Exercise 3: Digging into DNS (marked, include in the lab report, 5 Marks)

To answer the following questions, you will make DNS queries using some of the query types you have encountered in the above exercise. Some questions may require you to send multiple DNS queries. Before you proceed, read the manpage of dig (type man dig in the terminal). Make sure you understand how you can explicitly specify the following:

- nameserver to query
- type of DNS query to make (the default query types are those you saw in exercise 1)
- performing reverse queries

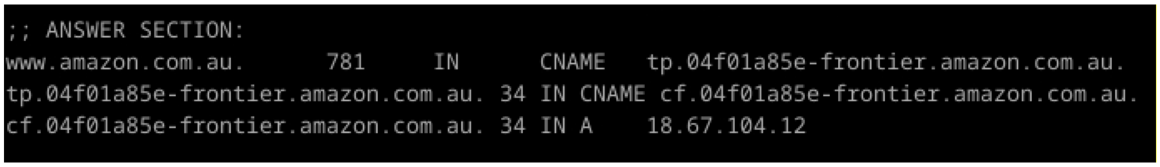
Note: Include the output of all the dig commands you have used in your answers.

To send a query to a particular name server (say x.x.x.x) you should use the following command:

```
dig @x.x.x.x hostname
```

Question 1. What is the IP address of www.amazon.com.au ? What type of DNS query is sent to get this answer?

Answer 1: Use dig command: dig www.amazon.com.au

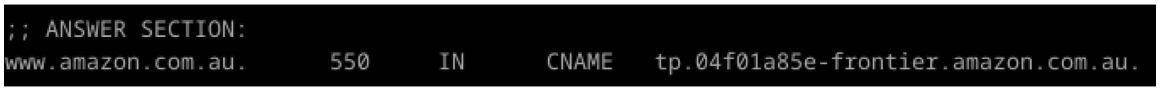


```
;; ANSWER SECTION:
www.amazon.com.au.      781     IN      CNAME   tp.04f01a85e-frontier.amazon.com.au.
tp.04f01a85e-frontier.amazon.com.au. 34 IN CNAME cf.04f01a85e-frontier.amazon.com.au.
cf.04f01a85e-frontier.amazon.com.au. 34 IN A    18.67.104.12
```

The IP address will be listed in the ANSWER SECTION. The query type used is **A** (Address record), which maps a domain name to an IPv4 address, as 18.67.104.12

Question 2. What is the canonical name for the webserver (i.e., www.amazon.com.au)? Suggest a reason for having an alias for this server.

Answer 2: Use command: dig www.amazon.com.au CNAME



```
;; ANSWER SECTION:
www.amazon.com.au.      550     IN      CNAME   tp.04f01a85e-frontier.amazon.com.au.
```

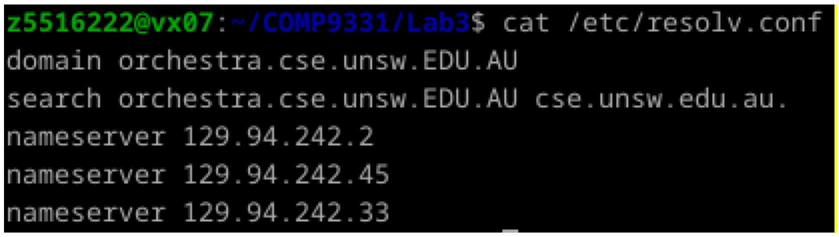
The canonical name (CNAME) will be listed in the ANSWER SECTION: tp.04f01a85e-frontier.amazon.com.au.

Question 3. What can you make of the rest of the response/what is it used for (i.e., the details available in the DNS response (cookies and other fields))?

Answer 3: The DNS response includes fields like: TTL, Flags, Additional Section.

Question 4. What is the IP address of the local nameserver for your machine?

Answer 4: Use command: cat /etc/resolv.conf



```
z5516222@vx07:~/COMP9331/Lab3$ cat /etc/resolv.conf
domain orchestra.cse.unsw.EDU.AU
search orchestra.cse.unsw.EDU.AU cse.unsw.edu.au.
nameserver 129.94.242.2
nameserver 129.94.242.45
nameserver 129.94.242.33
```

nameserver 129.94.242.2

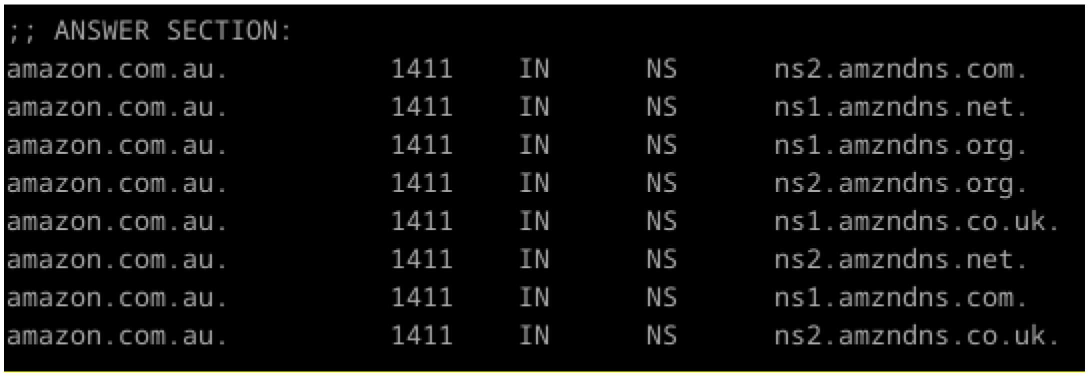
nameserver 129.94.242.45

nameserver 129.94.242.33

Question 5. What are the DNS nameservers for the " amazon.com.au " domain (note: the domain name is amazon.com.au and not www.amazon.com.au . This is an example of what is referred to as the apex/naked domain)? Find their IP addresses. Which DNS query type is used to obtain this information?

