Part 1

Request:

```
Hypertext Transfer Protocol
    GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1\r\n]
        Request Method: GET
        Request URI: /wireshark-labs/HTTP-wireshark-file1.html
        Request Version: HTTP/1.1
    Host: gaia.cs.umass.edu\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/63.0.3239.132 Safari/537.36\r\n
    Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8\r\n
    Referer: http://www.ida.liu.se/~TDTS04/labs/2016/Wireshark_HTTP/default.html\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7\r\n
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html]
    [HTTP request 1/1]
    [Response in frame: 196]
```

Answer:

```
Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
        [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
        Request Version: HTTP/1.1
        Status Code: 200
        [Status Code Description: OK]
        Response Phrase: OK
    Date: Thu, 18 Jan 2018 19:55:41 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16 mod_perl/2.0.10
Perl/v5.16.3\r\n
    Last-Modified: Thu, 18 Jan 2018 06:59:01 GMT\r\n
    ETag: "80-563077fb0f7f4"\r\n
    Accept-Ranges: bytes\r\n
    Content-Length: 128\r\n
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html; charset=UTF-8\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.147269000 seconds]
    [Request in frame: 194]
    File Data: 128 bytes
Line-based text data: text/html
```

Task A:

1- Our browser and the server are both running HTTP 1.0. These can be seen in the first line of the HTTP in the GET request for the browser and in the first line of the answer from the server.

2-fr-FR for French from France and en-US for English from the United States. We are also provided with the type of media accepted, the encoding supported, The page that referred the link and finally some information on the browser version as well as it's plug ins'. As for the user, we are provided with some hardware specification such as the CPU architecture and the operating system running.

Request:

```
Internet Protocol Version 4, Src: 192.168.0.103, Dst: 128.119.245.12
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 565
    Identification: 0x5fa7 (24487)
    Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x6288 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.0.103
    Destination: 128.119.245.12
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
```

3- We see our IP adress (192.168.0.103) and the server's (128.119.245.12) in the Internet Protocol.

4- The status code returned is 200 OK, which means the request was valid and is answered to.

5- Last-Modified: Thu, 18 Jan 2018 06:59:01 GMT

6- File Data: 128 bytes. We can see that 128 bytes of content were sent by the server, but the whole answer is longer since other information were received, such as the TCP information which takes 20 bytes for it's header.

7- No

Part 2

First Request of the page:

```
Hypertext Transfer Protocol
    GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n]
        Request Method: GET
        Request URI: /wireshark-labs/HTTP-wireshark-file2.html
        Request Version: HTTP/1.1
    Accept: text/html, application/xhtml+xml, image/jxr, */*\r\n
    Accept-Language: fr-FR,fr-CA;q=0.8,fr;q=0.6,en-CA;q=0.4,en;q=0.2\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/58.0.3029.110 Safari/537.36 Edge/16.16299\r\n
    Accept-Encoding: gzip, deflate\r\n
    Host: gaia.cs.umass.edu\r\n
    Connection: Keep-Alive\r\n
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]
    [HTTP request 1/2]
    [Response in frame: 314]
    [Next request in frame: 332]
```

8- No, there is nothing about modification since in the first request.

Answer to the first request:

```
Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
        [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
        Request Version: HTTP/1.1
        Status Code: 200
        [Status Code Description: OK]
        Response Phrase: OK
    Date: Thu, 18 Jan 2018 20:31:48 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16 mod_perl/2.0.10
Perl/v5.16.3\r\n
    Last-Modified: Thu, 18 Jan 2018 06:59:01 GMT\r\n
    ETag: "173-563077fb0f024"\r\n
    Accept-Ranges: bytes\r\n
    Content-Length: 371\r\n
        [Content length: 371]
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html; charset=UTF-8\r\n
    \r\n
    File Data: 371 bytes
Line-based text data: text/html
    \n
    <html>\n
    \n
    Congratulations again!  Now you've downloaded the file lab2-2.html. <br>\n
    This file's last modification date will not change.  <p>\n
    Thus  if you download this multiple times on your browser, a complete copy <br>\n
    will only be sent once by the server due to the inclusion of the IN-MODIFIED-SINCE<br>\n
    field in your browser's HTTP GET request to the server.\n
    \n
    </html>\n
```

9- Yes, we can see, first of all, the length of the content, which is 371 bytes, and then further in the response, we have the actual content, which is turned into text by Wireshark.

