CMPE 156/L, Winter 2019
Programming Assignment 5
Due: _____

The goal of this exercise is to develop a proxy HTTP server that can process HTTP requests generated by a Web browser and filter requests based on an access control list.

General Features

- The proxy server needs to support HTTP 1.0 and 1.1. It needs to handle only the GET and HEAD requests.
- Add the Forwarded header. The format of this header is "Forwarded: for=<client-ip>; proto=http; by=<proxy-ip>", where client-ip and proxy-ip are the addresses of the client and the proxy respectively (see RFC 7239).
- A standard browser client (e.g., Firefox) will be used to generate the HTTP requests. You need to configure the browser to direct its requests to the proxy server by setting up the browser with the IP address and listening port number associated with the proxy server.
- Only one copy of the browser will be run at a time, but the server must be designed to support concurrent requests from the browser.

Requirements

The proxy server needs to be concurrent using threads. It also should have the capability to parse the headers of incoming HTTP requests and forward them to their ultimate destinations if allowed by the access control mechanism. It should then forward any data received from the destination back to the browser.

The list of forbidden sites is maintained in a file *forbidden-sites* file. An example looks like:

```
www.cnn.com
www.youtube.com
```

The proxy is started by: ./proxy <listen-port> <forbidden-sites-file>

The proxy listens for requests on <listen-port>. The proxy server must return an HTTP error response 403 (Forbidden URL) to the browser if the access is to a site in the *forbidden-sites* file. If the proxy server received an HTTP request method other than GET or HEAD, it should return an error code of 405 (Method not allowed) or 501 (Not implemented). The proxy server must also respond to any errors in the request (for example HTTP header fields missing or inconsistent) with the appropriate responses (for example, 400 Bad Request).

The proxy server must always remain open and accept requests from the browser.

The proxy server does not need to support *https*. It also does not need to support the chunked transfer mode of HTTP 1.1.

Logging: The proxy server must maintain a log file, named access.log, to track all requests received. It must print one line for each request with the following information:

- Date and time the request was received
- Type of request and HTTP version
- Requesting (client) host name or IP address
- URI and server address
- Action taken by proxy (forwarded, filtered, etc.)
- Type of errors detected, if any

The log format has to be:

<date-format> <client-ip> <request-first-line> <http-status-code> <object-size-in-byte>

For example,

2017-02-27T20:19:44.852Z 127.0.0.1 "GET /apache_pb.gif HTTP/1.1" 200 2326

Persistent connections: Note that the browser will use persistent connections by default if it is using HTTP 1.1. In this case, either the client (browser) or the server may initiate the close. The proxy server must be able to deal with this by closing the connection on the other side when the TCP connection on one side closes.

What to submit?

You must submit all the files in a single compressed tar file (with tar.gz extension). The files should include

1. All source code necessary to build and run the client and server.
2. A README file including your name and a list of files in the submission with a brief description of each. If your code does not work completely, explain what works and what doesn't or has not been tested.
3. A Makefile that can be used to build the client and server binaries.
4. Documentation of your design in plain text or pdf. Do not include any Microsoft Word files. The documentation should describe the internal design of your client and server implementations.
5. Including in the design document, a specification of the application-layer protocol, describing the handshakes involved, message formats, error handling, etc.
6. Organize the files into directories (src, bin, doc, etc.)

Grading

Each submission will be tested to make sure it works properly and can deal with errors. Grades are allocated using the following guidelines:

Basic functionality	20%
Basic testing	20%
Dealing with errors	30%
Documentation	15%
Style/Code structure, etc.	15%

The files must be submitted before midnight on the due date.

Honor Code
All the code must be developed independently. All the work submitted must be your own.

Debugging Utilities

It's helpful to use wget or curl commands to download files from a remote site. These commands can take an environment variable to use a proxy. Also, for a given URL, the optional argument can be used to go through a proxy.

If the proxy is listening on the localhost port 9090:
> curl -v –proxy 127.0.0.1:9090 www.example.com