Answer 5: To find the nameservers, use: dig amazon.com.au NS

The nameservers will be listed in the ANSWER SECTION.:

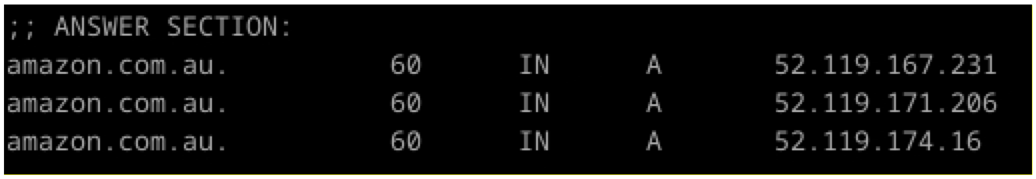


```
;; ANSWER SECTION:
amazon.com.au.          1411    IN      NS      ns2.amzndns.com.
amazon.com.au.          1411    IN      NS      ns1.amzndns.net.
amazon.com.au.          1411    IN      NS      ns1.amzndns.org.
amazon.com.au.          1411    IN      NS      ns2.amzndns.org.
amazon.com.au.          1411    IN      NS      ns1.amzndns.co.uk.
amazon.com.au.          1411    IN      NS      ns2.amzndns.net.
amazon.com.au.          1411    IN      NS      ns1.amzndns.com.
amazon.com.au.          1411    IN      NS      ns2.amzndns.co.uk.
```

To find their IP addresses, query each nameserver: dig @nameserver amazon.com.au

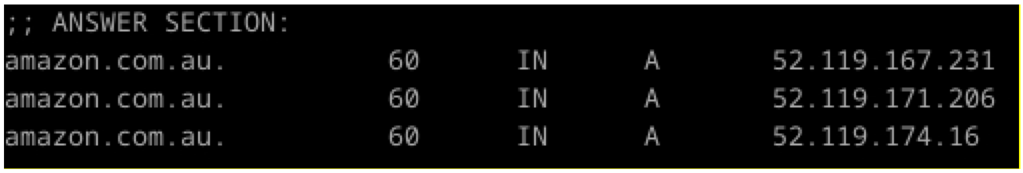
The query type used is **NS**:

ns2.amzndns.com.



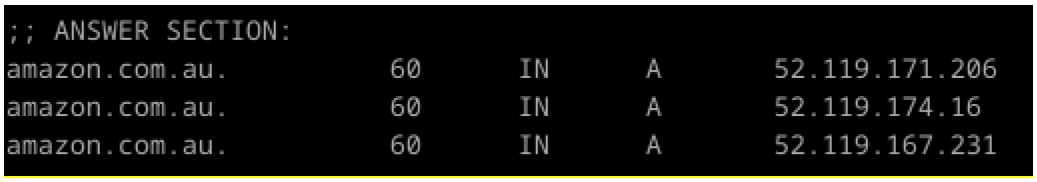
```
;; ANSWER SECTION:
amazon.com.au.          60      IN      A       52.119.167.231
amazon.com.au.          60      IN      A       52.119.171.206
amazon.com.au.          60      IN      A       52.119.174.16
```

ns1.amzndns.net.



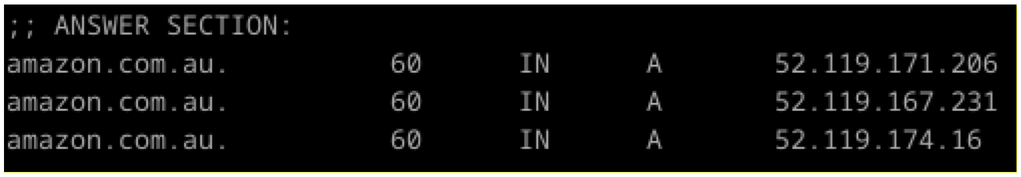
```
;; ANSWER SECTION:
amazon.com.au.          60      IN      A       52.119.167.231
amazon.com.au.          60      IN      A       52.119.171.206
amazon.com.au.          60      IN      A       52.119.174.16
```

ns1.amzndns.org.



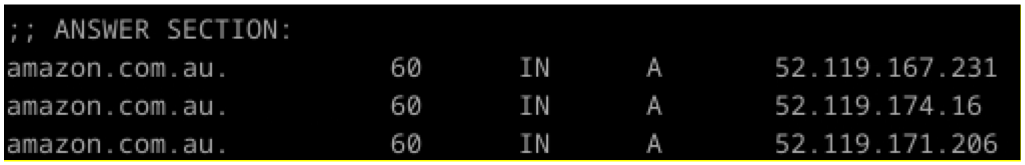
```
;; ANSWER SECTION:
amazon.com.au.          60      IN      A       52.119.171.206
amazon.com.au.          60      IN      A       52.119.174.16
amazon.com.au.          60      IN      A       52.119.167.231
```

ns2.amzndns.org.



