Problem 1

Consider the following string of ASCII characters that were captured by *Wireshark* when the browser sent an HTTP GET message (i.e., this is the actual content of an HTTP GET message). The characters <CR><LF> are carriage-return and line-feed characters. Answer the following questions, indicating where in the HTTP GET message below you find the answer.

```
GET /classes/spring17/cs118/project-1.html HTTP/1.1<CR><LF>
Host: web.cs.ucla.edu<CR><LF>
Connection: keep-alive<CR><LF>
Upgrade-Insecure-Requests: 1<CR><LF>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/56.0.2924.87 Safari/537.36<CR><LF>
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8<CR><LF>
Referer: http://web.cs.ucla.edu/classes/spring17/cs118/homeworks.html<CR><LF>
Accept-Encoding: gzip, deflate, sdch<CR><LF>
Accept-Language: en-US,en;q=0.8,lv;q=0.6,ru;q=0.4<CR><LF>
If-None-Match: "5a17-54c4847c4f640-gzip"<CR><LF>
If-Modified-Since: Mon, 03 Apr 2017 19:36:49 GMT<CR><LF>
```

1. What is the **full** URL of the document requested by the browser?
2. What version of HTTP is the browser running?
3. What type of browser initiates this message? Why is the browser type needed in an HTTP request message?
4. Can you find the IP Address of the host on which the browser is running from the captured HTTP request?

1. http://web.cs.ucla.edu/classes/spring17/cs118/project-1
First two lines of the message

2. HTTP 1.1
First line

3. Chrome and Safari from Mac OS X 10.12.3
Got from User-Agent information
This browser type information is needed to prevent malicious users or hackers from getting information.

4. No. There is no IP address information in the host header.

Problem 2

For each of the questions below, describe answer in terms of low-level packet sequences, drops, or network-level packet reordering.

1. A specific case where HTTP/1.1 wins in performance compared to HTTP/1.0
2. A specific case where HTTP with web caching wins in performance compared to HTTP without caching

1. Request to download a file with many images. In this case, HTTP/1.0 would close the protocol when an image is transmitted each time, and reestablish the connection again for another image, so that the server won't mess up with the order of the images. HTTP/1.1 allows persistent connection, which means the connection would not need to be closed and open again for several times. So in this case HTTP/1.1 wins the performance in terms of latency.

2. A service when users often request the same files, such as Netflix. In this case, the HTTP with caching would have the popular materials downloaded, so when the same materials are requested again, the transmission would be very fast. However, HTTP without caching would require the files to be downloaded each time they are requested, even with repetitive files. So in this case when people are having same requests, HTTP with caching wins in performance.

Problem 3

Suppose within your Web browser you click on a link to obtain a Web page. The IP address for the associated URL is cached in your local host, so a DNS look-up is not needed. Suppose that the Web page associated with the link is a small HTML file, consisting only of references to 100 very small objects on the same server. Let RTT_0 denote the RTT between the local host and the server containing the object. How much time elapses (in terms of RTT_0) from when you click on the link until your host receives all of the objects, if you are using:

1. HTTP/1.0 without parallel TCP connections?
2. HTTP/1.0 with parallel TCP connections?
3. HTTP/1.1 without parallel connections, but with pipelining?

Ignore any processing, transmission, or queuing delays in your calculation.

1.
transfer time = 2 * 100 = 200 RTT0
total time = transfer time + 2 RTT0 == 202 RTT0

2.
assume use 100 parallel connections
transfer time = 2 * 1 = 2 RTT0
total time = transfer time + 2 RTT0 = 4 RTT0

3.
transfer time = RTT0
total time = transfer time + 2 RTT0 = 3RTT0

Problem 4

BitTorrent is a communication protocol for peer-to-peer file sharing which is used to distribute data (or files) over the Internet.

1. Consider a new peer A that joins BitTorrent swarm without possessing any chunks. Since peer A has nothing to upload, peer A cannot become a top uploader for any of the other peers. How then will peer A get the first chunk?
2. Explain why BitTorrent is primarily useful for popular files but not for unpopular files.
3. Consider two DHTs (Distributed Hash Table) with a mesh overlay topology and a circular overlay topology, respectively. What are the advantages and disadvantages of each design?

1. In peer to peer transmission, the up-loaders pick a peer in the swarm randomly, so eventually A would be picked.

2. When there are many users, the users would be contributing to the downloading of files, so BitTorrent is useful. However, when there are few users, the users can't contribute much (since there are not many of them), the server still has to do a lot of work.

3.
mesh overlay:
advantage:
accept multiple delivery paths
based on adjacent key
disadvantage:
structure is complex, difficult to develop
required for multiple delivery paths

circular:
advantage:
contains only two peers: the predecessor and the successor.
disadvantage:
requires key and O(N) hopes
reduced amount of messages

Problem 5

The server tries to distribute a file of $F = 15Gbits$ to N clients (peers). The server has an upload rate of $u_s = 30Mbps$, and each peer has a download rate of $d_p = 2Mbps$ and upload rate of $u_p = 1Mbps$. How long does it take to distribute if there are 1,000 peers for both **client-server distribution** and **P2P distribution**.

Client-Server:
Td = max {1000 * 15 Gbits/ 30Mbps, 15Gbits / 2Mbps}
= max { 15 * 10^{12}/3 * 10^7 s, $15 * 10^9$ / $2 * 10^6$ s}
= max {500000s, 7500s}
= 500000 s

P2P:
Td = max { 15 Gbits/ 30Mbps, 15Gbits / 2Mbps , 1000 * 15Gbits / (1Mbps * 1000 + 30Mbps)}
= max {500s, 7500s, 15 * 10^{12}/1030 * 10^6 s}
= max {500s, 7500s, 14563s}
= 14563 s

Using P2P distribution, it takes 14536s to distribute.