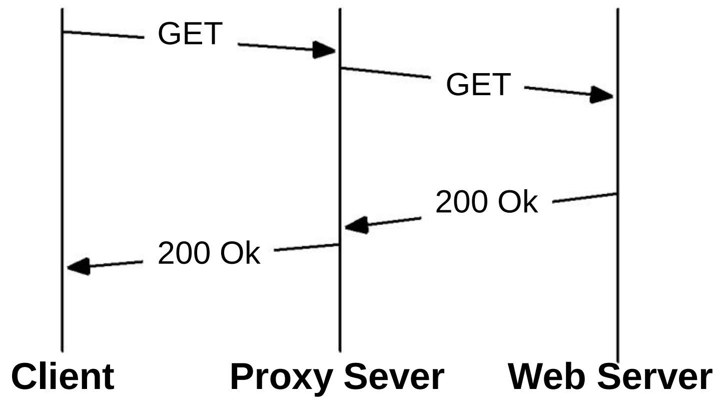
Book Questions

1. Problem 5.53 - 10 points
2. Problem 5.54 - 10 points
3. Problem 5.65 - 10 points
4. Problem 6.4 - 10 points
5. Problem 6.8 - 10 points
6. Problem 6.13 - 10 points
7. Problem 6.15 - 10 points
8. Problem 6.16 - 10 points

Programming Assignment - 100 points

Simple HTTP Proxy



In this assignment you will implement a simple HTTP proxy server and HTTP command line client. This implementation will work for HTTP/1.0, which is specified as RFC 1945. Your proxy will start on a user specified port and listen for incoming HTTP requests. When it receives a request it will first check its cached data in attempt to serve the request; however, if there are no valid entries the request will be proxied to the intended destination, and the results will be cached for later use. Your proxy cache should maintain at least 10 document entries in the cache. Entries should be replaced in a Least Recently Used (LRU) fashion; and entries should not be used if their timestamp indicates they are stale. Your server should provide useful logging messages to stdout during its operation.

Your client will take input from the command line as to the web proxy address and port as well as the url to retrieve. It will then display the resulting document retrieved, or error message, to stdout.

Keep in mind HTTP/1.0 uses TCP as its transport protocol.

Usage Syntax:

```
./proxy <ip to bind> <port to bind>
./client <proxy address> <proxy port> <url to retrieve>
```

Bonus - 50 points

Even though a document might be expired in the cache, its content might not have been updated at the source. There are several ways for the proxy to determine this during a subsequent request and either save message bandwidth or time if the the document has not changed. Implement either a conditional 'GET' or the 'HEAD' method to determine if the document is new. If it is not, then update your cache and serve the existing data. If it is new, then provide the new document back to the client.

Notes

1. Just as in the previous assignment it will be helpful for you to break the project up into parts and implement and test each independently.
2. You will need to use the same select loop architecture as in the previous project to handle concurrent client access.
3. Keep in mind HTTP/1.0 uses TCP as its transport protocol.
4. The client only needs to generate a 'GET" message.
5. You will need to learn how to parse timestamps and provide comparisons in order to manage your cache properly.

Submission Guidelines

1. Homework must be turned in at the beginning of class on the date that it is due.
2. All programming assignments must include: makefile, README, and the code.
3. The README should contain a description of your code: architecture, usage, and errata, etc.
4. Make sure all binaries and object code have been cleaned from your project before you submit.
5. Your project must compile on a standard linux development system. Your code will be graded on a linux testbed.