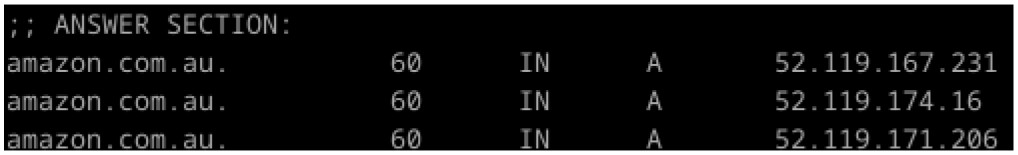
```
;; ANSWER SECTION:
amazon.com.au.          60      IN      A       52.119.171.206
amazon.com.au.          60      IN      A       52.119.167.231
amazon.com.au.          60      IN      A       52.119.174.16
```

ns1.amzndns.co.uk.



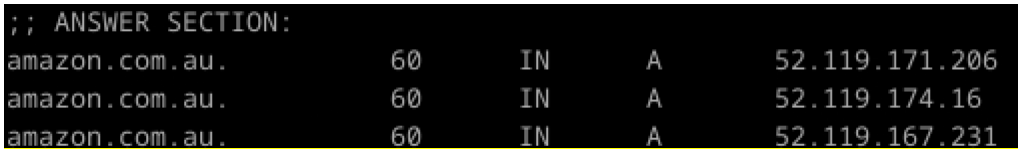
```
;; ANSWER SECTION:
amazon.com.au.          60      IN      A       52.119.167.231
amazon.com.au.          60      IN      A       52.119.174.16
amazon.com.au.          60      IN      A       52.119.171.206
```

ns2.amzndns.net.



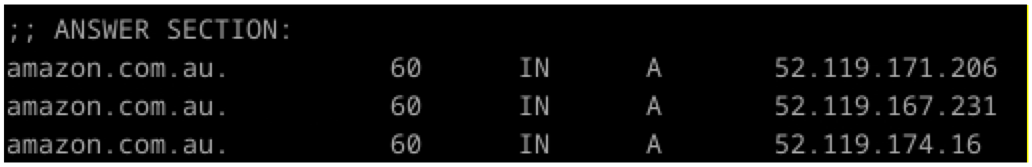
```
;; ANSWER SECTION:
amazon.com.au.          60      IN      A       52.119.167.231
amazon.com.au.          60      IN      A       52.119.174.16
amazon.com.au.          60      IN      A       52.119.171.206
```

ns1.amzndns.com.



```
;; ANSWER SECTION:
amazon.com.au.          60      IN      A       52.119.171.206
amazon.com.au.          60      IN      A       52.119.174.16
amazon.com.au.          60      IN      A       52.119.167.231
```

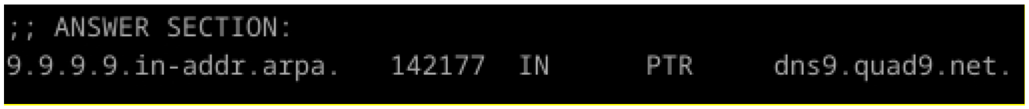
ns2.amzndns.co.uk.



```
;; ANSWER SECTION:
amazon.com.au.          60      IN      A       52.119.171.206
amazon.com.au.          60      IN      A       52.119.167.231
amazon.com.au.          60      IN      A       52.119.174.16
```

Question 6. What is the DNS name associated with the IP address 9.9.9.9? Which DNS query type is used to obtain this information?

Answer 6: To find the DNS name, use: dig -x 9.9.9.9

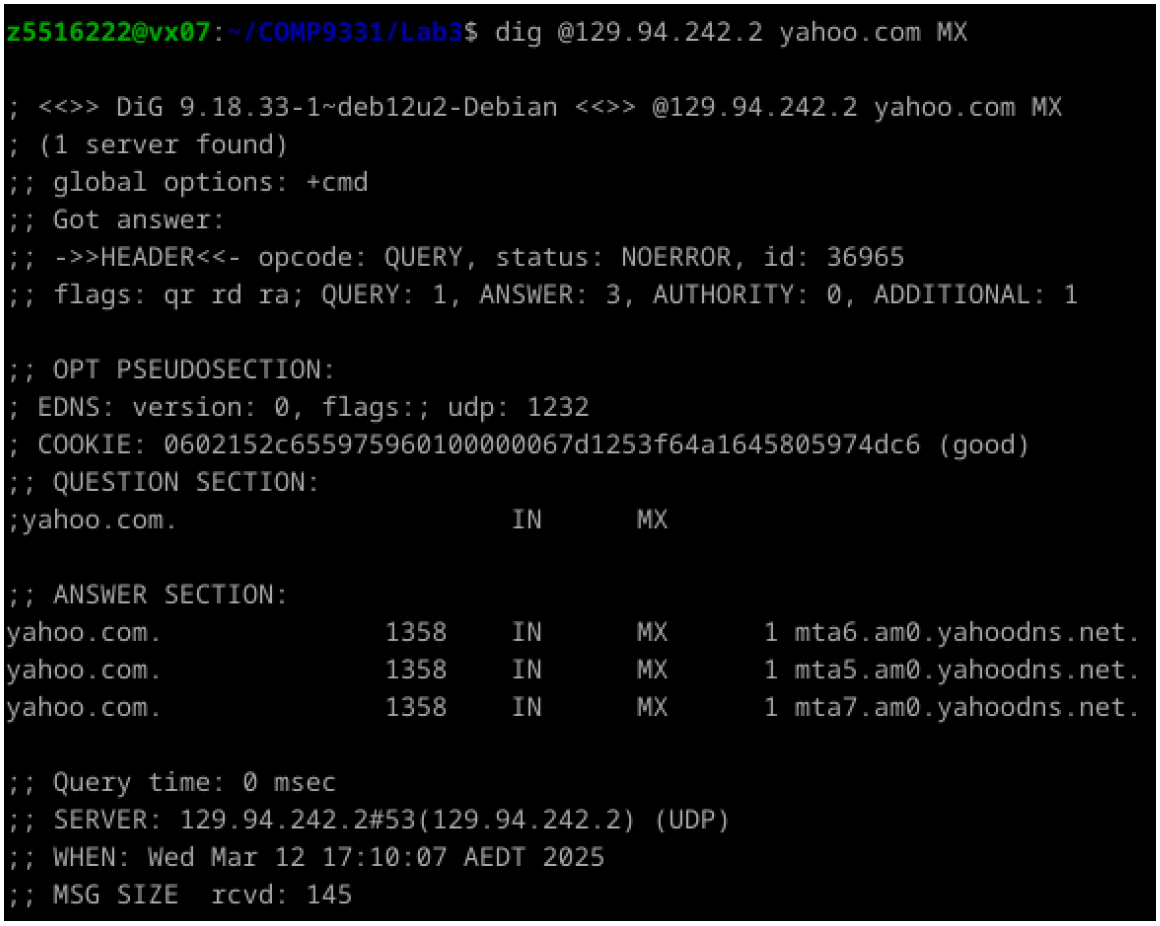


```
;; ANSWER SECTION:
9.9.9.9.in-addr.arpa.   142177  IN      PTR     dns9.quad9.net.
```

