

Ramakrishna Mission Vidyamandira

(An Autonomous College Under University of Calcutta)

Computer Science (Honors) Semester V 2025

Paper: 5CMSMJC3 Practical

|  |
| --- |
| **Submitted by** |
| Class Roll Number: 340  Registration Number:  B.Sc.  5th Semester  Batch: 2023-27 |

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SI NO.** | **ASSIGNMNET STATEMENT** | **D-O-A** | **D-O-S** | **SIGNATURE** |
| 1 | Develop a client-server application using UDP where the client will send principal, rate of interest and number of years to the server and the server will calculate the interest and send the result to the client. The client will display the interest amount. |  |  |  |
| 2 | Develop a client-Server application using UDP where the client will send a decimal integer to the server and the server will calculate the sum of its even positioned digits and send back the result to the client. The client will display the result. [Example: Input: 1248, Output: 2+8=10] |  |  |  |
| 3 | Develop a client-server application using UDP where the client will send a filename to the server and the server will display the content of the file, otherwise display a message “file does not exist on the server”. Note: The file is situated at the server side. |  |  |  |
| 4 | Develop a client-server application using TCP where the client will send two operands and an operator to the server in operand1 operator operand2 format and the server will calculate the result and display it. Allowed operators are +, -, \*, /, %. |  |  |  |
| 5 | Develop a client-Server application using TCP where the client will send two words to the server and the server will check whether same characters are present in both the words and they occur same number of times (irrespective of position of the characters) and send back the result to the client. The client will display the result. [Example: Input: listen silent, Output: Check condition satisfied.] |  |  |  |
| 6 | Develop a client-Server application using TCP where the client will send two file names to the server and the server will copy the content of the first file into the second file. After copying is done, the server will send an appropriate message to the client. The second file should be initially empty. Assume that both files are present at the server. |  |  |  |
| 7 | Develop a client-server application using TCP where the client sends the name of a text file and the size of the block of data to the server, the server checks the availability of the text file and if the text file is available, sends the text file content to the client as per the input block size after reading the file contents. After sending the whole file, display an appropriate message. |  |  |  |

**Q1. Develop a client-server application using UDP where the client will send principal, rate of interest and number of years to the server and the server will calculate the interest and send the result to the client. The client will display the interest amount.**

**Client Side Program:**

import socket

def client\_program():

    # Get the hostname of the machine

    host = socket.gethostname()

    port = 5000

    serveraddress = (host, port)

    # Create a UDP socket

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

    print("UDP Client running.....")

    while True:

        # Take input from user

        a = input("Enter the Amount: ")

        # If user wants to exit

        if a.lower().strip() == "exit":

            client\_socket.sendto("exit".encode(), serveraddress)  # notify server

            print("Client shutting down.")

            break

        b = input("Enter the Rate: ")

        c = input("Enter the Time in years: ")

        # Combine all values into string

        message = f"{a},{b},{c}"

        # Send the encoded message to the server

        client\_socket.sendto(message.encode(), serveraddress)

      # Receive reply (calculated interest) from server

        data, address = client\_socket.recvfrom(1024)

        # Print the result received from the server

        print("Received from server:", data.decode())

        print("\n")

    # Close the client socket after communication is done

    client\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    client\_program()

**Server Side Program:**

import socket

def server\_program():

    host = socket.gethostname()   # local host

    port = 5000                   # Port number

    print("->" + host)

    # Create a UDP socket

    server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

    # Bind the socket with the host and port

    server\_socket.bind((host, port))

    print("UDP Server running.....")

    while True:

        # Receive data from client

        data, address = server\_socket.recvfrom(1024)

        # Decode the data into string

        message = data.decode().strip()

        print(message)

        # If message is "exit", shut down the server

        if message.lower() == "exit":

            print("Server shutting down")

            break

        # Split the received string

        a, b, c = map(str, message.split(","))

        # Convert extracted string values to float

        principal = float(a)   # Principal Amount

        rate = float(b)        # Rate of Interest

        time = float(c)        # Time

        # Calculate Simple Interest

        result = (principal \* rate \* time) / 100

        interest = str(result)

        # Print the interest value

        print("The interest is: ", interest)

        print("\n")

        # Send the calculated interest to the client

        server\_socket.sendto(interest.encode(), address)

    server\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    server\_program()

**Output:**

**Client side Output:**

UDP Client running.....

