CSCI-4220 Network Programming - Spring 2016
Homework 3: HTTP Client

Your solution must be in C or C++ using the sockets API as discussed in class.
Please name your file <your_rcs>_hw3.zip (e.g. armena2_hw3.zip). .tar.gz is also OK.
Please include a plain text README file in your submission - 10pts penalty if missing
(markdown or other formats that can be read with "cat README" are also acceptable)

For this assignment, you will be creating a simple HTTP client. Your command line argument will be a URL in the standard form: http://host[:port]/path. (the :port portion, shown in brackets, is optional). To accomplish this, you will need to parse the URL, connect a TCP socket to the correct host and port, send a request with request headers, then read the response headers and body. Note that HTTP separates the headers from the request or response body with a blank line (that is, the string "\r\n\r\n"). Also note that an HTTP client is called a "user agent" by the RFC. Last but not least, **note that HTTP uses "\r\n" (CR LF) for newlines**.

Step 1: URL Parsing

You need to parse the URL into a host, a port, and a path. For example, for http://www.asquaredlabs.com:80/csci4220/, the host is www.asquaredlabs.com, the port is 80, and the path is /csci4220/. If no port is given, use port 80 as this is the standard, well-known port for HTTP.

Step 2: TCP Socket Connection

Use getaddrinfo() to convert your host and port strings to socket addresses. You may use a v6-only socket with v6 and v6-mapped v4 addresses, or you may create the appropriate type of socket for each address returned from getaddrinfo(). If you choose the second option, make sure to close any extra sockets that you don't use. You should attempt to connect() to each address returned until you find one which succeeds.

Step 3: Send Request and Headers

An HTTP request consists of a request line followed by headers - each header item on one line. The request line is of the form "method path version" - for example "GET /csci4220/ HTTP/1.1". The headers can contain additional information about the request or the client. Every HTTP/1.1 request must include a Host header. This header is used to host multiple sites on the same IP address by requiring the client to communicate the hostname part of the URL to the server. It must be in the form "Host: hostname:port" e.g. "Host: www.asquaredlabs.com:80". The port can be omitted if it is 80 e.g. "Host: www.asquaredlabs.com". While HTTP does not require it, you must also include a User-Agent header to identify your program to servers e.g. "User-Agent: armena2-netprog-hw3/1.0". Your program must not assume that the entire request headers can be written in a single write() or send().

Step 4: Read Response

Separate the response into headers and content - remember, these are separated by a blank line. Write the headers to standard error, and the response to standard output. You will need to use multiple calls to read() or recv() to accomplish this. Since we may receive a binary file from the server, you should use write(), fwrite(), or cout.write() to write the data bytes to standard output. The server will typically provide a "Content-Length: x" header where x is the number of bytes that will be contained in the response body. You will need to find and parse this header to determine the number of bytes to read. You may handle the case where the server does not include this header by exiting with an error. Be sure to close the socket after the response has been read.