The DNS name will be listed in the ANSWER SECTION. The query type used is **PTR** (Pointer record), which maps an IP address to a domain name.

Question 7. Run, dig and query the CSE nameserver (129.94.242.2) for the mail servers for yahoo.com (again, the domain name is yahoo.com, not www.yahoo.com). Did you get an authoritative answer? Why? (HINT: Just because a response contains information in the authoritative part of the DNS response message does not mean it came from an authoritative name server. You should examine the flags in the response message to determine the answer)

Answer 7: Use command: dig @129.94.242.2 yahoo.com MX

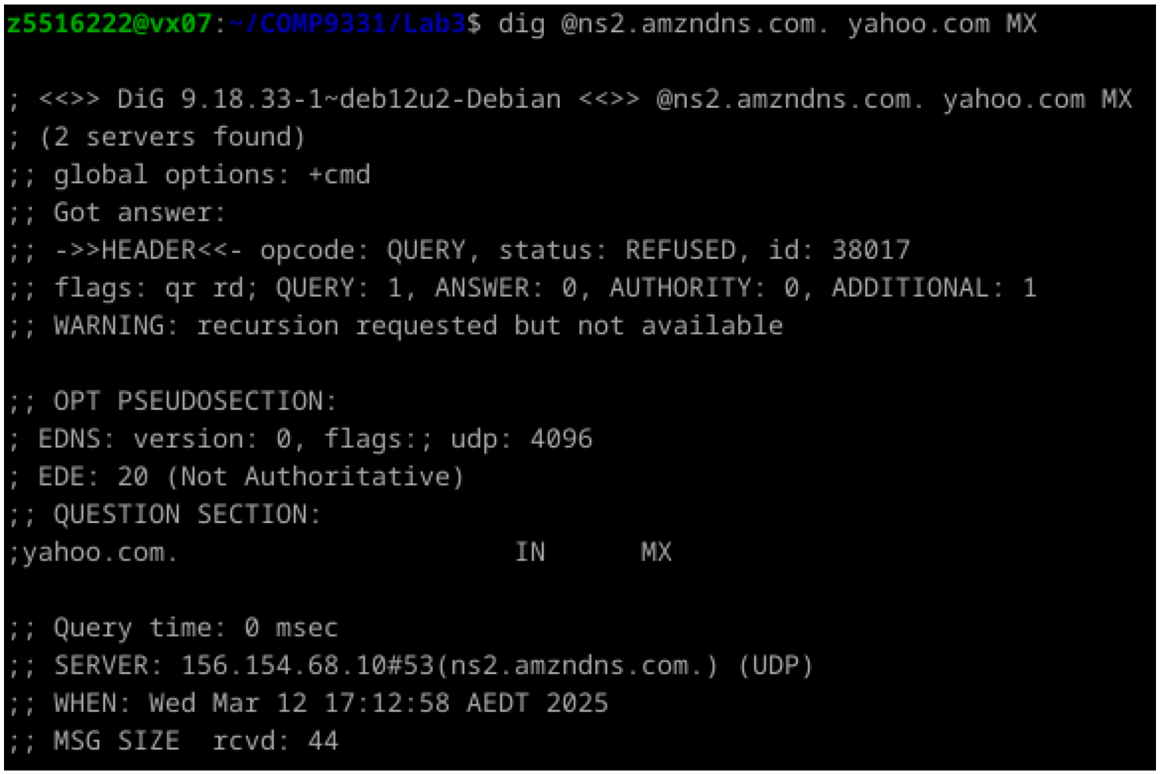


```
z5516222@vx07:~/COMP9331/Lab3$ dig @129.94.242.2 yahoo.com MX

; <<>> DiG 9.18.33-1~deb12u2-Debian <<>> @129.94.242.2 yahoo.com MX
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36965
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 0602152c655975960100000067d1253f64a1645805974dc6 (good)
;; QUESTION SECTION:
;yahoo.com.                     IN      MX

;; ANSWER SECTION:
yahoo.com.              1358    IN      MX      1 mta6.am0.yahoodns.net.
yahoo.com.              1358    IN      MX      1 mta5.am0.yahoodns.net.
yahoo.com.              1358    IN      MX      1 mta7.am0.yahoodns.net.

;; Query time: 0 msec
;; SERVER: 129.94.242.2#53(129.94.242.2) (UDP)
;; WHEN: Wed Mar 12 17:10:07 AEDT 2025
;; MSG SIZE  rcvd: 145
```

The aa (Authoritative Answer) flag is not set, the answer is not authoritative.