Enter the Amount: 12500

Enter the Rate: 12

Enter the Time in years: 10

Received from server: 15000.0

Enter the Amount: 250000

Enter the Rate: 17

Enter the Time in years: 25

Received from server: 1062500.0

Enter the Amount: exit

Client shutting down.

**Server side Output:**

->SUKDEB

UDP Server running.....

12500,12,10

The interest is:  15000.0

250000,17,25

The interest is:  1062500.0

exit

Server shutting down

**Q2. Develop a client-Server application using UDP where the client will send a decimal integer to the server and the server will calculate the sum of its even positioned digits and send back the result to the client. The client will display the result. [Example: Input: 1248, Output: 2+8=10]**

**Client Side Program:**

import socket

def client\_program():

    # Get the hostname of the machine

    host = socket.gethostname()   # local host

    # Define the same port as the server

    port = 5000

    #server address (host, port)

    serveraddress = (host, port)

    # Print host and address

    print(host)

    print(serveraddress)

    # Create a UDP socket

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

    print("UDP Client running.....")

    while True:

        # Take input number from user

        a = input("Write the number: ")

        if a.lower().strip() == "exit":

            print("Client shutting down.")

            client\_socket.sendto(a.encode(), serveraddress)

            break

        message = a

        # Send the number to the server

        client\_socket.sendto(message.encode(), serveraddress)

        # Receive processed result

        data, address = client\_socket.recvfrom(1024)

        # Print the result received from server

        print("Received the answer from the server: ", data.decode())

        print("\n")

    client\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    client\_program()

**Server Side Program:**

import socket

def server\_program():

    # Get the hostname of the machine

    host = socket.gethostname()  # local host

    port = 5000                  # Port number

    print("->" + host)

    # Create a UDP socket

    server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

    # Bind the socket with the host and port

    server\_socket.bind((host, port))

    print("UDP Server running.....")

    while True:

        # Receive data from client

        data, address = server\_socket.recvfrom(1024)

        # Print the data received from client

        print("data: ", data)

        # Print the address of the client

        print("Address: ", address)

        # Decode the data into string

        message = data.decode()

        if message.lower().strip() == "exit":

            print("Server shutting down.")

            break

        # Convert received message into integer

        a = int(message)

        # Take absolute value

        b = abs(a)

        s = 0

        # Convert number into string

        num = str(b)

        for i in range(1, len(num)):

            if (i + 1) % 2 == 0:

                s = s + int(num[i])

        print(s)

        # Convert result into string

        result = str(s)

        # Print the output

        print("The output is: ", result)

        print("\n")

        # Send the result back to the client

        server\_socket.sendto(result.encode(), address)

    # Close the server

    server\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    server\_program()

**Output:**

**Client Side Output:**

SUKDEB

('SUKDEB', 5000)

UDP Client running.....

Write the number: 1248

Received the answer from the server:  10

Write the number: -581

Received the answer from the server:  8

Write the number: 4321

Received the answer from the server:  4

Write the number: exit

Client shutting down.

**Server Side Output:**

*->*SUKDEB

UDP Server running.....

data:  b'1248'

Address:  ('10.142.105.6', 53417)

10

The output is:  10

data:  b'-581'

Address:  ('10.142.105.6', 53417)

8

The output is:  8

data:  b'4321'

Address:  ('10.142.105.6', 53417)

4

The output is:  4

data:  b'exit'

Address:  ('10.142.105.6', 53417)

Server shutting down.

