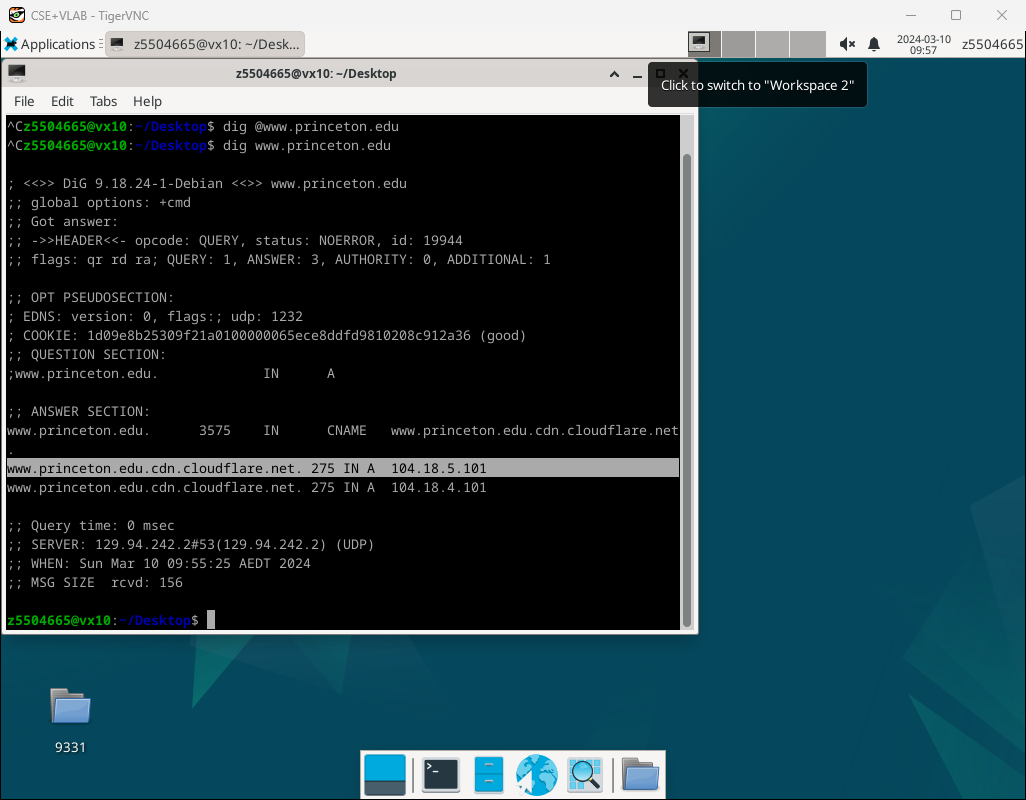
**Lab3**

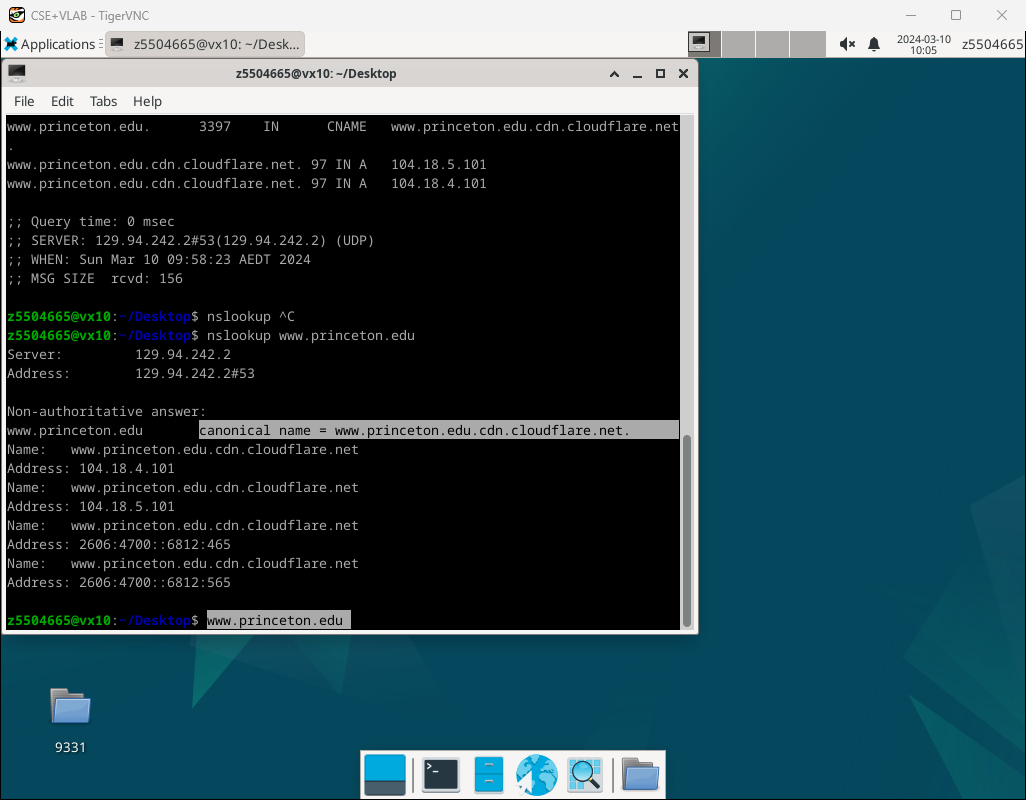
Exercise 3

**Q1**



The ip addresses of [www.princeton.edu](http://www.princeton.edu) are 104.18.5.101 and 104.18.4.101. And type of DNS query is A type. Which means this query will return a 32-bit [IPv4](https://en.wikipedia.org/wiki/IPv4) address, most commonly used to map [hostnames](https://en.wikipedia.org/wiki/Hostname) to an IP address of the host, but it is also used for [DNSBLs](https://en.wikipedia.org/wiki/DNSBL), storing [subnet masks](https://en.wikipedia.org/wiki/Subnet_mask) in RFC 1101, etc.

