Lab 3: SMTP Lab

By the end of this lab, you will have acquired a better understanding of SMTP protocol. You will also gain experience in implementing a standard protocol using Python.

Your task is to develop a simple mail client that sends email to any recipient. Your client will need to connect to a mail server, dialogue with the mail server using the SMTP protocol, and send an email message to the mail server. Python provides a module, called `smtplib`, which has built in methods to send mail using SMTP protocol. However, we will not be using this module in this lab, because it hide the details of SMTP and socket programming.

In order to limit spam, some mail servers do not accept TCP connection from arbitrary sources. For the experiment described below, you may want to try connecting both to your university mail server and to a popular Webmail server, such as a AOL mail server. You may also try making your connection both from your home and from your university campus.

Code

Below you will find the skeleton code for the client. You are to complete the skeleton code. The places where you need to fill in code are marked with **`#Fill in start`** and **`#Fill in end`**. Each place may require one or more lines of code.

Additional Notes

In some cases, the receiving mail server might classify your e-mail as junk. Make sure you check the junk/spam folder when you look for the e-mail sent from your client.

What to Hand in

In your submission, you are to provide the complete code for your SMTP mail client as well as a screenshot showing that you indeed receive the e-mail message.

Skeleton Python Code for the Mail Client

```
from socket import *


msg = "\r\n I love computer networks!"

endmsg = "\r\n.\r\n"


# Choose a mail server (e.g. Google mail server) and call it mailserver

mailserver = #Fill in start   #Fill in end


# Create socket called clientSocket and establish a TCP connection with mailserver

#Fill in start


#Fill in end

recv = clientSocket.recv(1024).decode()

print(recv)

if recv[:3] != '220':

       print('220 reply not received from server.')


# Send HELO command and print server response.

heloCommand = 'HELO Alice\r\n'

clientSocket.send(heloCommand.encode())

recv1 = clientSocket.recv(1024).decode()

print(recv1)

if recv1[:3] != '250':

    print('250 reply not received from server.')
```

```
# Send MAIL FROM command and print server response.
# Fill in start


# Fill in end


# Send RCPT TO command and print server response.
# Fill in start


# Fill in end


# Send DATA command and print server response.
# Fill in start


# Fill in end


# Send message data.
# Fill in start


# Fill in end
# Message ends with a single period.
# Fill in start


# Fill in end


# Send QUIT command and get server response.
# Fill in start
# Fill in end
```

Optional Exercises

1. Mail servers like Google mail (address: smtp.gmail.com, port: 587) requires your client to add a Transport Layer Security (TLS) or Secure Sockets Layer (SSL) for authentication and security reasons, before you send MAIL FROM command. Add TLS/SSL commands to your existing ones and implement your client using Google mail server at above address and port.
2. Your current SMTP mail client only handles sending text messages in the email body. Modify your client such that it can send emails with both text and images.