Simple Mail Transfer Protocol (SMTP) simplified.

Introduction

The aim of this exercise is to implement a simplified mail protocol. The task is split into stages, each more advanced than the previous one.

You will be responsible for the client code (server code will be provided for each step). Try to write down the session type for each stage yourself, however, if you get very stuck, look at the source code and move on from there.

Use GV syntax for this exercise.

Stage 1 (Easy)

Protocol:

1) Client can only select **MAIL** command. It then sends an **Address** to the server.

2) Having received an **Address**, server can choose either **REJECT** or **ACCEPT**. In case of **REJECT**, server sends an **Error** message and the session **ends**. In case of **ACCEPT**:

3) Server sends an **Accept** message to the client. Client can then only select **RCPT** label and then send a 'from' **Address**.

4) The server can choose either **REJECT** or **ACCEPT**. In case of **REJECT**, server sends an **Error** message and the session **ends**. In case of **ACCEPT**:

5) Server sends an **Accept** message to the client. Only **DATA** command is then available for the client. The client then sends a **Message** they want to send as e-mail body, receives an **Accept** message from the server and the session **ends**.

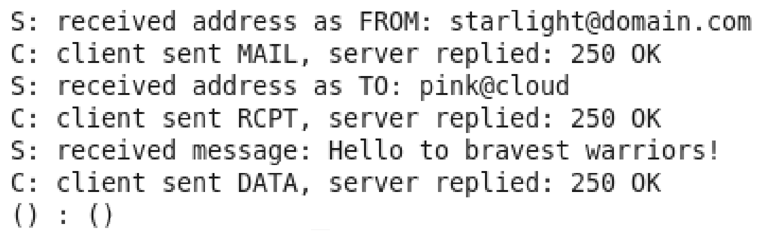
Corresponding session type:

```
typename SMTPServer =
 [&|MAIL:?Address.
          [+|REJECT:!Error.EndBang,
             ACCEPT:!Accept.
                     [&|RCPT:?Address.
                              [+|REJECT:!Error.EndBang,
                                 ACCEPT:!Accept.
                                         [&|DATA:?Message.!Accept.EndBang |&] |+] |&] |+] |&] ;
```

Address, Error, Accept and Message are all types of String (example: typename Address = String;). Now use the provided server code for this stage and implement the client.

The server will print out the information it receives from the client. Make sure your client code prints out the messages it receives from the server as well.

Sample output:

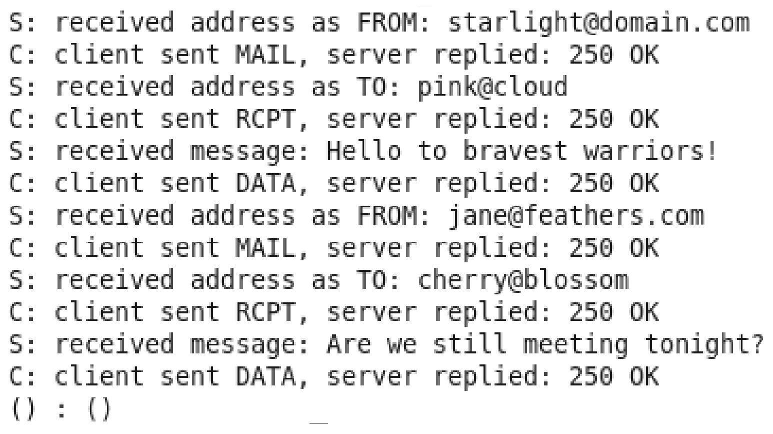


```
S: received address as FROM: starlight@domain.com
C: client sent MAIL, server replied: 250 OK
S: received address as TO: pink@cloud
C: client sent RCPT, server replied: 250 OK
S: received message: Hello to bravest warriors!
C: client sent DATA, server replied: 250 OK
() : ()
```

S: lines are printed out by the server. C: lines are printed out by the client.

Stage 2 (Medium)

Now extend the session type from stage 1 so that when the client receives an **Accept** message from the server in step 5, the session would not end. Instead it would go **back to step 1** and the client could send another e-mail. However, the session has to end at some point, therefore add a choice of **QUIT** in step 1 (= client can choose either **MAIL** or **QUIT** in step 1). Sample output (two e-mails sent in one session):



```
S: received address as FROM: starlight@domain.com
C: client sent MAIL, server replied: 250 OK
S: received address as TO: pink@cloud
C: client sent RCPT, server replied: 250 OK
S: received message: Hello to bravest warriors!
C: client sent DATA, server replied: 250 OK
S: received address as FROM: jane@feathers.com
C: client sent MAIL, server replied: 250 OK
S: received address as TO: cherry@blossom
C: client sent RCPT, server replied: 250 OK
S: received message: Are we still meeting tonight?
C: client sent DATA, server replied: 250 OK
() : ()
```

Hint: take a look at recursion in session types.

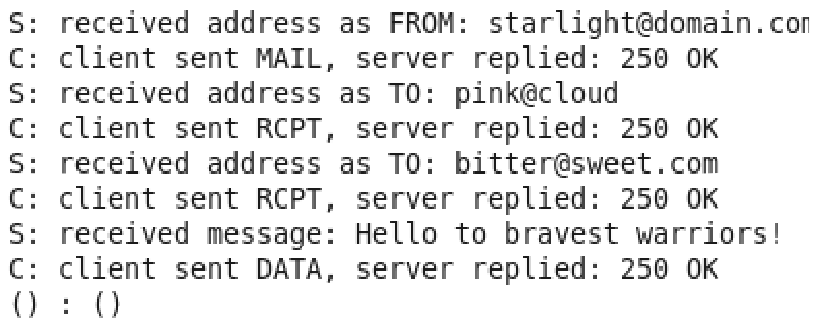
Stage 3 (Difficult)

You might have noticed that our session type only allows one recipient at a time. Extend stage 2 in the following way:

In step 3, client can now choose both **RCPT** and **DATA**. In case of **RCPT**, the server can issue **REJECT** or **ACCEPT**. In case of **REJECT**, server sends an **error** message and the session goes back to **step 3**. In case of **ACCEPT**, an **Accept** message is sent and the session goes back to **step 3**. In case of **DATA**, the client then sends a **Message** they want to send as e-mail body, receives an **Accept** message from the server and then can again choose either **MAIL** or **QUIT**.

It means the client can select zero or more recipients. The client can keep adding recipients until **DATA** is selected (sounds like recursion!)

Sample output:



```
S: received address as FROM: starlight@domain.co
C: client sent MAIL, server replied: 250 OK
S: received address as TO: pink@cloud
C: client sent RCPT, server replied: 250 OK
S: received address as TO: bitter@sweet.com
C: client sent RCPT, server replied: 250 OK
S: received message: Hello to bravest warriors!
C: client sent DATA, server replied: 250 OK
() : ()
```

Stage 4 (Expert)

This stage is designed for those seeking a challenge. Do not worry at all if you do not manage to complete this in the lab time!

Extend the session type in such a way that the client would have to add **at least one** recipient whose e-mail address is valid, but could also add multiple recipients, if they wanted to.