**Q2**

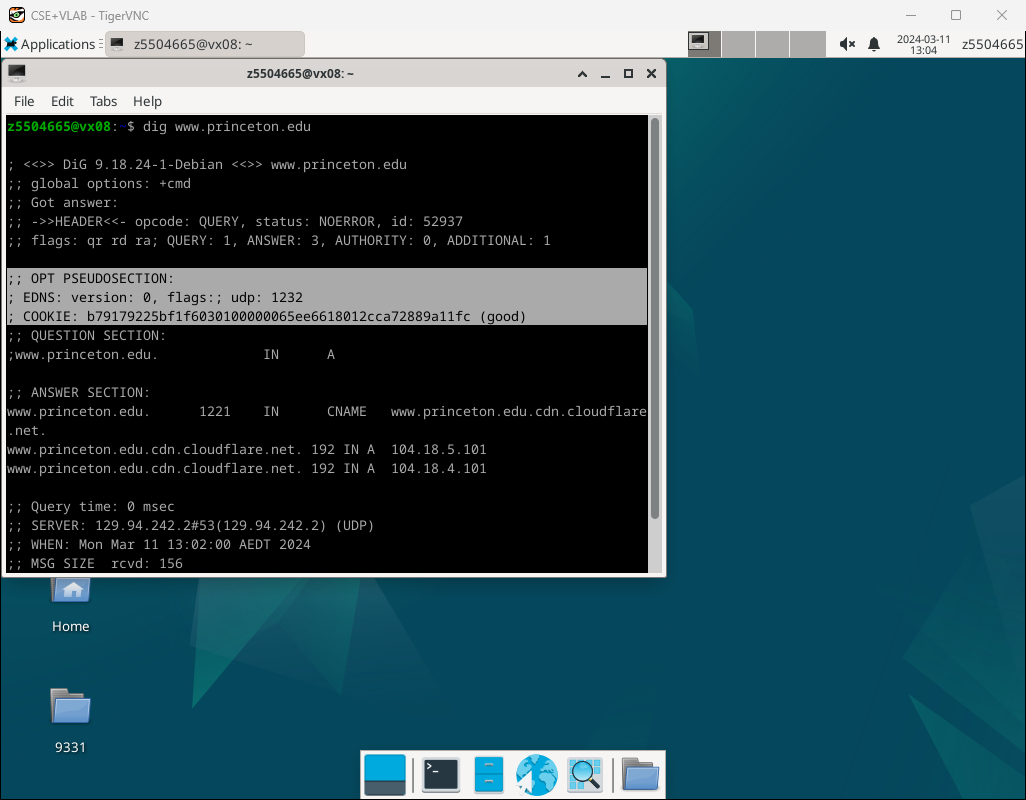


The canonical name of the Princeton webserver is [www.princeton.edu.cdn.cloudflare.net](http://www.princeton.edu.cdn.cloudflare.net).

The one reason for having alias is load balancing, this sever may accept a large amount of traffic, by having these alias can help ensure the website remains responsive and available even during periods of high demand.

And another reason is to make the server have a brief and Memorable name for users.

**Q3**



**EDNS:** EDNS stands for "Extension Mechanisms for DNS" (Domain Name System). It's a set of extensions to DNS that enable DNS clients (such as web browsers or other applications) and servers to communicate additional information that goes beyond the traditional DNS protocol limitations.

**EDNS: version: 0, flags:; udp: 1232** indicates that EDNS version 0 is used, with no specific flags set, and the maximum UDP packet size supported by the responder is 1232 bytes.

**COOKIES:** n the context of DNS, a COOKIE is often used as part of DNS Cookies, which is a mechanism for mitigating DNS-based attacks such as amplification attacks or cache poisoning. The COOKIE serves as a token to validate the authenticity of DNS transactions and to establish or maintain stateful communication between the client and server.

**COOKIE:b79179225bf1f6030100000065ee6618012cca72889a11fc (good)** is a randomly generated token (COOKIE) used for stateful tracking or verification of DNS transactions. In this case, "(good)" suggests that the cookie is valid.

**Query time: 0 msec**

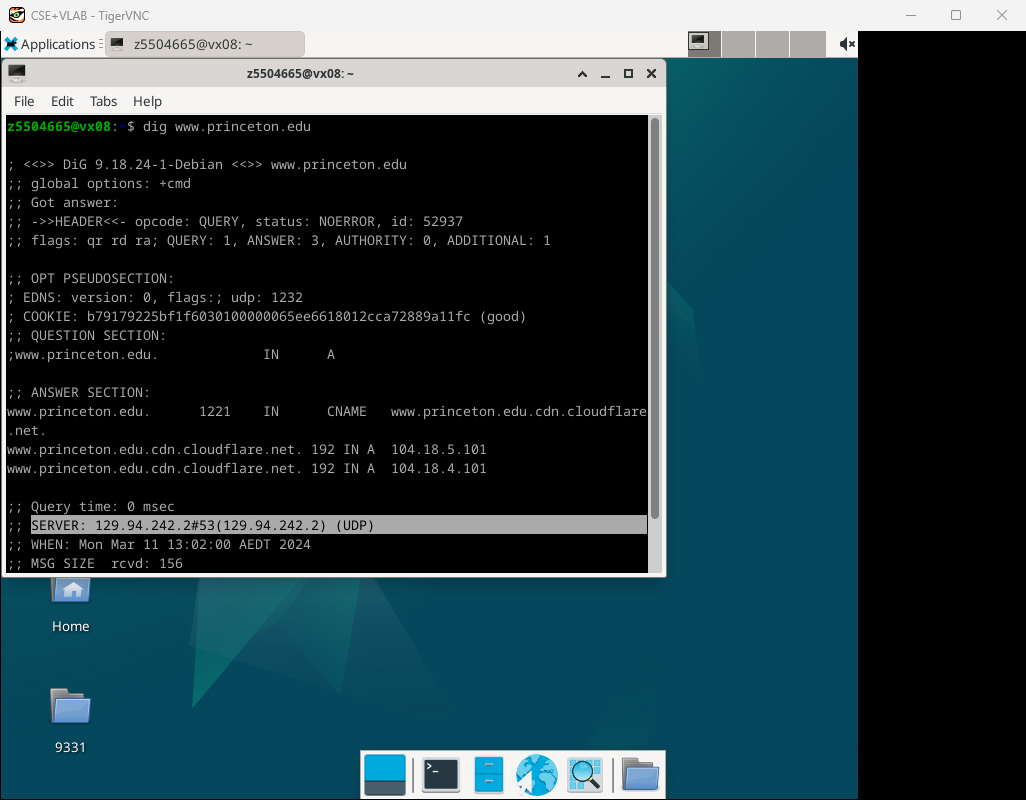
**;; SERVER: 129.94.242.2#53(129.94.242.2) (UDP)**

**;; WHEN: Mon Mar 11 13:02:00 AEDT 2024**

**;; MSG SIZE rcvd: 156**

These message are some information of response including message size,time ,server ip and server type.

