack_no=0
B -> A 20-byte header + 100-byte payload, seq_no =0, ack_no=100
B -> A 20-byte header + 200-byte payload, seq_no=100, ack_no=100
A -> B 20-byte header + 50-byte payload, seq_no=100, ack_no=300

Q5. Suppose that a socket client running on an ARM processor is receiving an integer from a socket server running on an INTEL processor. Suppose also that the ARM processor stores an integer in 32 bits and the INTEL processor stores an integer in 64 bits. Can the socket client receive the correct integer from the socket server? Make sure that you explain your answers.

To make sure that the integer can be correctly transferred through the a socket, we need to ensure the following:

1. The length of integer to be transmitted must be commonly agreed by the sender and receiver. For example, we can ask the sender to transmit only 32-bit integers although it supports integers up to 64 bits.

2. The byte-order of each integer needs to be ensured. We need to first of all convert an integer to be sent to the network byte order. Later at the receiver side, we need to convert the integer received from the network byte order back to the host byte order.

(Q1) Discuss how Android uses processes to control and track applications.

Each application has its own user ID and will run in the context of a separate process. If it is a Java application, the process will also host a Delvik virtual machine as the Java runtime for the Android application. This is important to ensure that Android can separate control the use of system resources and services for each application.

It is possible to enable multiple applications to share the user ID. However whenever this happens, these applications will run in the context of the same process and cannot be individually tracked and controlled by Android.

(Q2) Explain how to use WebView to display an image retrievable from the Web with a given URL (e.g. https://upload.wikimedia.org/wikipedia/commons/6/66/Android_robot.png).

The answer may follow the example given in http://www.tutorialspoint.com/android/android_webview_layout.htm.

Particularly, the answer should include three key parts:

Part 1: add **<WebView>** element to your xml layout file.

Part 2: In order to use it, you have to get a reference of this view in Java file. To get a reference, create an object of the class WebView. Its syntax is:

WebView browser = (WebView) findViewById(R.id.webview);

Part 3: In order to load a web url into the WebView, you need to call a method **loadUrl(String url)** of the WebView class, specifying the required url. Its syntax is:

browser.loadUrl("http://www.tutorialspoint.com");

(Q3) Discuss, with the help of some sample code, how to use AsyncTask to perform a task on a Background Thread.

Students must make it clear that the doInBackground() method is to be used to perform a task in the background. They should also mention that, to start the execution of a background task, we have to create a task instance (object) and call its execute() method.

```
public class PhotoGalleryFragment extends Fragment {

    private static final String TAG = "PhotoGalleryFragment";

    private RecyclerView mPhotoRecyclerView;

    ...

    private class FetchItemsTask extends AsyncTask<Void,Void,Void> {
        @Override
        protected Void doInBackground(Void... params) {
            try {
                String result = new FlickrFetchr()
                        .getUrlString("https://www.bignerdranch.com");
                Log.i(TAG, "Fetched contents of URL: " + result);
            } catch (IOException ioe) {
                Log.e(TAG, "Failed to fetch URL: ", ioe);
            }
            return null;
        }
    }
}
```

```
public class PhotoGalleryFragment extends Fragment {

    private static final String TAG = "PhotoGalleryFragment";

    private RecyclerView mPhotoRecyclerView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRetainInstance(true);
        new FetchItemsTask().execute();
    }

    ...
}
```

(Q4) A single video source transmits 30 frames per second, each containing 2Mbits of data. Suppose that the average communication delay between the video source and the destination is 10s. In practice, the shortest possible communication delay can be 7s and the longest possible communication delay can be 15s. What would be the minimum size for the delay buffer at the destination to completely eliminate the impact of jitter on continued play of the video content?

We have to consider the worst case, after receiving a frame, the next frame can be received in 15-7=8s, because of the jitter.

So we need to keep 8s of video in the buffer. That means the size of the buffer should be 30*2Mbit*8=480Mbit=60MB

(Q5) Both the HTTP and the FTP protocols can be used to transfer files in the Internet. Compare the two protocols by showing the advantages and disadvantages of using each for file transferring.

Answer to this question is not unique. Below are some of the differences. As long as students can identify FOUR correct points, we can consider the answer to be correct.

1. HTTP is used to view websites while FTP is used to access and transfer files. FTP's file transfer purpose is more or less for website maintenance and batch uploads, while HTTP is for client-end work and for end users to upload things such as movies, pictures and other files to the server.

2. HTTP and FTP clients: The common HTTP client is the browser while FTP can be accessed via the command line or a graphical client of its own.

3. HTTP Headers: HTTP Headers contains metadata such as last modified date, character encoding, server name and version and more which is absent in FTP.

4. Age Difference: FTP is about 10 years older than HTTP.

5. Data Formats: FTP can send data both in ASCII and Binary Format but HTTP only uses Binary Format.

6. Pipelining in HTTP: HTTP supports pipelining. It means that a client can ask for the next transfer already before the previous one has ended, which thus allows multiple documents to get sent without a round-trip delay between the documents, but this pipelining is missing in FTP.

7. Dynamic Port Numbers in HTTP: One of the biggest hurdles about FTP in real life is its use of two connections. It uses a first primary connection to send control commands on, and when it sends or receives data, it opens a second TCP stream for that purpose. HTTP uses dynamic port numbers and can go in either direction,

8. Persistent Connection in HTTP: For HTTP communication, a client can maintain a single connection to a server and just keep using that for any amount of transfers. FTP must create a new one for each new data transfer. Repeatedly making new connections are bad for performance due to having to do new handshakes/connections all the time.

9. Compression Algorithms in HTTP: HTTP provides a way for the client and server to negotiate and choose among several compression algorithms. The gzip algorithm being the perhaps most compact one but such kind of sophisticated algorithms are not present in FTP.

10. Support for Proxies in HTTP: One of the biggest selling points for HTTP over FTP is its support for proxies, already built-in into the protocol.

11. One area in which FTP stands out somewhat is that it is a protocol that is directly on file level. It means that FTP has for example commands for listing dir contents of the remote server, while HTTP has no such concept.