Question 8. Repeat the above (i.e. Question 7), but use one of the nameservers obtained in Question 5. What is the result?

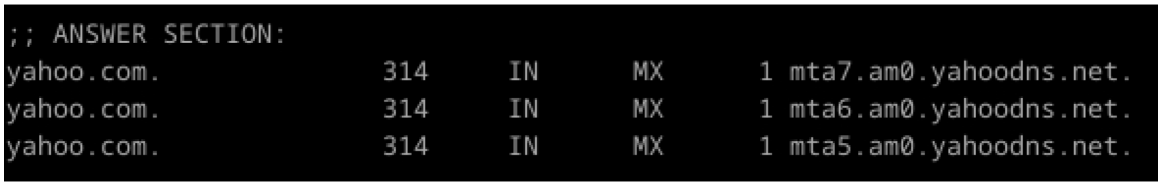
Answer 8: dig @ns2.amzndns.com. yahoo.com MX



```
z5516222@vx07:~/COMP9331/Lab3$ dig @ns2.amzndns.com. yahoo.com MX

; <<>> DiG 9.18.33-1~deb12u2-Debian <<>> @ns2.amzndns.com. yahoo.com MX
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 38017
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; EDE: 20 (Not Authoritative)
;; QUESTION SECTION:
;yahoo.com.                     IN      MX

;; Query time: 0 msec
;; SERVER: 156.154.68.10#53(ns2.amzndns.com.) (UDP)
;; WHEN: Wed Mar 12 17:12:58 AEDT 2025
;; MSG SIZE  rcvd: 44
```

Question 9. Obtain the authoritative answer for the mail servers for yahoo.com. What type of DNS query is sent to obtain this information?

Answer 9: dig @8.8.8.8 yahoo.com MX



```
;; ANSWER SECTION:
yahoo.com.              314     IN      MX      1 mta7.am0.yahoodns.net.
yahoo.com.              314     IN      MX      1 mta6.am0.yahoodns.net.
yahoo.com.              314     IN      MX      1 mta5.am0.yahoodns.net.
```

The query type used is **MX** (Mail Exchange).

Question 10. In this exercise, you simulate the iterative DNS query process to find the IP address of your machine (e.g. lyre00.cse.unsw.edu.au). If you are using VLAB then find the IP address of one of the following: lyre00.cse.unsw.edu.au, lyre01.cse.unsw.edu.au, flute00.cse.unsw.edu.au or flute01.cse.unsw.edu.au. First, find the name server (query type NS) of the "." domain (root domain). Query this nameserver to find the authoritative name server for the "au." domain. Query this second server to find the authoritative nameserver for the "edu.au." domain. Now query this nameserver to find the authoritative nameserver for "unsw.edu.au". Next, query the nameserver of unsw.edu.au to find the authoritative name server of cse.unsw.edu.au. Now, query the nameserver of cse.unsw.edu.au to find your host's IP address. How many DNS servers do you have to query for an authoritative answer?

Answer 10:

1. Query the root nameserver for ..
2. Query the nameserver for au..
3. Query the nameserver for edu.au..

4. Query the nameserver for unsw.edu.au..
5. Query the nameserver for cse.unsw.edu.au..

Question 11. Can one physical machine have several names and/or IP addresses associated with it?

Answer 11: Yes, one physical machine can have multiple names (aliases) and/or IP addresses. This is achieved through:

CNAME records: Multiple domain names pointing to the same machine.

Multiple A records: One domain name mapped to multiple IP addresses for load balancing or redundancy.

Exercise 4: A Simple Web Server (Marked, submit your code, 5 Marks)

Please submit the source code as a separate file. The tutor will run the code to check if the output is as expected.

In this exercise, you will learn the basics of TCP socket programming: how to create a socket, bind it to a specific address and port, and send and receive an HTTP packet. You will also learn some basics of HTTP header format. You will develop a web server that handles one HTTP request at a time. Your server should handle persistent connections (**1 mark**) (i.e. **HTTP 1.1, the default version of HTTP used by all browsers)** . Specifically, your web server should do the following:

(i) Create a connection socket when contacted by a client (browser).

(ii) Receive HTTP requests from this connection. Your server should only process GET requests. You may assume that only GET requests will be received.

(iii) Parse the request to determine the specific file being requested.

(iv) Get the requested file from the server's file system.

(v) Create an HTTP response message consisting of the requested file preceded by header lines.

(vi) Send the response over the TCP connection to the requesting browser.

(vii) If the requested file is not present on the server, the server should send an HTTP “404 Not Found” message back to the client.

(viii) The server should listen in a loop, waiting for the next request from the browser.

(ix) The server should be able to handle HTTP 1.1 persistent connections. This means It should be able to handle multiple requests from the same connection. This will carry 1 mark if you manage to implement this.

You don't have to deal with any other error conditions.

Your program should be called WebServer.c, WebServer.java, or WebServer.py.