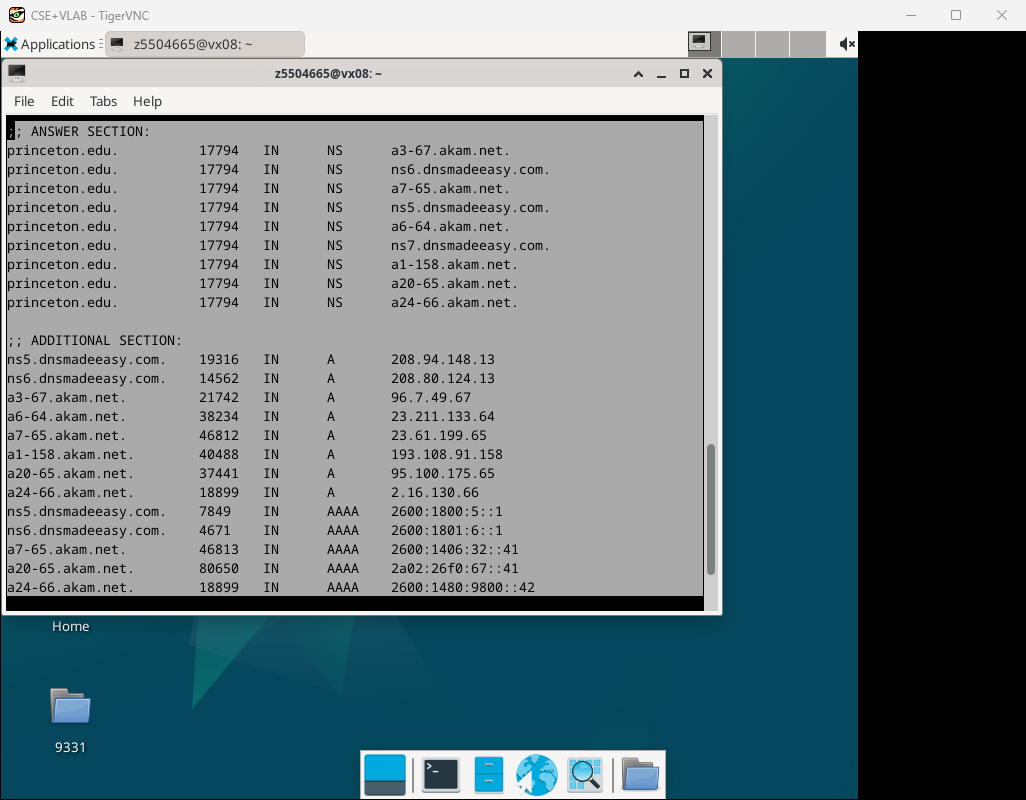
**Q4**

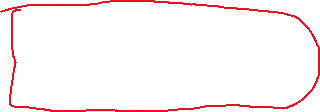
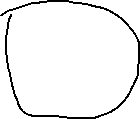


It is 129.94.242.2

**Q5**

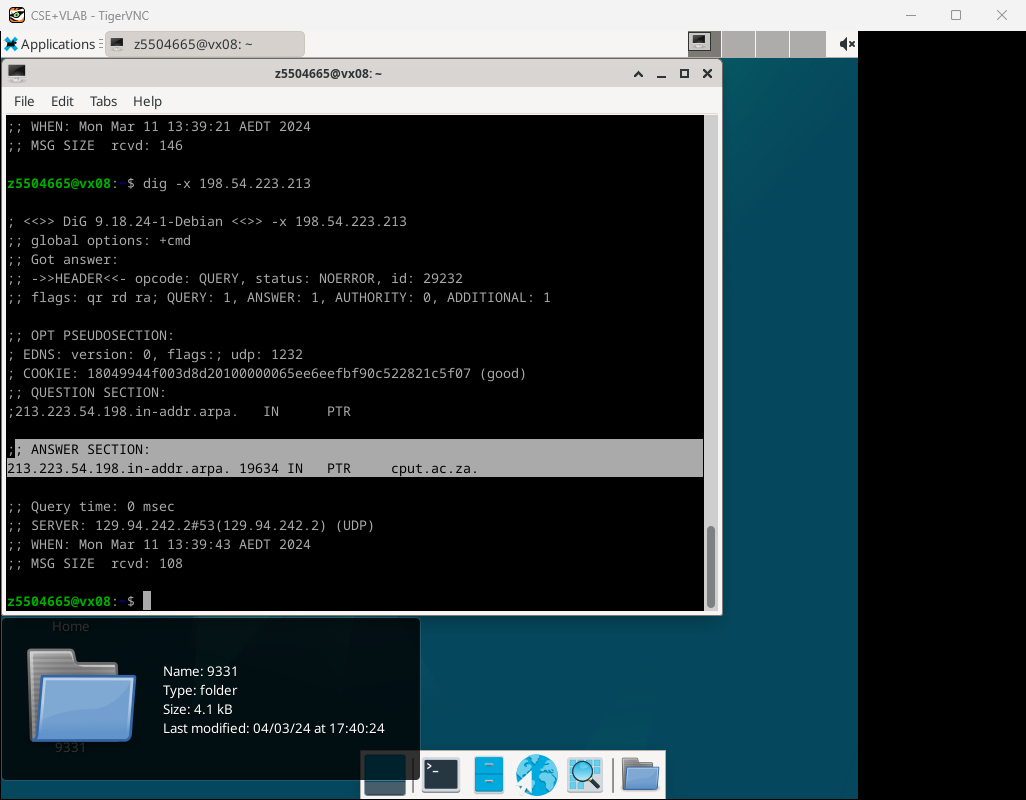
The type of query should be NS





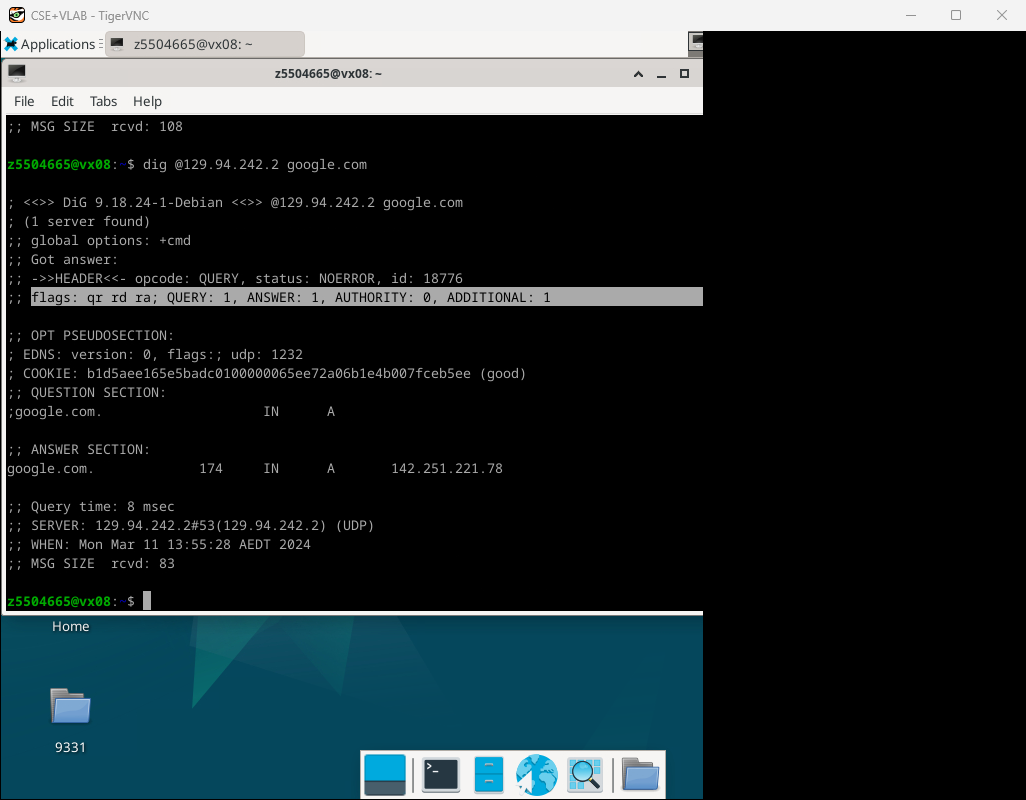
And IP addresses of these nameservers are marked in the picture.

