# *Prova in itinere – Luglio 2018 – Laboratorio (6 punti)*

|  |  |
| --- | --- |
| **Nome Cognome** |  |
| **Matricola** |  |

**SOLUZIONI IN ROSSO**

Il codice sotto riportato rappresenta una versione semplificata di un sistema di prenotazione di posti per eventi. Il client richiede e visualizza la lista degli eventi disponibili e chiede all’utente di il numero dell’evento per cui vuole prenotare un posto. Gli eventi possono essere pieni, il server deve verificare se ci sono posti disponibili per l’evento richiesto prima di dare conferma al client.

**Server**

|  |
| --- |
| from socket import \*  serverPort = 12000  listaEventi = ["1 - Evento 1", "2 - Evento 2", "3 - Evento 3"]  postiEventi = [0, 1, 2]  serverSocket = socket(AF\_INET, SOCK\_STREAM) # Welcome socket  serverSocket.bind(('', serverPort))  serverSocket.listen(1)  while 1:  print "Server in attesa di connessioni"  connectionSocket, addr = serverSocket.accept()  request = ""  while request != ".":  request = connectionSocket.recv(2048)  if request == "EVENTI":  reply = ""  for ev in listaEventi:  reply += ev + "\n"  connectionSocket.send(reply)  # v.isdigit() restituisce True/False se la var. v è una  # stringa che rappresenta un numero  elif request.isdigit() and int(request) in range(1,len(listaEventi)+1):  ev\_num = int(request)  if postiEventi[ev\_num-1] > 0:  **postiEventi[ev\_num-1] -= 1**  **reply = “OK”**  else:  **reply = “NO\_SPACE”**  connectionSocket.send(reply)  elif request != ".":  reply = "KO"  connectionSocket.send(reply)  connectionSocket.close() |

**Client**

|  |
| --- |
| from socket import \*  serverName = "localhost"  serverPort = 12000  clientSocket = socket(AF\_INET, SOCK\_STREAM)  clientSocket.connect((serverName,serverPort)  clientSocket.send("EVENTI")  elenco\_eventi = clientSocket.recv(**XXX**)  reply = "KO"  while reply == "KO" or reply == 'NO\_SPACE':  print "Elenco degli eventi disponibili:"  print elenco\_eventi  request = raw\_input("Inserisci la tua scelta: ")  clientSocket.send(request)  **reply = clientSocket.recv(2048)**  if reply == "KO":  print "Scelta non valida."  elif reply == "NO\_SPACE":  print "Non ci sono posti disponibili per l'evento selezionato"  else:  print "Prenotazione andata a buon fine!"  clientSocket.send(".")  **clientSocket.close()** |

**Q1.** Completare il codice del Client **(2 punti) Vedi Codice**

**Q2**. Completare il codice del Server **(3 punti) Vedi Codice**

**Q3**. Qual è il valore minimo che può assumere **XXX** affinchè riceva correttamente la lista degli eventi? Giustificare la risposta. Hint: anche spazi e andate a capo sono considerati caratteri **(1 punto)  
 Il valore minimo è 39. Ogni stringa associata agli eventi ha 12 caratteri, per ogni evento viene aggiunto un carattere di andata a capo “\n”**

**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(handle