Cornelius Peck
ID: 11780145

I ended up implementing most of the methods used in csapp.c file myself, as I had persistent import errors that interfered with my ability to debug the proxy as it was running when I tried to use functions in csapp.c.

I begin my proxy program by registering a signal handler that allows me to cleanly break out of the proxy when needed instead of force killing the process. I then check that explicitly 2 commandline arguments were entered and convert the port argument to an integer. After this I declare my variables for storing the proxy and client sockets and addresses, then create the proxy socket, initialize the proxy address, and bind them together. Next I set the proxy up to listen to LISTENQ requests, and begin my proxy running while loop. From here I initialize the client address, accept the connection to the proxy socket, from the client socket, and begin my proxy function for the connection. Within the proxy function, I ensure that my buffer space is iniitialized properly, then read the request from the client using buffered rio input, and check that the request was a GET request. If it was any other type it is rejected and the proxy moves on to the next connection. Once I know it's a GET request, I extract the hostname, first looking for the Host: header then checking for an http then a https. Next is checking if the hosts name has an explicit port to be used, then scanning the filepath of the request. I also check if the filepath is empty and set it to just a '/' if one is not present already. From there I check for additional headers beyond Host, User-Agent, Connection, and Proxy-Connection, and copy them into a list of all the additional headers. Next comes connecting to the server, starting with setting up the server socket and address, then connecting to the socket. After that I construct the new modified request to be sent to the server with the host, filepath, HTTP type of 1.0 and closed connection and proxy-connection headers, as well as any additional headers from the original request. Next I send this request to the server with rio_writen, and receive the server's response using buffered rio input again. In this step I check if the server response is a 301 or a 302 before sending it back to the client, and if so I modify the request and redirect it to the new location, as well as checking if the https port is required. If neither a 301 or 302 is recieved, I return the server's response to the client again using rio_writen, and close the server and client socket. Next, if the ^C signal has been sent, the loop exits and the proxy socket is closed, otherwise the loop continues to the next connection.

Some urls like google or bing can't be accessed with the proxy because it's sending and receiving unencrypted information since everything is over http instead of https. For instance, when trying to connect to eecs.wsu.edu through my proxy, I get a message that I'm trying to communicate with a https only server using plain http requests, so my request was denied. Sites like google and bing will refuse to connect entirely due to the unprotected nature of the http requests.