Prova in itinere – 11-04-2022 – Laboratorio (3 punti) SOLUZIONE!!!!!

Nome Cognome	**Studente Bravo**
Matricola	

Il codice sotto riportato implementa un'applicazione Client-Server. Il Client implementa una versione "affidabile" di UDP che gestisce la ritrasmissione in caso di mancata risposta dal Server. In particolare, lato Client vengono eseguiti i seguenti passaggi: a) viene creata una stringa inizialmente vuota, b) l'utente inserisce un messaggio che viene concatenato alla stringa, c) il client tenta di trasmettere la stringa intera, attendendo una risposta dal Server, d) se il Server non risponde dopo 1 secondo il Client torna al punto b), altrimenti il programma termina. Dopo 3 tentativi senza successo, il Client chiude il socket e stampa a video un messaggio di avviso per la mancata risposta dal Server. Il Server, se attivo, risponde inviando al Client la stringa ricevuta, con i caratteri in maiuscolo. **(3 punti)**

Hint:

- Concatenazione stringhe: x = 'a' + 'b' equivale a x = 'ab'
- Formattazione f-string:
  - message = 'ciao'
  - print(f'{message} a tutti') # stampa 'ciao a tutti'

Client

```
from socket import *
serverPort = 19277
serverName = '127.0.0.1'
clientSocket = socket(AF_INET, SOCK_DGRAM)
tout = 1
clientSocket.settimeout(tout)
maxretrial = 3
stringa = ''
success = False
for i in range(maxretrial):
    print(f'Tentativo di comunicazione {i+1}...')
    message = input('Inserisci stringa: ')
    stringa = stringa + ' ' + message
    try:
        clientSocket.sendto(stringa.encode('utf-8') , (serverName,serverPort))
        resp, serverAddress = clientSocket.recvfrom(1024)
        print('Dal Server: ', resp.decode('utf-8'))
        success = True
        break
    except timeout:
        print(f'Tentativo {i+1} fallito.')
        print(f'Stringa accumulata: {stringa}')
if not success:
    print('Comunicazione con il server non riuscita.')
clientSocket.close()
```

Server

```
from socket import *

serverPort = 19277
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('',serverPort))

while True:
    msg, client_address = serverSocket.recvfrom(1024)
    resp = msg.decode('utf-8').upper()
    serverSocket.sendto(resp.encode('utf-8'), client_address)
```

Q1. (2pt) Si completino i codici di Client e Server.

Q2. (1pt) Si scriva quante volte viene chiamata la funzione "print()" lato Client nel caso in cui vengano effettuati 3 tentativi di trasmissione caratterizzati come segue **(8 chiamate):**

- `Tentativo 1: l'utente inserisce "ciao" <-> Server spento;`
- `Tentativo 2: l'utente inserisce "come" <-> Server spento;`
- `Tentativo 3: l'utente inserisce "stai?" <-> Server acceso;`

Codice esercizi laboratorio

UDP client

```
from socket import *
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message, (serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()
```

UDP server

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "The server is ready to receive"
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    print "Datagram from: ", clientAddress
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

UDP error management

```
from socket import *
serverName = 'localhost'
serverPort = 12001
clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.settimeout(5)
message = raw_input('Input lowercase sentence:')
try:
    clientSocket.sendto(message, (serverName, serverPort))
    modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
    # in case of error blocks forever
    print modifiedMessage
except error, v:
    print "Failure"
    print v
finally:
    clientSocket.close()
```

TCP client

```
from socket import *
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

TCP server

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while True:
    connectionSocket,  clientAddress = serverSocket.accept()
```

```
    print "Connection form: ", clientAddress
    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

TCP client persistent

```
from socket import *
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
while True:
    sentence = raw_input('Input lowercase sentence ( . to stop):')
    clientSocket.send(sentence)
    if sentence == '.':
        break
    modifiedSentence = clientSocket.recv(1024)
    print 'From Server:', modifiedSentence
clientSocket.close()
```

TCP server persistent

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
while True:
    print 'The server is ready to receive'
    connectionSocket,  clientAddress = serverSocket.accept()
    print "Connection form: ", clientAddress
    while True:
        sentence = connectionSocket.recv(1024)
        if sentence == '.':
            break
        capitalizedSentence = sentence.upper()
        connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

TCP auto client

```
from socket import *
import time
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
for a in range(100):
    clientSocket.send('A')
time.sleep(1)
clientSocket.send('.')
#clientSocket.recv(1024)
clientSocket.close()
```

TCP auto server

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
while True:
    print 'The server is ready to receive'
    connectionSocket,  clientAddress = serverSocket.accept()
    print "Connection form: ", clientAddress
    while True:
```

```
        sentence = connectionSocket.recv(1024)
        if sentence == '.':
            break
        print len(sentence)
#        connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

TCP server thread

```
from socket import *
import thread
def handler(connectionSocket):
    while True:
        sentence = connectionSocket.recv(1024)
        if sentence == '.':
            break
        capitalizedSentence = sentence.upper()
        connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
while True:
    print 'The server is ready to receive'
    newSocket,  addr = serverSocket.accept()
    thread.start_new_thread(handler, (newSocket,))
```