**CN Lab 6**

Question 1:

**Server Code:**

import socket

def main():

# Creating the socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) #The parameter SOCK\_STREAM is for TCP connection establishment

# Binding the socket to IP and port

server\_address = ('127.0.0.1', 2000) # Localhost, port 2000

try:

server\_socket.bind(server\_address)

print("Socket successfully bound to IP and port")

except socket.error as err:

print(f"Bind failed. Error: {err}")

return

# Putting the socket into listening state

server\_socket.listen(1) #parameter is backlog which is the maximum number of connections that can be at a single time in the queue so max 1 conection at a time in this case

print("Listening for incoming connections...")

while True:

# Accepting a connection

client\_socket, client\_address = server\_socket.accept() #blocks execution and waits for a connection

#returns a connection socket and the client address

print(f"Client connected with IP: {client\_address[0]} and Port: {client\_address[1]}")

# Receiving the message from the client

try:

client\_message = client\_socket.recv(2000).decode('utf-8')

# print(f"Client Message: {client\_message}")

if client\_message!="exit":

finalMsg=f"Hello I am server. Your received ID is {client\_message[len(client\_message)-1]}"

print(finalMsg)

except socket.error as err:

print(f"Receive failed. Error: {err}")

client\_socket.close()

continue

if client\_message.lower()=="exit":

print("Exiting the loop...")

break

# Sending the message back to the client

# try:

# server\_message = client\_message # Echoing back the same message

# client\_socket.send(server\_message.encode('utf-8'))

# except socket.error as err:

# print(f"Send failed. Error: {err}")

# Closing the client connection

client\_socket.close()

# Closing the server socket (this won't be reached in an infinite loop, but could be useful in modifications)

server\_socket.close()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Client Code:**

import socket

def main():

# Creating the socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Server details (IP and port)

server\_address = ('127.0.0.1', 2000)

# Connecting to the server

try:

client\_socket.connect(server\_address)

print("Connected to server")

except socket.error as err:

print(f"Connection failed. Error: {err}")

return

# Get input from the user

client\_message = input("Enter ID (or exit):")

printMsg=True

if client\_message=="exit":

pass

else:

while not (int(client\_message)>=0 and int(client\_message)<=9):

client\_message=input("Enter ID between 0 and 9 (or exit):")

if (client\_message=="exit"):

printMsg=False

break

if printMsg:

client\_message=f"Hello I am client and my ID is {client\_message}"

print(client\_message)

# Send the message to the server

try:

client\_socket.send(client\_message.encode('utf-8'))

except socket.error as err:

print(f"Send failed. Error: {err}")

client\_socket.close()

return

# Receive the message back from the server

# try:

# server\_message = client\_socket.recv(2000).decode('utf-8')

# print(f"Server Message: {server\_message}")

# except socket.error as err:

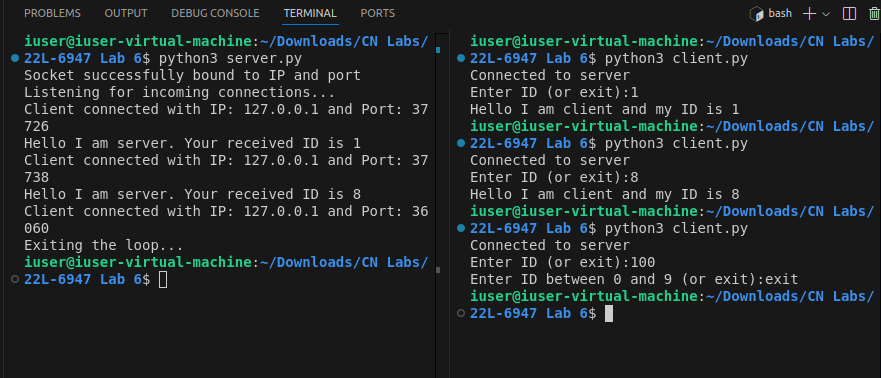
# print(f"Receive failed. Error: {err}")

# Closing the socket

client\_socket.close()

if \_\_name\_\_ == "\_\_main\_\_":

main()



**Question 2:**

**Server Code:**

import socket

def reverseStringOnVowels(str):

vowels="aeiouAEIOU"

wordList=str.split(" ")

newList=[]

# print(wordList)

for x in wordList:

isRev=False

for z in x:

if not isRev:

for y in vowels:

if z==y:

isRev=True

if isRev:

newList.append(x[::-1])

else:

newList.append(x)

reverseString=" ".join(newList)

return reverseString

def main():

# Creating the socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) #The parameter SOCK\_STREAM is for TCP connection establishment

# Binding the socket to IP and port

server\_address = ('127.0.0.1', 2000) # Localhost, port 2000

try:

server\_socket.bind(server\_address)

print("Socket successfully bound to IP and port")

except socket.error as err:

print(f"Bind failed. Error: {err}")

return

# Putting the socket into listening state

server\_socket.listen(1) #parameter is backlog which is the maximum number of connections that can be at a single time in the queue so max 1 conection at a time in this case

print("Listening for incoming connections...")

while True:

# Accepting a connection

client\_socket, client\_address = server\_socket.accept() #blocks execution and waits for a connection

#returns a connection socket and the client address

print(f"Client connected with IP: {client\_address[0]} and Port: {client\_address[1]}")

# Receiving the message from the client

try:

client\_message = client\_socket.recv(2000).decode('utf-8')

# print(f"Client Message: {client\_message}")

if client\_message!="exit":

revStr=reverseStringOnVowels(client\_message)

print(revStr)

except socket.error as err:

print(f"Receive failed. Error: {err}")

client\_socket.close()

continue

if client\_message.lower()=="exit":

print("Exiting the loop...")

break

# Sending the message back to the client

try:

server\_message = revStr # Echoing back the same message

client\_socket.send(server\_message.encode('utf-8'))

except socket.error as err:

print(f"Send failed. Error: {err}")

# Closing the client connection

client\_socket.close()

# Closing the server socket (this won't be reached in an infinite loop, but could be useful in modifications)

server\_socket.close()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Client Code:**

import socket

def reverseStringOnNoVowels(str):

vowels="aeiouAEIOU"

wordList=str.split(" ")

newList=[]

# print(wordList)

for x in wordList:

isRev=True

for z in x:

if isRev:

for y in vowels:

if z==y:

isRev=False

if isRev:

newList.append(x[::-1])

else:

newList.append(x)

reverseString=" ".join(newList)

return reverseString

def main():

# Creating the socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Server details (IP and port)

server\_address = ('127.0.0.1', 2000)

# Connecting to the server

try:

client\_socket.connect(server\_address)

print("Connected to server")

except socket.error as err:

print(f"Connection failed. Error: {err}")

return

# Get input from the user

client\_message = input("Enter a string:")

printMsg=True

if client\_message=="exit":

printMsg=False

if printMsg:

print("Client Message: ", client\_message)

# Send the message to the server

try:

client\_socket.send(client\_message.encode('utf-8'))

except socket.error as err:

print(f"Send failed. Error: {err}")

client\_socket.close()

return

# Receive the message back from the server

try:

server\_message = client\_socket.recv(2000).decode('utf-8')

print(f"Server Message: {server\_message}")

revStr=reverseStringOnNoVowels(server\_message)

print(f"Final Message: {revStr}")

except socket.error as err:

print(f"Receive failed. Error: {err}")

# Closing the socket

client\_socket.close()

if \_\_name\_\_ == "\_\_main\_\_":

main()

