OUTPUTS

NETWORKS LAB

106119055

JEREMIAH THOMAS

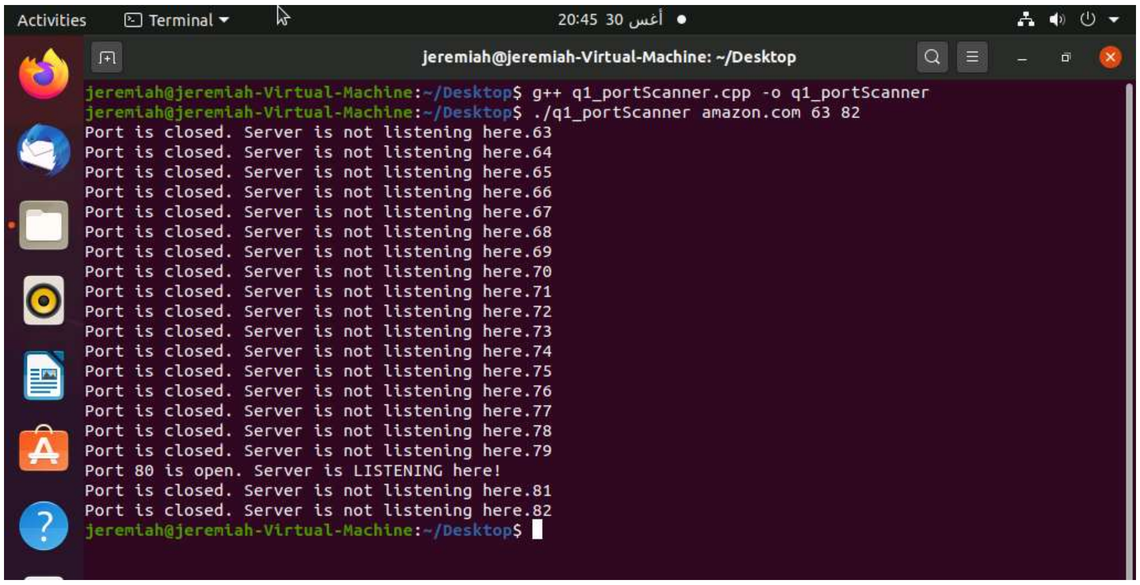
Do note:-

- The program files, as well as the executables have been attached in a zip folder in the assignment tab.
- Also, do not forget to run the program in a **linux** environment as it does not work with any other OS.

Q1.

- The code that I've written is capable of handling any valid port no input. For the purpose of illustration, ports from 63 to 82 were passed as command line arguments as seen below: -

q1_portScanner.cpp



```
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ g++ q1_portScanner.cpp -o q1_portScanner
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ ./q1_portScanner amazon.com 63 82
Port is closed. Server is not listening here.63
Port is closed. Server is not listening here.64
Port is closed. Server is not listening here.65
Port is closed. Server is not listening here.66
Port is closed. Server is not listening here.67
Port is closed. Server is not listening here.68
Port is closed. Server is not listening here.69
Port is closed. Server is not listening here.70
Port is closed. Server is not listening here.71
Port is closed. Server is not listening here.72
Port is closed. Server is not listening here.73
Port is closed. Server is not listening here.74
Port is closed. Server is not listening here.75
Port is closed. Server is not listening here.76
Port is closed. Server is not listening here.77
Port is closed. Server is not listening here.78
Port is closed. Server is not listening here.79
Port 80 is open. Server is LISTENING here!
Port is closed. Server is not listening here.81
Port is closed. Server is not listening here.82
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ 
```