**Q3. Develop a client-server application using UDP where the client will send a filename to the server and the server will display the content of the file, otherwise display a message “file does not exist on the server”. Note: The file is situated at the server side.**

**Client Side Program:**

import socket

def client\_program():

    host = socket.gethostname() #local host

    port = 5000

    serveraddress = (host,port)

    print(host)

    print(serveraddress)

    #  create socket

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

    # takes input

    while True:

        a = input("write the filename: ")

        message = a   #Combine into one string

        client\_socket.sendto(message.encode(),serveraddress) #sends the data

        if a.lower().strip() == "exit":  # notify server to stop

            print("Client exiting...")

            break

        data, address = client\_socket.recvfrom(1024) #receive the data

        print("Receive the answer from the server: ", data.decode()) #print the result

    client\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    client\_program()

**Server Side Program:**

import socket

import os

def server\_program():

    host = socket.gethostname() #local host

    port = 5000

    print("->" + host)

    # socket create

    server\_socket = socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM)

    # connection

    server\_socket.bind((host, port))

    print("UDP Server running.....")

    while True:

        data,address = server\_socket.recvfrom(1024)

        print("data: ", data)

        print("Address: ", address)

        file = data.decode()

        print(file)

        if file.lower().strip() == "exit":

            print("Server shutting down.")

            break

        if os.path.exists(file):

            f=open(file, "r")

            content = f.read()

            print("the content of this file: " + content)

            server\_socket.sendto(content.encode(), address)

        else:

            message = "File does not exist on the server"

            server\_socket.sendto(message.encode(), address)

    server\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    server\_program()

**Output:**

**Client Side Output:**

SUKDEB

('SUKDEB', 5000)

write the filename: test1.txt

Receive the answer from the server:  hi there this  a test1 file content....

write the filename: text.txt

Receive the answer from the server:  File does not exist on the server

write the filename: exit

Client exiting...

**Server Side Output:**

->SUKDEB

UDP Server running.....

data: b'test1.txt'

Address: ('10.142.105.6', 59026)

test1.txt

the content of this file: hi there this a test1 file content....

data: b'text.txt'

Address: ('10.142.105.6', 59026)

text.txt

data: b'exit'

Address: ('10.142.105.6', 59026)

exit

Server shutting down.

**Q4. Develop a client-server application using TCP where the client will send two operands and an operator to the server in operand1 operator operand2 format and the server will calculate the result and display it. Allowed operators are +, -, \*, /, %.**

**Client Side Program:**

import socket

def client\_program():

    host = socket.gethostname()

    port = 5000

    print("Client is starting on host:", host)

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

    client\_socket.connect((host, port))

    while True:

        a = input("Enter the number1: ")

        if a.lower().strip() == "exit":

            # Inform server also before shutting down

            client\_socket.send("exit".encode())

            print("Client exiting...")

            break

        b = input("Enter the operator: ")

        c = input("Enter the number2: ")

        # Combine into one string

        message = f"{a},{b},{c}"

        client\_socket.send(message.encode())

        # Receive result from server

        data = client\_socket.recv(1024).decode()

        print("Result show from the client side: " + data)

        print("\n")

    client\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    client\_program()

**Server Side Program:**

import socket

def server\_program():

    # Get the hostname of the machine

    host = socket.gethostname()

    port = 5000   # Port number for the server

    print("Server is starting on host:", host)

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

    # Bind socket with host and port

    server\_socket.bind((host, port))

    # Server can listen for incoming client requests

    server\_socket.listen()

    # Accept client connection

    connection, address = server\_socket.accept()

    print("Connected to client:", str(address))

    while True:

        # Receive data from client

        data = connection.recv(1024).decode()

        if not data:

            break

        if data.lower().strip() == "exit":

            print("Exit command received. Server shutting down.")

            break

        # Split the received message

        a, b, c = map(str, data.split(","))

        a = int(a)

        c = int(c)

        try:

            if b == "+":

                res = a + c

            elif b == "-":

                res = a - c

            elif b == "\*":

                res = a \* c

            elif b == "/":

                if c == 0:

                    res = "Error beacuase Division by zero"

                else:

                    res = a / c

            elif b == "%":

                if c == 0:

                    res = "Error beacuase Modulo by zero"

                else:

                    res = a % c

            else:

                res = "Invalid operator"

        except Exception as e:

            print(e)

        # Convert result into string

        message = str(res)

        # Show result

        print("Result calculated on server:", message)

        print("\n")

        # Send result back to client

        connection.send(message.encode())

    # Close the server socket

    server\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    server\_program()