You should write the server so that it executes with the following command:

```
$ java WebServer port          # (for Java)
$ ./WebServer port              # (for C)

$ python3 WebServer.py port     # (for Python)
```

where the port is the port number your web server will listen to. Specify a non-standard port No (other than 80 and 8080, and > 1024). We will use this port No in the URL while issuing requests from the web browser. We recommend to choose a port number from the dynamic port range: 49152 to 65535.

1. Place a simple HTML file (index.html, without any hyperlinks) in the same directory as the server program. A sample index.html file is provided here . Run the server program as indicated above. Open a web browser on the same machine. Type the following URL:

```
http://127.0.0.1:port/index.html
```

where the *port* is the port number your server is listening on. Note that if you forget to include the port number, the browser will assume the default port of 80 while we are running our web server on a non-standard port. The browser should display the content of index.html.

2. Place multiple image files (.jpeg) in the same directory as the server program. Run the server program as indicated above. Open a web browser on the same machine. The type

```
http://127.0.0.1:port/myimage.jpeg
```

where the *port* is the port number the server listens to, and *myimage.jpeg* is one of the image files present in the server's directory. The browser should display the image.

3. Now try and request an object that does not exist in the server directory, e.g.:

```
http://127.0.0.1:port/bio.html
```

The browser should display the 404 error message. Note that showing a specific error message depends on the browser. If you are sending the correct 404 error message and the browser is still not displaying it, you may consider sending a custom error message as text/html in the body of your response.

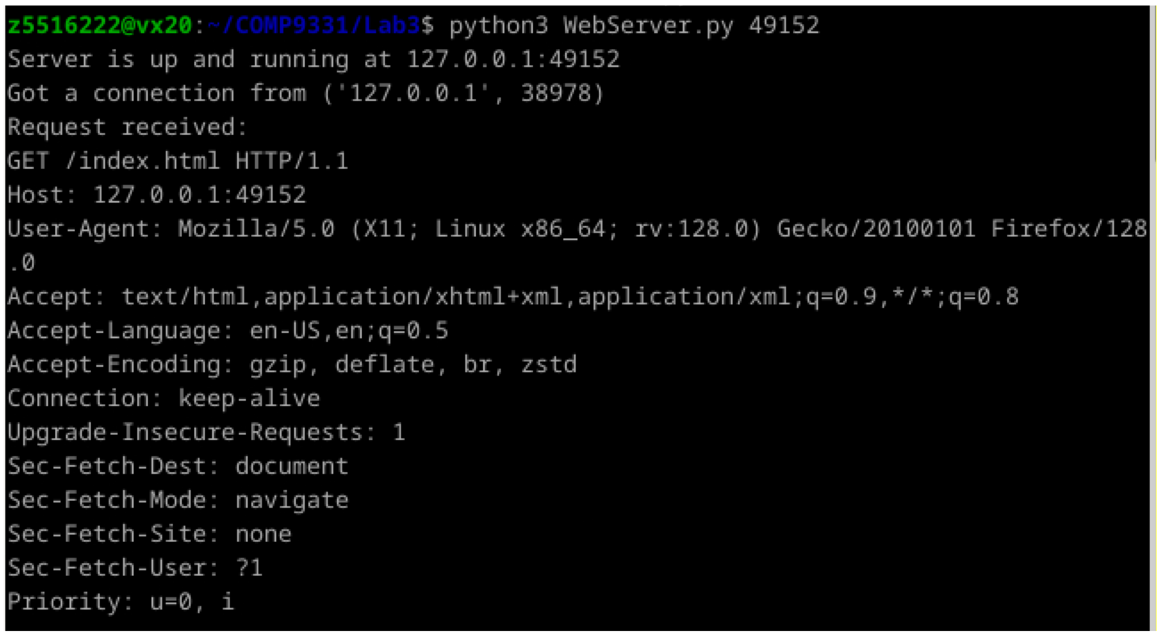
Note that you cannot use any pre-made web servers in different programming languages. Examples include http.server in Python.

Note 1: Most browsers send a GET request for a "favicon.ico" object when accessing a website. This is the site-specific icon you usually see in your browser's address bar. Your server is not required to handle this request. Your server can respond back with a 204 error (no content) for such requests. Further details about the favicon can be found here .

Note 2: We have added a tag to the header field of the index.html file, which should prevent the browser from sending the favicon.ico request. Use the updated index.html page (linked at the top of this page)

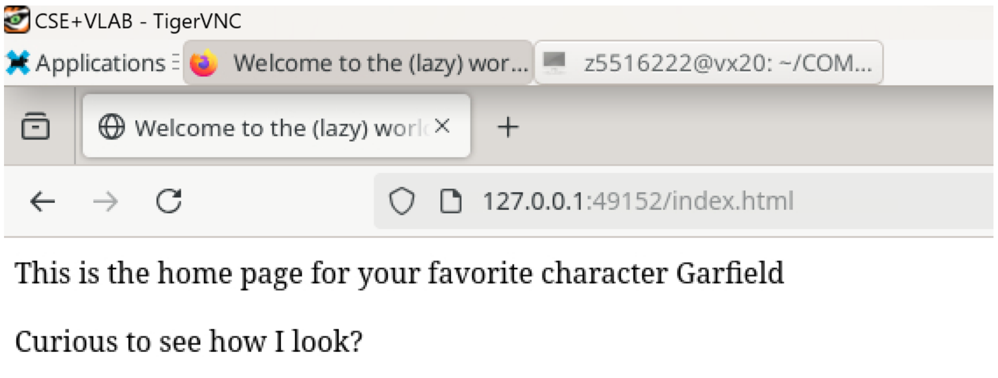
Answer: The screenshots of feedback is as follows, and the python code “WebClient.py” document is shown as attachment:

Run the python code with port 49152 (since dynamic port is 49152-65535), with opening feedback of GET “index.html”



```
z5516222@vx20:~/COMP9331/Lab3$ python3 WebServer.py 49152
Server is up and running at 127.0.0.1:49152
Got a connection from ('127.0.0.1', 38978)
Request received:
GET /index.html HTTP/1.1
Host: 127.0.0.1:49152
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128
.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u=0, i
```

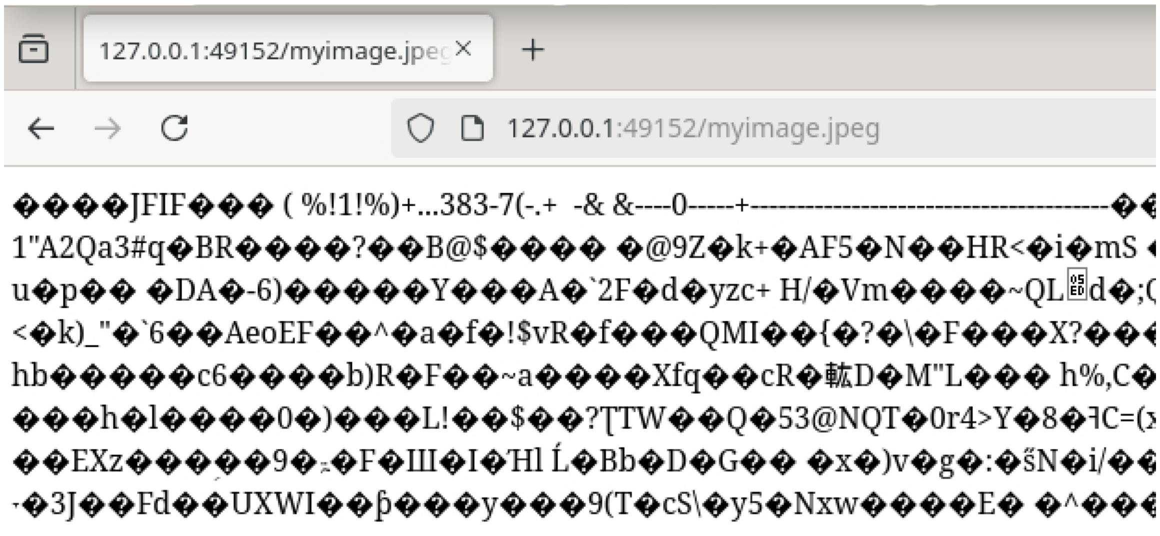
Run the web address in FireFox Web browser, the feedback is content of the html file


CSE+VLAB - TigerVNC
Applications
Welcome to the (lazy) wor...
z5516222@vx20: ~/COM...
Welcome to the (lazy) worl
127.0.0.1:49152/index.html

This is the home page for your favorite character Garfield

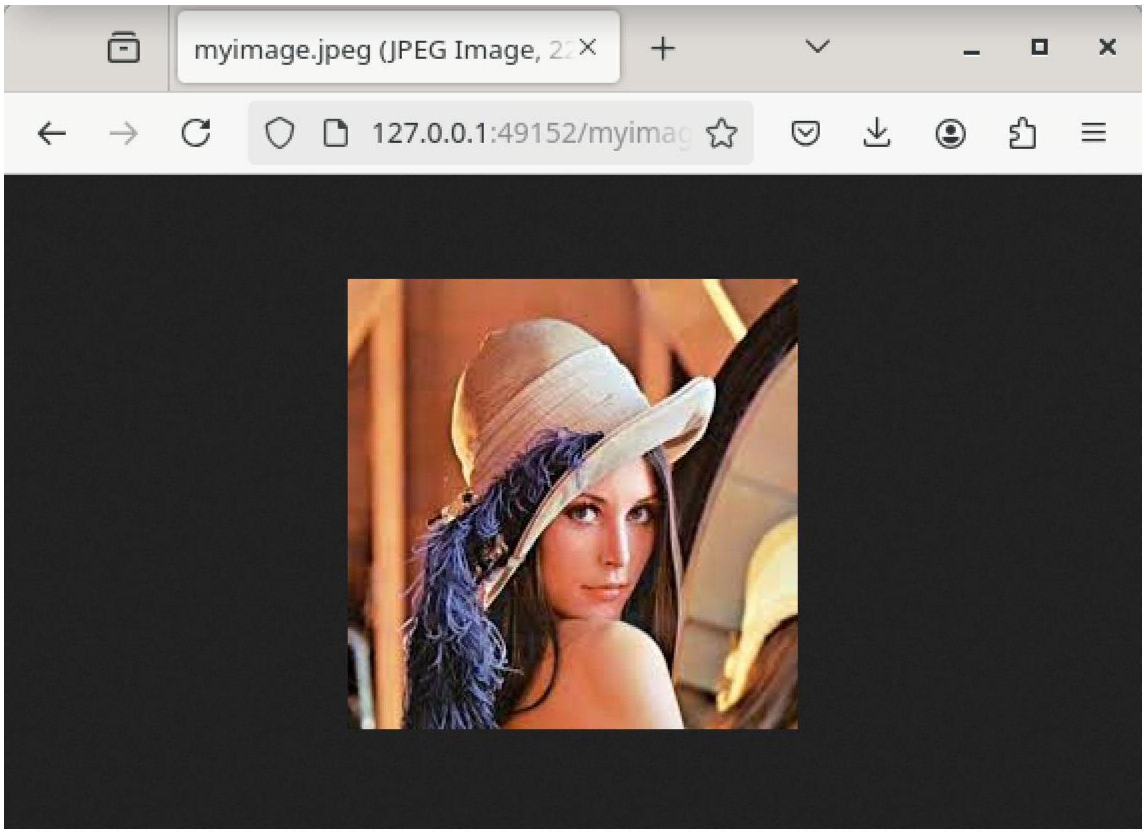
Curious to see how I look?