Q6



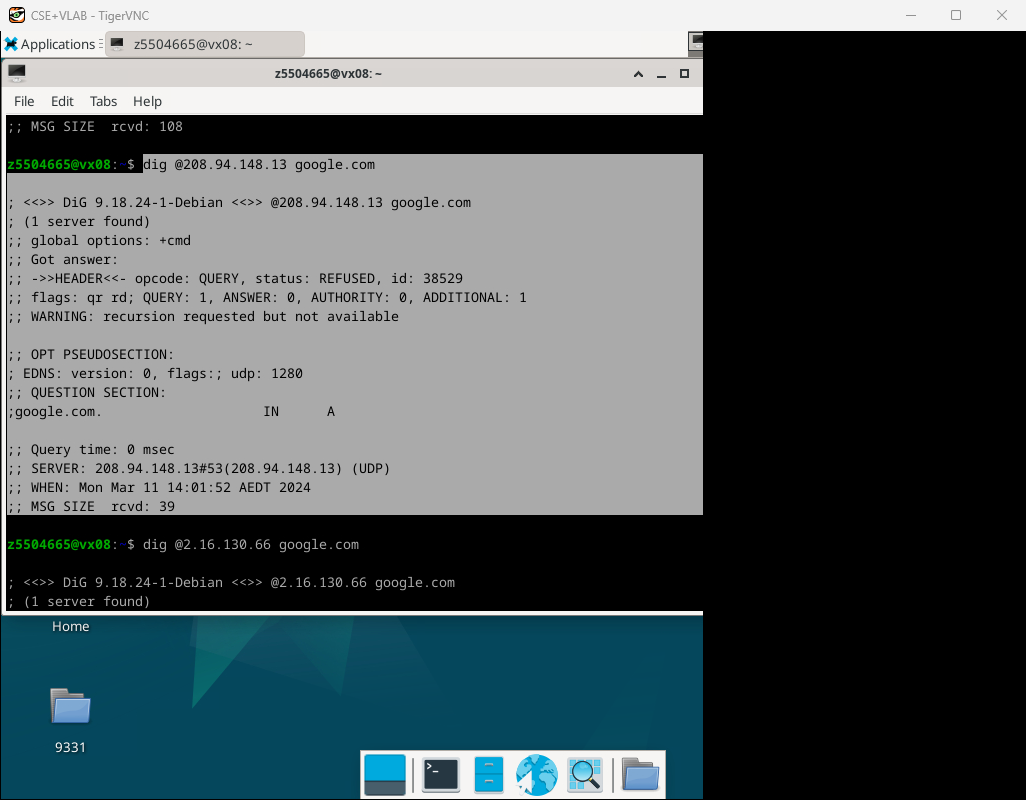
 the DNS name associated with the IP address 198.54.223.213 is **cput.ac.za,** and **the type of query is -x.**

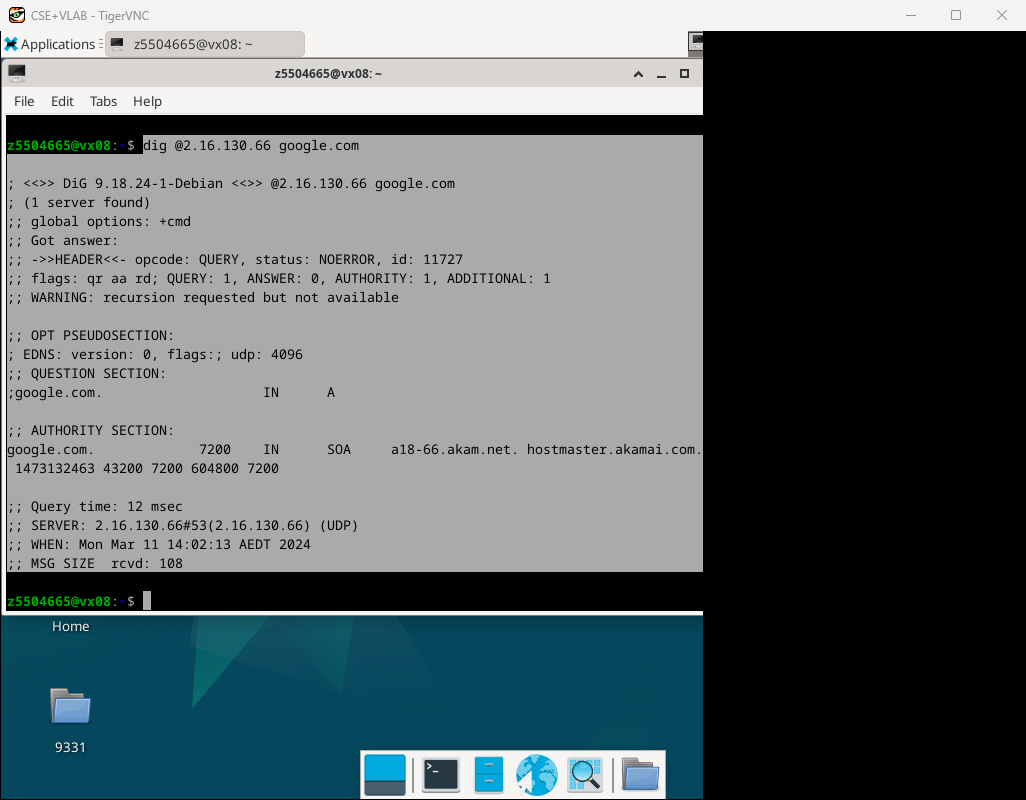
**Q7**



No, I did not. Because the flags line isn’t contain the “aa” tag.