**Output:**

**Client Side Output:**

Client is starting on host: SUKDEB

Enter the number1: 26

Enter the operator: +

Enter the number2: 67

Result show from the client side: 93

Enter the number1: 65

Enter the operator: -

Enter the number2: 34

Result show from the client side: 31

Enter the number1: 56

Enter the operator: \*

Enter the number2: 21

Result show from the client side: 1176

Enter the number1: 56

Enter the operator: /

Enter the number2: 2

Result show from the client side: 28.0

Enter the number1: 45

Enter the operator: /

Enter the number2: 0

Result show from the client side: Error beacuase Division by zero

Enter the number1: 26

Enter the operator: %

Enter the number2: 0

Result show from the client side: Error beacuase Modulo by zero

Enter the number1: 34

Enter the operator:

Enter the number2: 7

Result show from the client side: Invalid operator

Enter the number1: 78

Enter the operator: &

Enter the number2: 5

Result show from the client side: Invalid operator

Enter the number1: exit

Client exiting...

**Server Side Output:**

Server is starting on host: SUKDEB

Connected to client: ('10.142.105.6', 53580)

Result calculated on server: 93

Result calculated on server: 31

Result calculated on server: 1176

Result calculated on server: 28.0

Result calculated on server: Error beacuase Division by zero

Result calculated on server: Error beacuase Modulo by zero

Result calculated on server: Invalid operator

Result calculated on server: Invalid operator

Exit command received. Server shutting down.

**Q5. Develop a client-Server application using TCP where the client will send two words to the server and the server will check whether same characters are present in both the words and they occur same number of times (irrespective of position of the characters) and send back the result to the client. The client will display the result. [Example: Input: listen silent, Output: Check condition satisfied.]**

**Client Side Program:**

import socket

def client\_program():

    host=socket.gethostname()  # get the hostname

    port=5000 # socket server port number

    print("Client is starting on host:", host)

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

    client\_socket.connect((host,port)) # connects to the server

    while True:

        a=input("enter the first word: ")

        if not a:

            print("Client exiting...")

            break

        b=input("enter the second word: ")

        message=f"{a},{b}"

        client\_socket.send(message.encode()) # sends message

        data=client\_socket.recv(1024).decode()  # receives response

        print("The condition check from the server side: " + data) #print

        print("\n")

    client\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    client\_program()

**Server Side Program:**

import socket

def server\_program():

    host=socket.gethostname()  # get the hostname

    port=5000  # initiates port no

    print("Server is starting on host:", host)

    server\_socket=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) # creates server side socket

    server\_socket.bind((host,port))  # binds host address and port together

    server\_socket.listen() # how many client the server can listen

    conn,address=server\_socket.accept() # accepts new connection

    print("the connection cllient: " + str(address))

    while True:

        data=conn.recv(1024).decode()

        if not data:

            print("Server shutting down.")

            break

        a,b=map(str,data.split(","))  #data split

        c=""

        if sorted(a) == sorted(b):  #check the condition

            c="Check condition satisfied"

        else:

            c="Check condition not satisfied"

        print("The condition check result: " + c)

        print("\n")

        message=c

        conn.send(message.encode()) #send to the client

    conn.close()

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

    server\_program()

**Output:**

**Client Side Output:**

Client is starting on host: SUKDEB

enter the first word: listen

enter the second word: silent

The condition check from the server side: Check condition satisfied

enter the first word: sukdeb

enter the second word: bedkus

The condition check from the server side: Check condition satisfied

enter the first word: hello

enter the second word: hi

The condition check from the server side: Check condition not satisfied

enter the first word:

Client exiting...

**Server Side Output:**

Server is starting on host: SUKDEB

the connection cllient: ('10.142.105.6', 58806)

The condition check result: Check condition satisfied

The condition check result: Check condition satisfied

The condition check result: Check condition not satisfied

Server shutting down.

