Assignment 2 (Bonus):

Due Date: Nov. 29th,11:59 p.m.

Objectives:

- Practicing more socket programming in python
- Better understanding of SMTP protocol
- Implementing a standard email protocol, SMTP, using python

General Instruction

- You may work in groups of at most two students or individually.
- **Start early.**
- Your program should use TCP protocol.
- **You must NOT use python smtplib module**

Submission Instructions

- A file named `SMTPClient.py`
- **The mail addresses of sender, their password, and the email address of the receiver should be provided as a command line argument**
- **The content of messages should be hardcoded in the program itself**
- **Do NOT submit an eclipse project!**
- Make sure to put your names (as well as your partner's name if any) on top of the `SMTPClient.py` file.
- **Make sure to have one submission per group**

Description

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 through the mail server. Refer to the slides on Application layer where a sample SMTP interaction is provided.

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

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 assignment**, because it hides the details of SMTP and socket programming.

In order to limit spam, some mail servers do not accept TCP connection from arbitrary sources. For testing your program, you may want to try connecting both to your university mail server and to a popular Webmail server, such as a gmail mail server. You may also try making your connection both from your home (off campus) and from your university campus.

Skeleton Python Code for the Mail Client

```
#import libraries needed for communication with a secure email server
from socket import *
import ssl
import re
from email.base64mime import body_encode as encode_base64

def auth_plain(sender_email, password):
     """ Authobject to use with PLAIN authentication. Requires
         self.user and
         self.password to be set."""
    return "\0%s\0%s" % (sender_email, password)

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

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

# Choose a mail server (e.g. Google mail server)
mailserver = 'smtp.gmail.com'

# Make a TCP connection with mailserver and receive the server
#response
#check the server response and print it to the screen

# Send HELO command to the server and print server response. It
should be ehlo not helo
heloCommand = 'ehlo [' + addr +']\r\n'

# Send STARTTLS command and print server response.
# wrap the socket you created earlier in a ssl context. Assuming you
# named you socket, clientSocket, you can use the following two lines
# to do so:

context = ssl.create_default_context()
clientSocket = context.wrap_socket(clientSocket,
server_hostname=mailserver)
```

Now, the client needs to authenticate to the server. The AUTH command is used for this purpose. The AUTH command sends the clients username and password to the e-mail server. AUTH can be combined with some other keywords such as PLAIN, LOGIN, CRAM-MD5 and DIGEST-MD5 (e.g. AUTH LOGIN) to choose an authentication mechanism. The authentication mechanism chooses how to login and which level of security that should be used. To authenticate using AUTH PLAIN you do so:

Send the AUTH PLAIN command to the server.
Command = ‘AUTH PLAIN\r\n’

After the client has sent the AUTH PLAIN command to the server, the server responds with a 334 reply code. Then the username and password are sent from the client to the server. The username and password are combined to one string and BASE64 encoded (using `encode_base64` function in python). To combine username and password you can use the `auth_plain` function given above. Although the keyword PLAIN is used, the username and password are not sent as plain text over the Internet - they are always BASE64 encoded. If you want to read more about BASE64 encoding, look here.

```
# Send DATA command and print server response.
# Send message data.
# Message ends with a single period.
# Send QUIT command and get server response.
```

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.

- 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. The details are mentioned below
- RFC for SMTP protocol: **https://datatracker.ietf.org/doc/html/rfc5321**
- If you are using Gmail server, instead of the password, you need to use app password. An app password is a 16-digit passcode that gives a less secure app or device permission to access your Google Account. App passwords can only be used with accounts that have 2-Step Verification turned on.
- https://support.google.com/accounts/answer/185833?hl=en
- If you do not have a gmail app password, create a new app using generate password. Check your apps and passwords https://myaccount.google.com/apppasswords. Good Luck.

Extra Marks

The above 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.