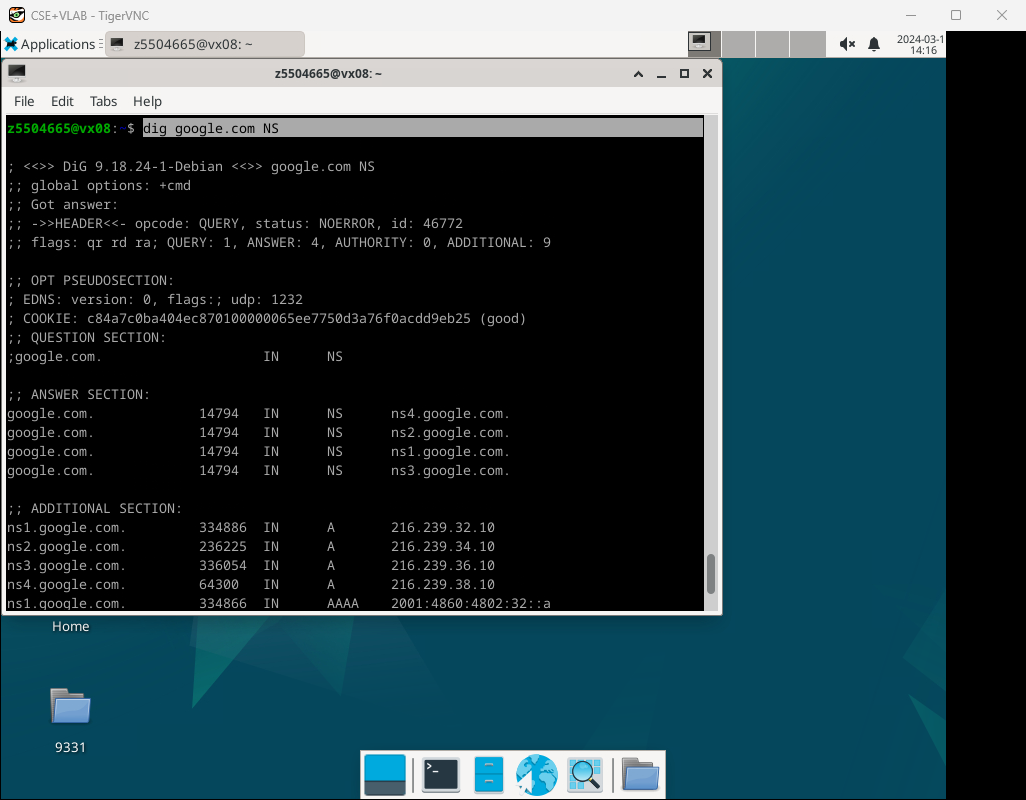
**Q8**

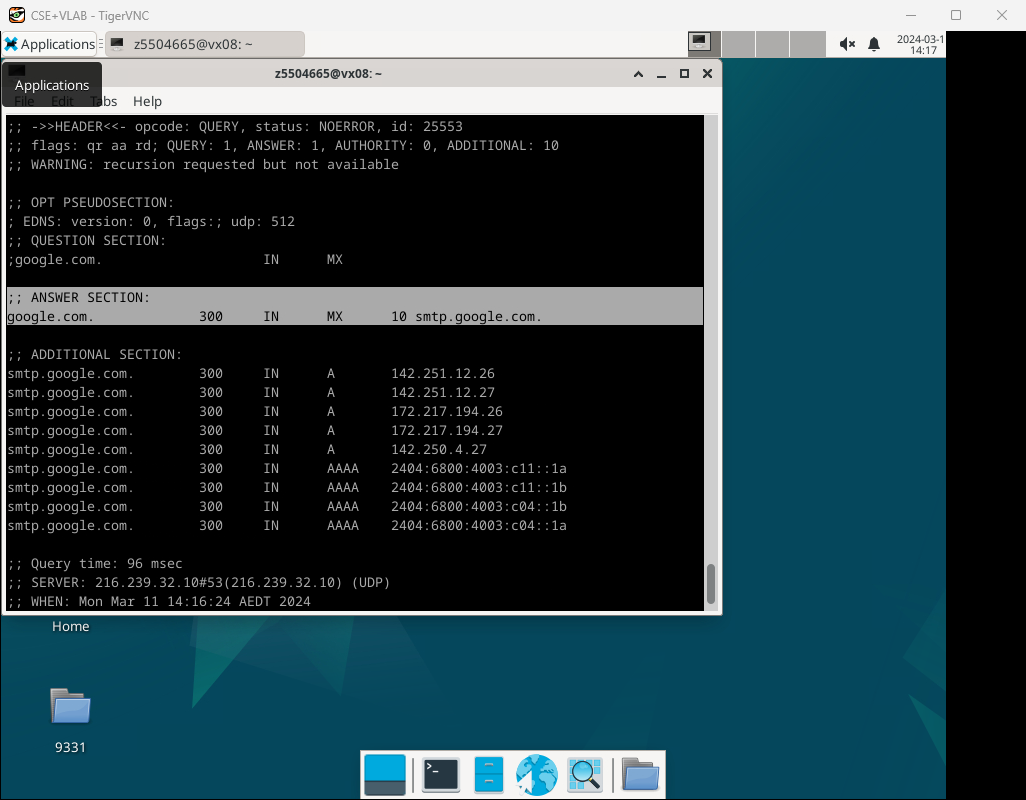




I tried it twice by using two different ip address that I got in Q5, and by using 208.94.148.13 I got no Authority answer, but I got Authority answer by using 2.16.130.66.