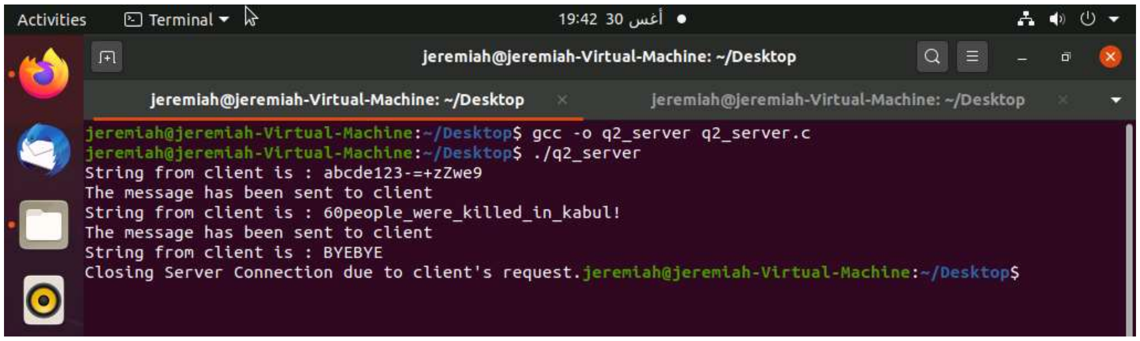
(DO SCROLL DOWN)

Q2.

- The encoding string function exists in the server side and encodes strings from the client with the rules as given in the question.
- Client and server communication terminates when client sends server the string -> **"BYEBYE"**
- To test the functionality of my code, I took three strings of varied types to test for edge cases.

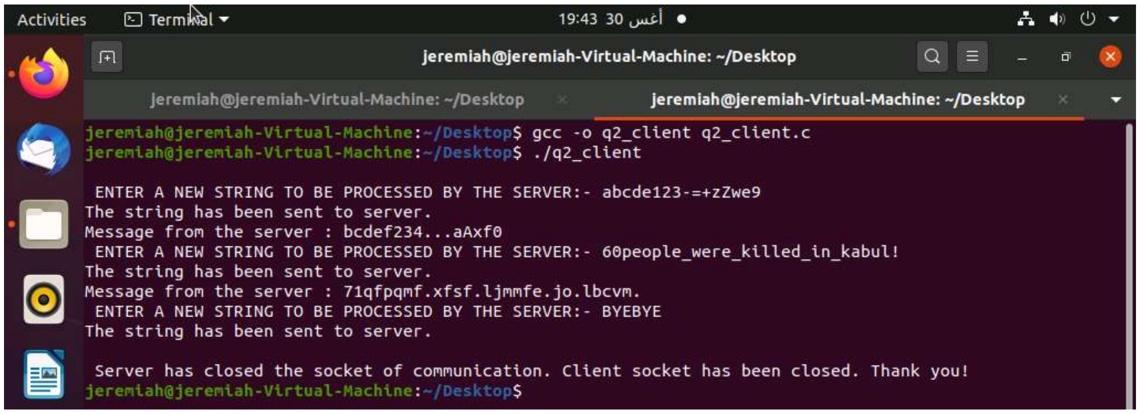
Kindly find the results below: -

- SERVER:-



```
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ gcc -o q2_server q2_server.c
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ ./q2_server
String from client is : abcde123-=+zZwe9
The message has been sent to client
String from client is : 60people_were_killed_in_kabul!
The message has been sent to client
String from client is : BYEBYE
Closing Server Connection due to client's request.jeremiah@jeremiah-Virtual-Machine:~/Desktop$
```

- CLIENT:-



```
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ gcc -o q2_client q2_client.c
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ ./q2_client

 ENTER A NEW STRING TO BE PROCESSED BY THE SERVER:- abcde123-=+zZwe9
The string has been sent to server.
Message from the server : bcdef234...aAxf0
 ENTER A NEW STRING TO BE PROCESSED BY THE SERVER:- 60people_were_killed_in_kabul!
The string has been sent to server.
Message from the server : 71qfpqmf.xfsf.ljmmfe.jo.lbcvm.
 ENTER A NEW STRING TO BE PROCESSED BY THE SERVER:- BYEBYE
The string has been sent to server.

 Server has closed the socket of communication. Client socket has been closed. Thank you!
jeremiah@jeremiah-Virtual-Machine:~/Desktop$
```

