The Domain Name System (DNS) provides a vital service to the modern Internet, translating human-friendly and easily readable URLs and Addresses to the IP addresses routing protocols used to direct connections and packets. Many protocols such as HTTP, SMTP, and FTP use this the DNS system.

Your Assignment:

Create a simple server servicing requests for address to IP translation (DNS)

- The server should listen on port 9889, use TCP, and be able to handle multiple requests at the same time (eg: you will need to use threading). Servicing a request to completion should not close the server down.
- The server should utilize a basic DNS algorithm you will write:
    1. A Query comes in from a client or application
        a. <addressURL> like www.google.com or www.myserver.org
    2. Check the host's local cache (this can be a file system, database, etc)
        a. If the local cache has the response, send the entry as the response
        b. If not in the local cache, attempt to resolve with a DNS api call (such as *gethostbyname*), if the address is not resolvable return an error message as the response
            i. Note: it is possible for DNS to resolve to multiple addresses, take the first one
            ii. (bonus) if resolving multiple addresses use ping to return the fastest address
        c. Save the response into the local cache (if not already present) to reduce the need to resolve via the DNS api for future queries
            i. Cache Format: <host>,<ipaddress> = www.google.com, 192.1.1.1
    3. Send the response back in the format "Source:<hostname>:<IP address>". For example "Local DNS:google.com: 216.58.216.68".
- The server should output a log file of the requests in a file (as dns-server-log.csv)
    - <host>,<resolved response>
    - An example has been included (logfileexample.csv)

What To Submit:

- A zip file (.tar.xz or .zip formats), named as <firstAndLastName>-project1.zip (or .tar.xz) (example: AliceBoberson-project1.tar.xz), to the class canvas site containing:
    - Your server source code
        - (you can use any language you wish, but it must run, compile, and execute on a CSE-Lab machine)
            - If using a compiled language (like c, c++) you must include a compiled executable of your code and any make files you used
    - A logfile from your server showing the queries and their resolutions (dns-server-log.csv)

        - Formatted as each line <URL requested>,<resolved answer>
        - Example: www.google.com,192.1.1.1
        - Example: www.nothere.no,"Host not found"
    - A readme file (as README.md using Markdown syntax) with sections:
        - A header
            - <Your name>,Zhang CSCI4211,<current date (day/month/year)>
            - Ex: Marky Markdowns,Zhang CSCI4211,31/12/1969
        - Compilation section
            - Step by step instructions on how to compile and setup your code
            - Ex:
                1. Run gcc -o test test.c in command line
                2. Copy input.txt to dir/
                3. Etc…
        - Execution/Running section
            - Step by step instruction on how to run/execute your program
            - Ex:
                1. Run myprogram.exe
                2. Etc…
        - Description section
            - Describe overall what your program does and how logical it operates, be detailed
            - Ex: This program does x with y and creates z ...

To help with the assignment:

- Some client code has been written that you can use to test your server and included with the assignment zip
    - How to use the client code:
        - For each DNS request, client opens a socket to server to query its desired host names as needed,providing host name for the query.
        - The format of the query will have the host name "www.yahoo.com" or "google.com" (of course for stdin input, you need to follow that with a carriage return).
        - Holding the socket open, the client will wait for the server to respond with a corresponding IPaddress (or an error message, or "Host Not Found").
        - Entering a 'q' or 'Q' from input, terminates the clients program
            - All the sockets will be closed
            - Server will remain online
- Skeleton server code has been written to help you structure your server code and included with the assignment zip