Q9

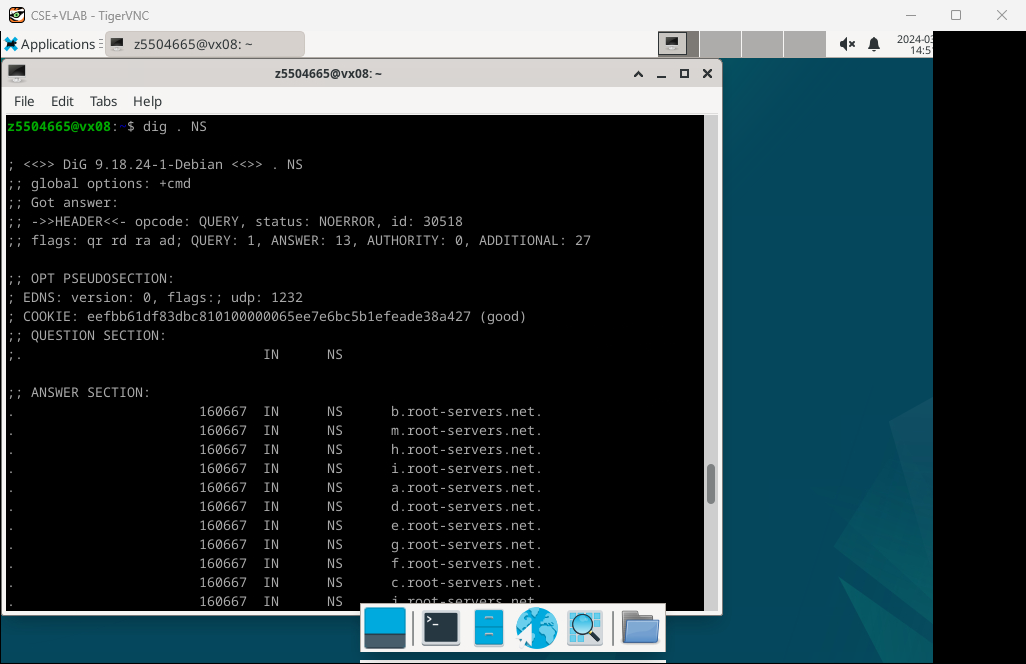


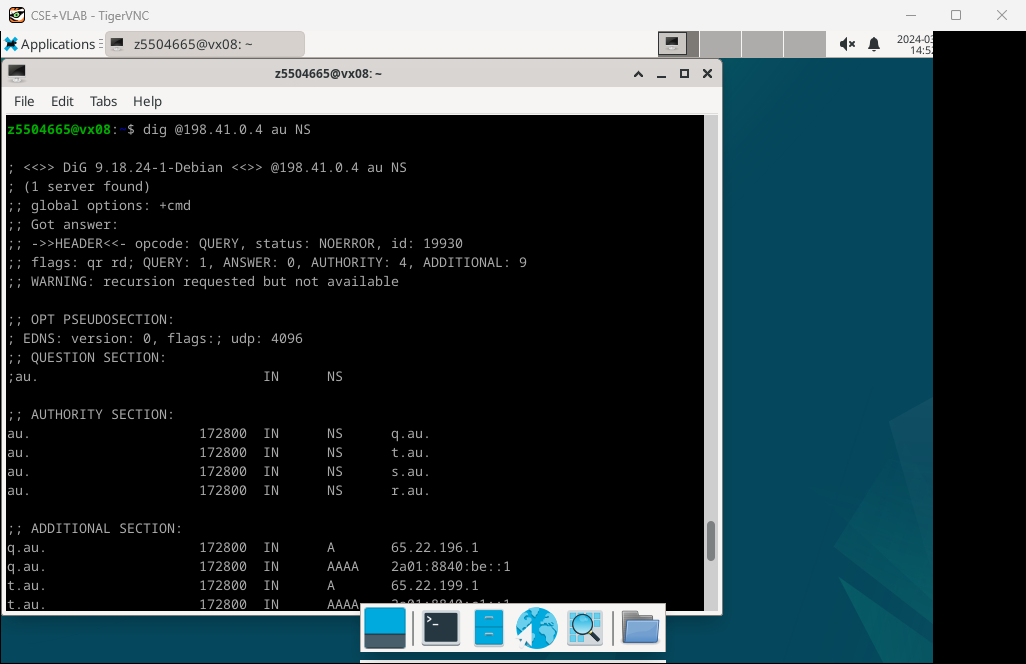


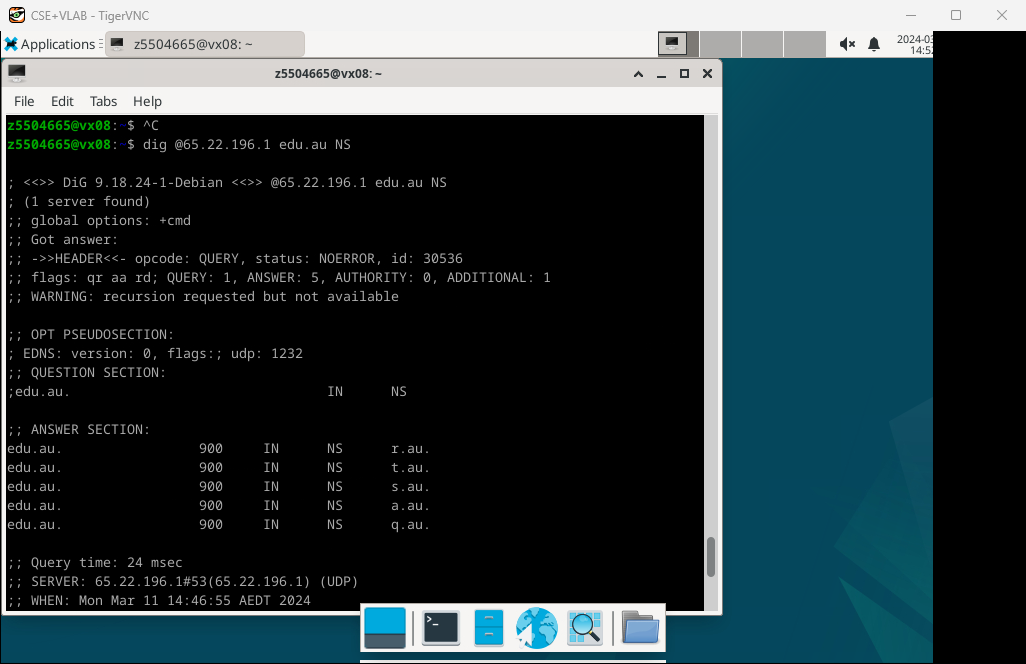
The answer is **google.com. 300 IN MX 10 smtp.google.com.** And **the type of query is MX.**