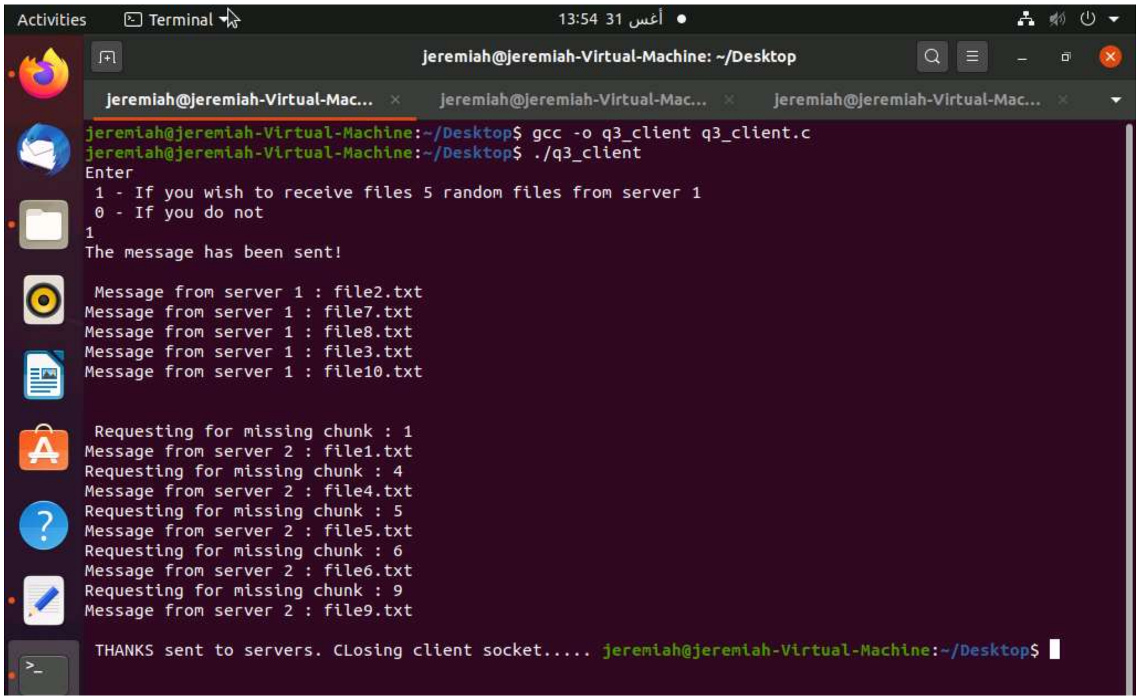
Q3.

- The original file has been partitioned into **10 separate chunks** (file1.txt, file2.txt …. file10.txt)

- Also, two sockets have been created on PORT **3000** and **3001** for communication between client and server1(**sock**) and server2(**sock2**) respectively.
- The client requests for the original file from server1. Server 1 in return partially sends **5 chunks chosen randomly** to the client.
- The client receives the chunks and uses a flag array -> **chunks[]** and a function **findChunkNo(char[])** to figure out which chunk server1 sent it and accordingly **find the missing chunks**.
- The client then proceeds to request for the missing chunks from server 2, one at a time as prescribed in the question
- Once the client has received all 10 chunks, it sends a **"THANKS"** message to the servers, giving them the signal that can close the sockets.
- I've tested my client and servers with a sample test case.

Kindly find the results attached below.

