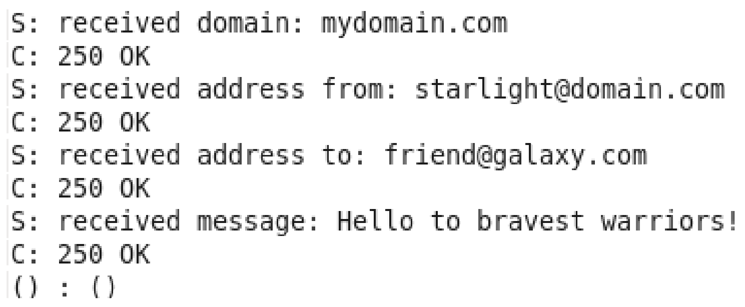
Exercise 2: Simple Mail Transfer Protocol (SMTP) simplified.

0. Introduction

The aim of this exercise is to implement a simplified mail protocol. You will be responsible for the client in both parts. You should be able to produce two versions of this exercise for each part: in GV and CP (4 in total).

1. Part 1 (Simple)

In this part you will write a client side for SMTP. You are provided with the session type and server side code. The session type for this task is very basic, there always is only one possible choice. The server side will print all the data it receives from the client you will write. The client should print all the data it receives from server as well! The output of sample solution:



```
S: received domain: mydomain.com
C: 250 OK
S: received address from: starlight@domain.com
C: 250 OK
S: received address to: friend@galaxy.com
C: 250 OK
S: received message: Hello to bravest warriors!
C: 250 OK
() : ()
```

S: lines are printed out by server (so you do not need to worry about them) and they include the data client has sent. C: lines are sent by server to the client and the client should print them out.

Session type (for server):

```
typename Domain = String;
typename Address = String;
typename Accept = String;
typename Message = String;

typename SMTPServer = [&|EHLO:?Domain.!Accept.
                   [&|MAIL:?Address.!Accept.
                     [&|RCPT:?Address.!Accept.
                       [&|DATA:?Message.!Accept.
                         [&|QUIT:EndBang|&]|&]|&]|&]|&];
```

You will need to write a client for this protocol that sends a single message. You may wish to begin by figuring out the session type from the client’s perspective (i.e., the dual of the session type above).

2. Part 2 (Advanced)

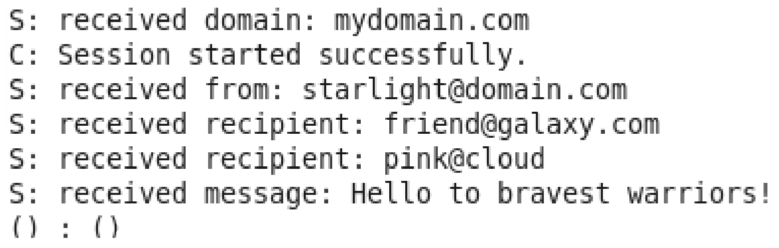
In this part you will extend the existing session type from part 1 to make the protocol more realistic. You are be provided with the code for the server side, but it is your task to come up with an appropriate session type matching the description below as well as implement it as the client side.

Protocol description:

1. The first and only command that is available for the client is **EHLO**. The client then sends a **domain** and expects a **welcome message** from server.
2. The client is now able to choose either QUIT or MAIL. In case of **QUIT**, the session **ends**. In case of **MAIL**, the client sends an e-mail **address** that acts as ‘from’. The client can then receive either **REJECT** or **ACCEPT** labels from the server. In case of REJECT, client receives an **error message** from the server and goes back to step 2. In case of ACCEPT, client enters step 3.
3. The client can then choose **RCPT** command followed by an e-mail **address** that acts as ‘to’. The client can then receive either **REJECT** or **ACCEPT** labels from the server. In case of REJECT, client receives an **error message** from the server and goes back to step 3. In case of ACCEPT, client enters step 4.
4. The client can choose either RCPT or DATA. In case of **RCPT**, the client sends an e-mail **address** (acts as ‘to’ again). The client can then receive either **REJECT** or **ACCEPT** labels from the server. In case of REJECT, client receives an **error message** from the server and goes back to step 4. In case of ACCEPT, client immediately goes back to step 4. In case of DATA, the client sends a message to the server and enters step 2.

Syntax hint: for reject/accept, the client becomes a server. That is, rather than selecting an option as before, it expects the server to select an option and then deals with it.

This protocol requires at least one recipient before being able to send a message (there can be multiple). In your client example you can choose to have a single recipient and then send a message, or have couple of recipients and then send a message. Then you may quit or send another e-mail – these decisions are up to you. You can also provide incorrect e-mail address at some point and print out the error message. In our sample solution, there are 2 valid recipients and the session is ended after sending one e-mail. Sample output:



```
S: received domain: mydomain.com
C: Session started successfully.
S: received from: starlight@domain.com
S: received recipient: friend@galaxy.com
S: received recipient: pink@cloud
S: received message: Hello to bravest warriors!
() : ()
```

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