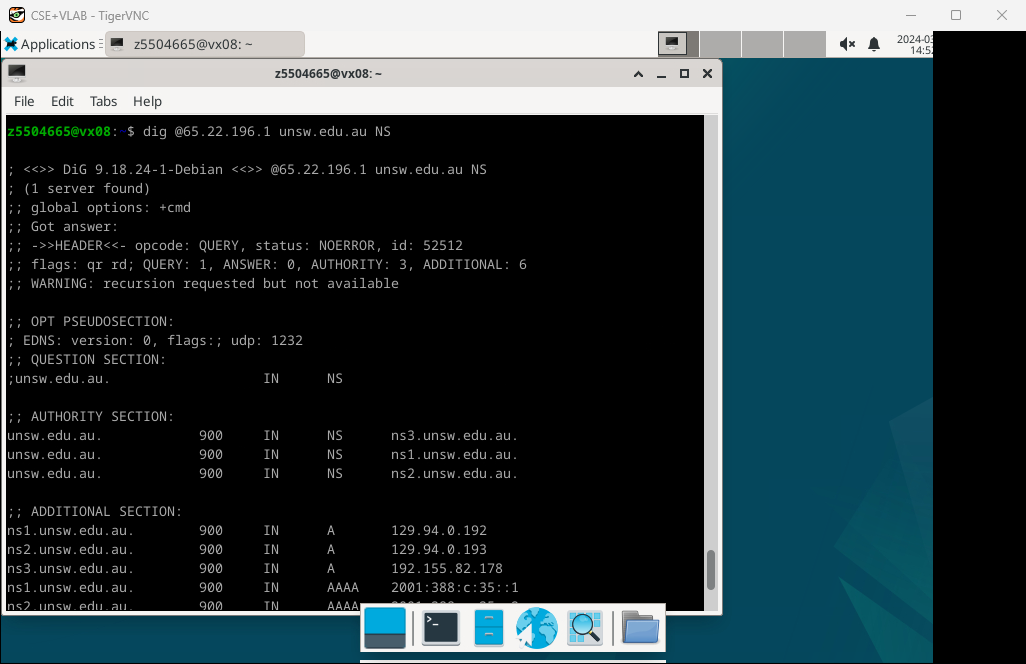
**Q10**

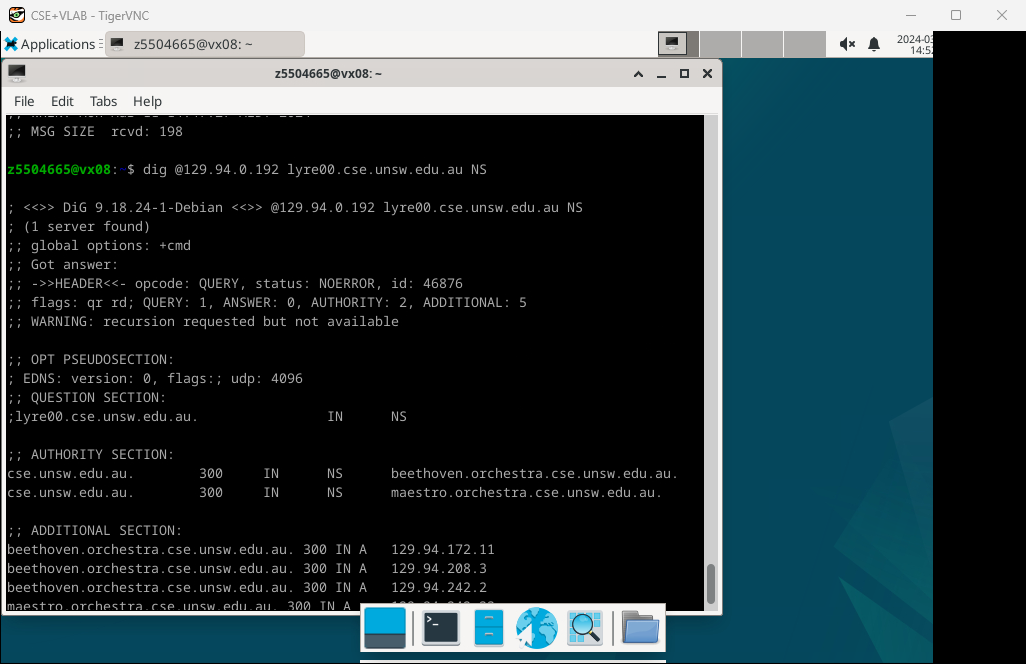
5 times









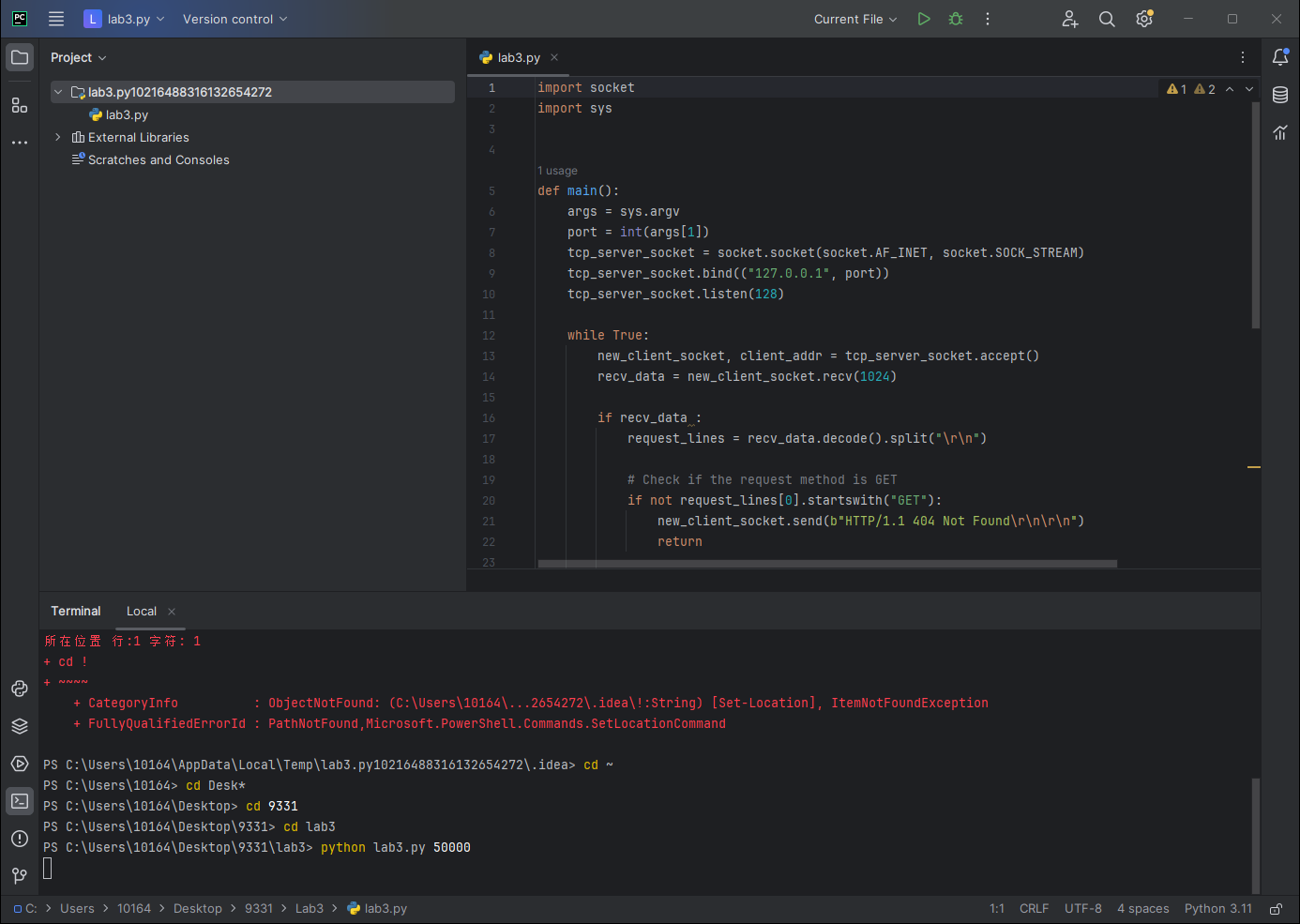


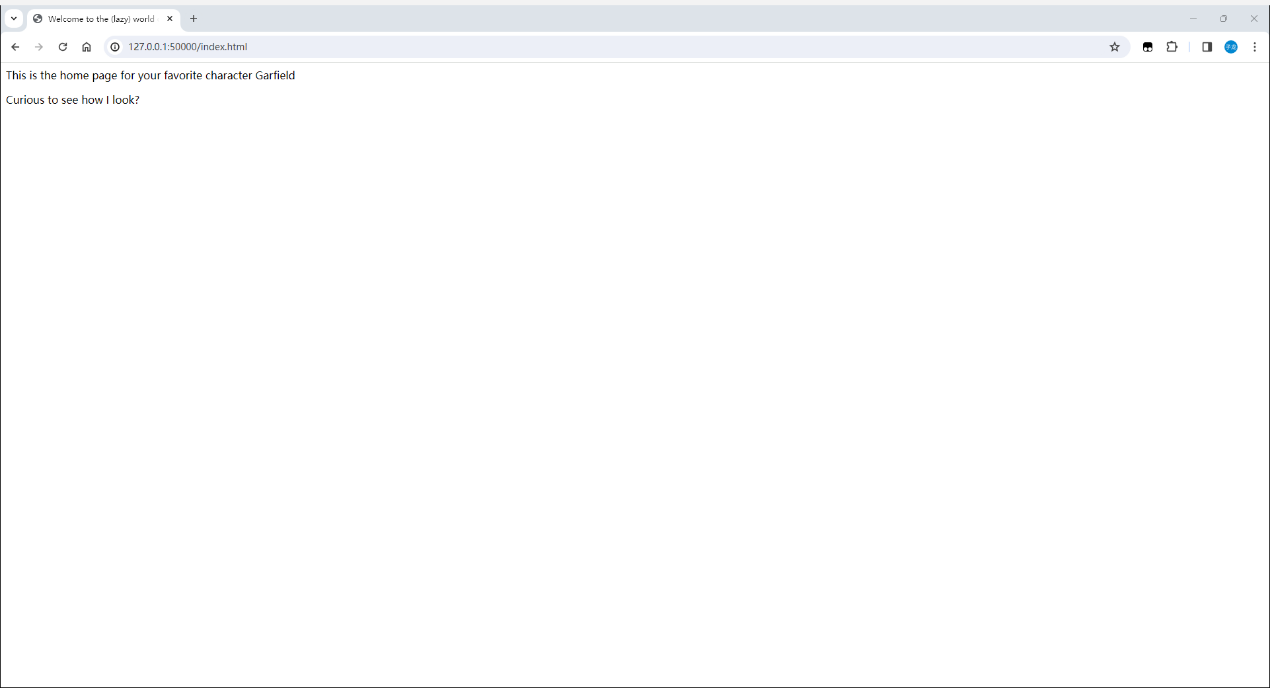
**Q11**

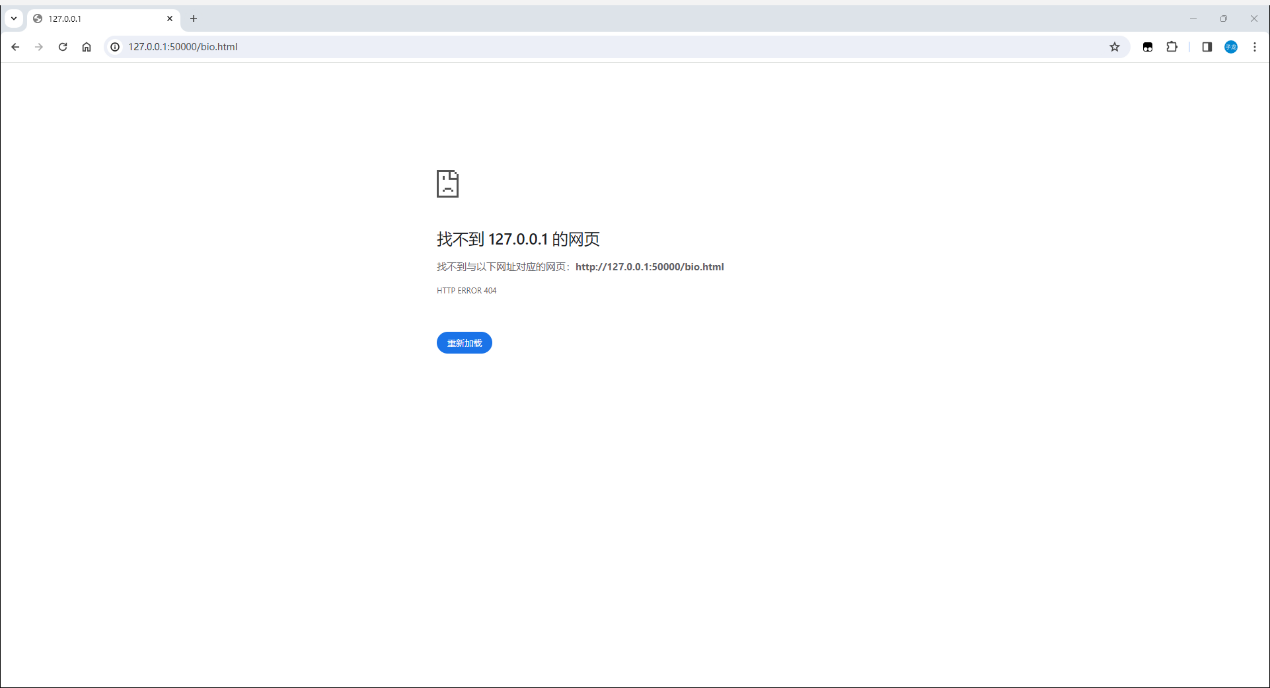
Yes, one physical machine can indeed have several names and/or IP addresses associated with it. This is a common practice in networking and can serve various purposes, such as load balancing, virtual hosting, and network segmentation.

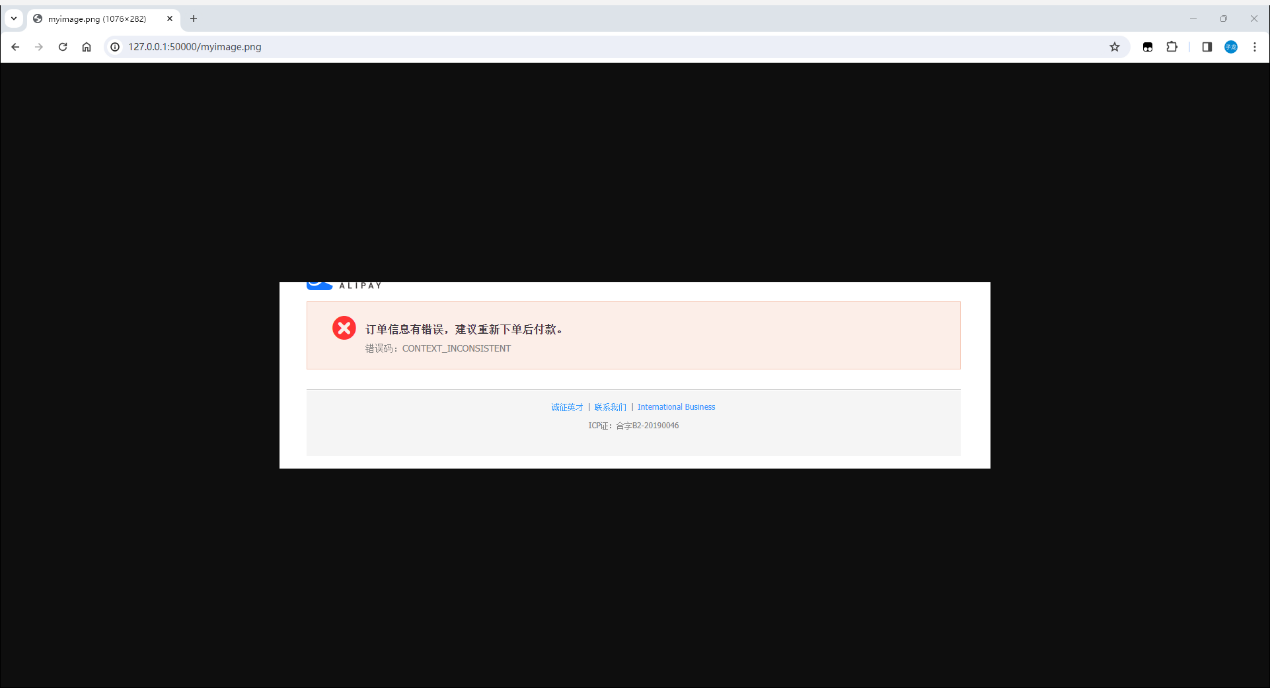
Exercise 4

Test：









Code：

import socket  
import sys  
  
  
def main():  
 args = sys.argv  
 port = int(args[1])  
 tcp\_server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 tcp\_server\_socket.bind(("127.0.0.1", port))  
 tcp\_server\_socket.listen(128)  
  
 while True:  
 new\_client\_socket, client\_addr = tcp\_server\_socket.accept()  
 recv\_data = new\_client\_socket.recv(1024)  
  
 if recv\_data :  
 request\_lines = recv\_data.decode().split("\r\n")  
  
 # Check if the request method is GET  
 if not request\_lines[0].startswith("GET"):  
 new\_client\_socket.send(b"HTTP/1.1 404 Not Found\r\n\r\n")  
 return  
  
 # Parse the requested file path  
 requested\_file\_path = request\_lines[0].split()[1]  
  
 # Serve the requested file if it exists, else return 404  
 try:  
 with open("." + requested\_file\_path, "rb") as file:  
 file\_content = file.read()  
 response\_headers = "HTTP/1.1 200 OK\r\nContent-Length: {}\r\n\r\n".format(len(file\_content))  
 new\_client\_socket.send(response\_headers.encode() + file\_content)  
 except FileNotFoundError:  
 new\_client\_socket.send(b"HTTP/1.1 404 Not Found\r\n\r\n")  
  
 new\_client\_socket.close()  
  
 tcp\_server\_socket.close()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()