**Q6. Develop a client-Server application using TCP where the client will send two file names to the server and the server will copy the content of the first file into the second file. After copying is done, the server will send an appropriate message to the client. The second file should be initially empty. Assume that both files are present at the server.**

**Client Side Program:**

import socket

def client\_program():

    host=socket.gethostname()  # get the hostname

    port=5000 # socket server port number

    print("Client is starting on host:", host)

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

    client\_socket.connect((host,port)) # connects to the server

    while True:

        # takes input

        a=input("enter the first file: ")

        if not a:

            print("client exiting.....")

            break

        b=input("enter the second file: ")

        message=f"{a},{b}"

        client\_socket.send(message.encode()) # sends message

        data=client\_socket.recv(1024).decode()  # receives response

        print("The condition check from the client side: " + data) #print

        print("\n")

    client\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    client\_program()

**Server Side Program:**

import socket

import os

def server\_program():

    host = socket.gethostname()   # get local host name

    port = 5000                   # choose port number

    print("Server is starting on host:", host)

    # create server socket )

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

    server\_socket.bind((host, port))   # bind host and port

    server\_socket.listen()             # listen for client connections

    conn, address = server\_socket.accept()   # accept connection

    print("Connected client: " + str(address))

    while True:

        data = conn.recv(1024).decode()      # receive filenames from client

        if not data:

            print("Server shutting down.")

            break

        file1, file2 = map(str, data.split(","))  # split filenames

        print(file1, file2)

        try:

            if os.path.exists(file1) and os.path.exists(file2):   # check both files

                f = open(file1, "r")     # open source file

                content = f.read()       # read content

                f.close()

                e = open(file2, "w")     # open destination file

                e.write(content)         # write content

                e.close()

                c = "Content copied successfully!!"

            else:

                c = "File not found"

        except Exception as e:

            print(e)

            c = "Error during copy"

        print("Server response: " + c)

        conn.send(c.encode())     # send response to client

    server\_socket.close()

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

    server\_program()

**Output:**

**Client side Output:**

Client is starting on host: SUKDEB

enter the first file: test1.txt

enter the second file: test2.txt

The condition check from the client side: Content copied successfully!!

enter the first file: file.txt

enter the second file: file2.txt

The condition check from the client side: File not found

enter the first file:

client exiting.....

**Server Side Output:**

Server is starting on host: SUKDEB

Connected client: ('192.168.0.104', 65520)

test1.txt test2.txt

Server response: Content copied successfully!!

file.txt file2.txt

Server response: File not found

Server shutting down.

**Q7. Develop a client-server application using TCP where the client sends the name of a text file and the size of the block of data to the server, the server checks the availability of the text file and if the text file is available, sends the text file content to the client as per the input block size after reading the file contents. After sending the whole file, display an appropriate message.**

**Client Side Program:**

import socket

def client\_program():

    host=socket.gethostname()

    port=5000

    print("Client is starting on host:", host)

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

    client\_socket.connect((host,port))

    print("client side start........")

    while True:

        filename = input("Enter the filename: ")

        if not filename:

            print("Client exiting.....")

            break

        block\_size = input("Enter block size (number of bytes): ")

        message=f"{filename},{block\_size}"

        client\_socket.send(message.encode())

        data=client\_socket.recv(1024).decode()

        print("The condition check from the server side: " + data)

        print("\n")

    client\_socket.close()

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

    client\_program()

**Server Side Program:**

import socket

import os

def server\_program():

    host=socket.gethostname()

    port=5000

    print("Server is starting on host:", host)

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

    server\_socket.bind((host,port))

    server\_socket.listen()

    conn,address=server\_socket.accept()

    print("server side start........")

    print("the conncetion client: " + str(address))

    while True:

        data=conn.recv(1024).decode()

        if not data:

            print("Server shutting down.")

            break

        print(data)

        file,block\_size=map(str,data.split(","))

        print(file)

        try:

            if(os.path.exists(file)):

                f=open(file,"r")

                content=f.read(int(block\_size))

                f.close()

        except Exception as e:

            print(e)

        print("The content: " + content)

        print("\n")

        conn.send(content.encode())

    server\_socket.close()

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

    server\_program()

**Output:**

**Client Side Output:**

Client is starting on host: SUKDEB

client side start........

Enter the filename: test1.txt

Enter block size (number of bytes): 5

The condition check from the server side: hi th

Enter the filename: test1.txt

Enter block size (number of bytes): 7

The condition check from the server side: hi ther

Enter the filename:

Client exiting.....

**Server Side Output:**

Server is starting on host: SUKDEB

server side start........

the conncetion client: ('10.142.105.6', 64501)

test1.txt,5

test1.txt

The content: hi th

test1.txt,7

test1.txt

The content: hi ther

Server shutting down.