Trying to open a local “myimage.jpeg” file, but the browser couldn’t analyze:

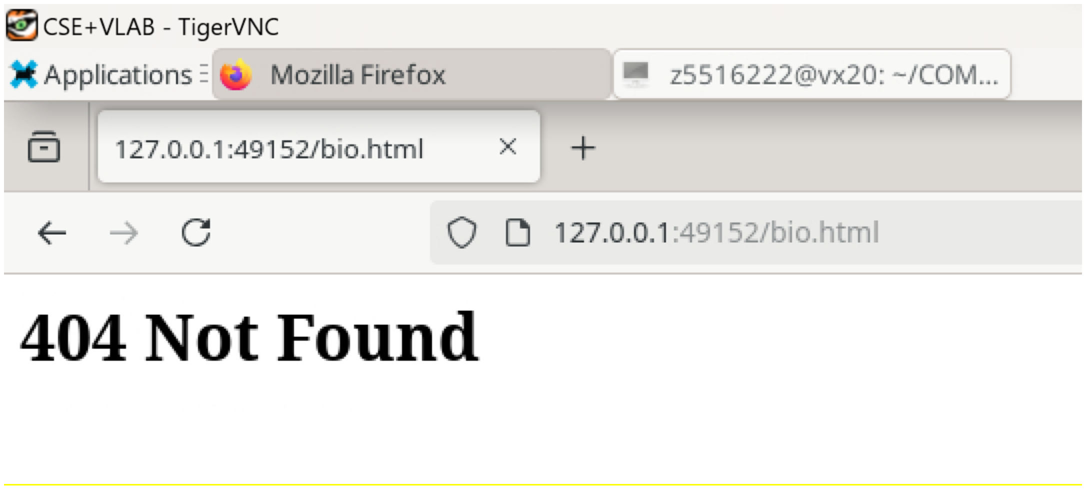


```
����JFIF��� ( %!1!%)+...383-7(-.+  -& &----0-----+----------------------------------------��
1"A2Qa3#q�BR�����?��B@$����� �@9Z�k+�AF5�N��HR<�i�mS �
u�p�� �DA�-6)������Y���A�`2F�d�yzc+ H/�Vm�����~QL d�;(
<�k)_"�`6��AeoEF��^�a�f�!$vR�f����QMI��{�?�\�F����X?���
hb������c6�����b)R�F��~a�����Xfq��cR�軚D�M"L���� h%,C�
���h�l�����0�)���L!��$���?ŢTW��Q�53@NQT�0r4>Y�8�ꟻC=(x
��EXz�������9�:�F�Ш�I�Hl Ĺ�Bb�D�G�� �x�)v�g�:�šN�i/��
·�3J��Fd��UXWI��þ����y����9(T�cS\�y5�Nxw�����E� �^���
```

Adding line in code “response += f"Content-Length: {len(file_content)}\r\n"”, problem solved

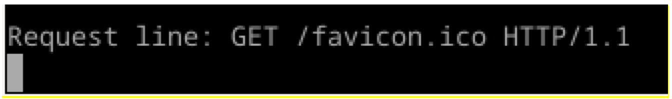


Try an inexistent document “bio.html”, showing “404 Not Found”



404 Not Found

Feedback in command line



```
Request line: GET /favicon.ico HTTP/1.1
```

Attachment (WebClient.py):

import socket

```
import os

def start_server(host='127.0.0.1', port=49152):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(5)
    print(f"Server is up and running at {host}:{port}")

    while True:
        client_socket, client_address = server_socket.accept()
        print(f"Got a connection from {client_address}")
        handle_request(client_socket)

def handle_request(client_socket):
    request = client_socket.recv(1024).decode()
    print(f"Request received:\n{request}")

    request_line = request.splitlines()[0]
    print(f"Request line: {request_line}")

    if request_line.startswith('GET'):
        file_path = request_line.split()[1]
        if file_path == '/':
            file_path = '/index.html'

        try:
            with open('.' + file_path, 'rb') as file:
                file_content = file.read()
            response = "HTTP/1.1 200 OK\r\n"
            response += "Content-Type: text/html; charset=utf-8\r\n"
            response += f"Content-Length: {len(file_content)}\r\n"
            response += f"Content-Length: {len(file_content)}\r\n"
            response += "Connection: keep-alive\r\n"
            response += "\r\n"
            response = response.encode() + file_content
        except FileNotFoundError:
            response = "HTTP/1.1 404 Not Found\r\n"
            response += "Content-Type: text/html; charset=utf-8\r\n"
            response += "Connection: keep-alive\r\n"
            response += "\r\n"
            response += "<h1>404 Not Found</h1>"
            response = response.encode()
    else:
        response = "HTTP/1.1 404 Not Found\r\n"
        response += "Content-Type: text/html; charset=utf-8\r\n"
        response += "Connection: keep-alive\r\n"
        response += "\r\n"
        response += "<h1>404 Not Found</h1>"
        response = response.encode()

    client_socket.sendall(response)
    client_socket.close()

if __name__ == '__main__':
    start_server()
```