Second request of the Page:

```
Hypertext Transfer Protocol
    GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n]
        Rtion in the HTTP GET and response messages, answer the following (7) questions. When
answering the following questions, you should print out the GET anequest Method: GET
        Request URI: /wireshark-labs/HTTP-wireshark-file2.html
        Request Version: HTTP/1.1
    Accept: text/html, application/xhtml+xml, image/jxr, */*\r\n
    Accept-Language: fr-FR,fr-CA;q=0.8,fr;q=0.6,en-CA;q=0.4,en;q=0.2\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/58.0.3029.110 Safari/537.36 Edge/16.16299\r\n
    Accept-Encoding: gzip, deflate\r\n
    Host: gaia.cs.umass.edu\r\n
    If-Modified-Since: Thu, 18 Jan 2018 06:59:01 GMT\r\n
    If-None-Match: "173-563077fb0f024"\r\n
    Connection: Keep-Alive\r\n
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]
    [HTTP request 2/2]
    [Prev request in frame: 289]
    [Response in frame: 333]
```

10- Yes, we can see it followed by the date and time that we first got the content of the page.

Answer to the second request:

```
Hypertext Transfer Protocol
    HTTP/1.1 304 Not Modified\r\n
        [Expert Info (Chat/Sequence): HTTP/1.1 304 Not Modified\r\n]
        Request Version: HTTP/1.1
        Status Code: 304
        [Status Code Description: Not Modified]
        Response Phrase: Not Modified
    Date: Thu, 18 Jan 2018 20:31:50 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16 mod_perl/2.0.10
Perl/v5.16.3\r\n
    Connection: Keep-Alive\r\n
    Keep-Alive: timeout=5, max=99\r\n
    ETag: "173-563077fb0f024"\r\n
    \r\n
    [HTTP response 2/2]
    [Time since request: 0.128132000 seconds]
    [Prev request in frame: 289]
    [Prev response in frame: 314]
    [Request in frame: 332]
```

11- The server status code response was 304 Not Modified, which means that the content that we currently have in cache is still valid, and thus, the answer does not contain explicitly the content since our version is up to date.

Task B: In this experiment, we can see the behaviour of multiple requests of the same page or content. Depending on the answer from the server to the initial request, the content can be cached. If it is, a new request of the page will ask if it was modified since the initial request. If it hasn't, the server will answer with its 304 Not Modified, allowing the user client to know that the content in the cache is still valid. This type of measure allows more efficient control of the server as it doesn't have to send content for no reason.

Part 3

12- It took only 1 request from the user.

13- It took 4 data containing TCP segments to fully carry the response. First, there are 3 normal TCP packets with 1460 bytes each, with one additional HTTP packet containing 436 bytes of the TCP segment.

14- 200 OK

15- No, since the TCP works in the transport layer, and thus, it does not have to go to the application layer where the HTTP protocol operates.

Task C: What we could observe in the experiment, is that for a single request from the user, the server can answer in multiple segments. It first acknowledges the connection, and then it starts sending packets TCP packets in sequences. For each sequences, the receiving end of the user acknowledges that it has correctly received the entire sequence. Once it is acknowledged, it continues with the following sequences following the same steps until it is done. Finally, once every packet was sent, there is a final HTTP answer to indicate that everything was sent using the push flag. The push flag tells the TCP layer of the user that the data can be pushed to the application layer. The last step is for the user's TCP layer to acknowledge the last packet and answer from the server.

Part 4

16- We see a total of 3 HTTP GET requests. One for the original web page, and two more for each of the embedded pictures in the page. The request destination are all different. The first once is to the server hosting the website, the two others are to the website hosting the pictures. This means that the website does not have the picture saved locally, and is rather referencing them in his HTML.

17- Both request were sent in parallel, it is possible to see through the port numbers of each request. Indeed, the user opened multiple ports so that it could send and receive them separately, and not have to wait for one request to be completed.

Task D: In this experiment, we observed how the user side reacts when it notices that the web page is not fully loaded as there are external data to download. In order to make the loading faster, it opens multiple ports for each request that it will have to make for each data content missing. However, it is impossible to carry this through only one HTTP request, it had to make a request of every piece of data content.

Part 5

18- 401 Authorization Required

19- it added : Authorization: Basic ZXRoLXN0dWRlbnRzOm5ldHdvcmtz

20- Connection: Keep Alive allows multiple requests to use the same connection without having the do the handshake with the server again. It reduces latency, and the number of connection that have to be opened, which is more efficient. Connection: Close means that the connection will be closed after the response from the server. Connection: Keep alive is usually better to use in most cases, Close is used if the server does not support persistent connection. It can also tell the server to explicitly close the connection instead of waiting for the time-out in order to be more efficient.

Task E: By knowing the uses of Keep-alive, which is for persistent connections, and close, for unsupported servers or explicit end of connection, we will be able to manage connections efficiently in the second assignment. This will allow us to make sure our work does not consume connection for no reason, and that it doesn't have to do the handshake with the server more times than it needs.