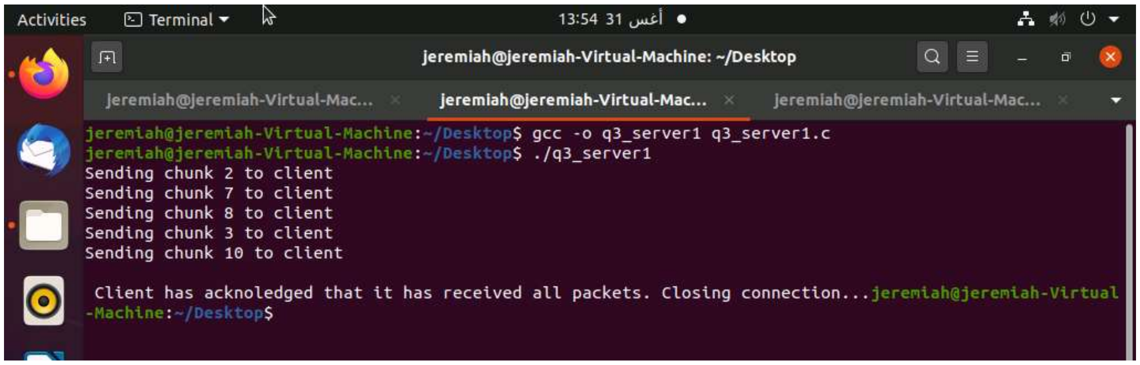
- CLIENT:-



```
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ gcc -o q3_client q3_client.c
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ ./q3_client
Enter
 1 - If you wish to receive files 5 random files from server 1
 0 - If you do not
1
The message has been sent!

 Message from server 1 : file2.txt
Message from server 1 : file7.txt
Message from server 1 : file8.txt
Message from server 1 : file3.txt
Message from server 1 : file10.txt


 Requesting for missing chunk : 1
Message from server 2 : file1.txt
Requesting for missing chunk : 4
Message from server 2 : file4.txt
Requesting for missing chunk : 5
Message from server 2 : file5.txt
Requesting for missing chunk : 6
Message from server 2 : file6.txt
Requesting for missing chunk : 9
Message from server 2 : file9.txt

 THANKS sent to servers. CLosing client socket..... jeremiah@jeremiah-Virtual-Machine:~/Desktop$
```

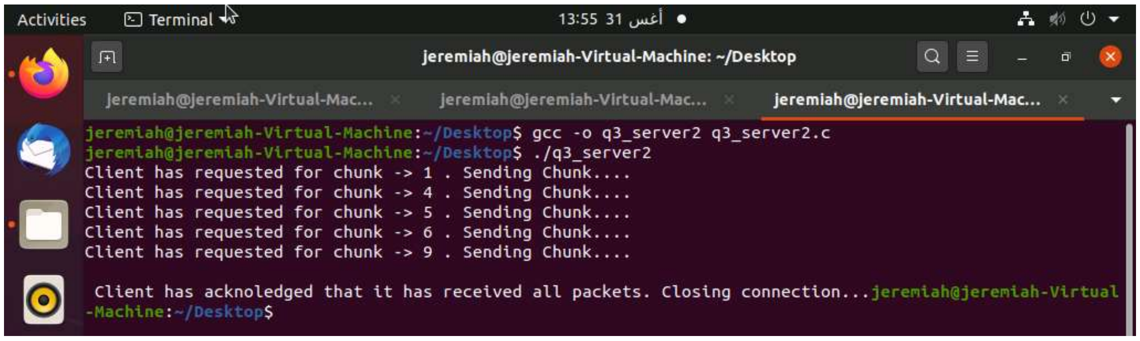
(DO SCROLL DOWN)

- <u>SERVER 1:-</u>


Activities
Terminal
13:54 31 أغس
jeremiah@jeremiah-Virtual-Machine: ~/Desktop
jeremiah@jeremiah-Virtual-Mac...
jeremiah@jeremiah-Virtual-Mac...
jeremiah@jeremiah-Virtual-Mac...

```
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ gcc -o q3_server1 q3_server1.c
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ ./q3_server1
Sending chunk 2 to client
Sending chunk 7 to client
Sending chunk 8 to client
Sending chunk 3 to client
Sending chunk 10 to client

 Client has acknoledged that it has received all packets. Closing connection...jeremiah@jeremiah-Virtual
-Machine:~/Desktop$
```

- <u>SERVER 2:-</u>


Activities
Terminal
13:55 31 أغس
jeremiah@jeremiah-Virtual-Machine: ~/Desktop
jeremiah@jeremiah-Virtual-Mac...
jeremiah@jeremiah-Virtual-Mac...
jeremiah@jeremiah-Virtual-Mac...

```
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ gcc -o q3_server2 q3_server2.c
jeremiah@jeremiah-Virtual-Machine:~/Desktop$ ./q3_server2
Client has requested for chunk -> 1 . Sending Chunk....
Client has requested for chunk -> 4 . Sending Chunk....
Client has requested for chunk -> 5 . Sending Chunk....
Client has requested for chunk -> 6 . Sending Chunk....
Client has requested for chunk -> 9 . Sending Chunk....

 Client has acknoledged that it has received all packets. Closing connection...jeremiah@jeremiah-Virtual
-Machine:~/Desktop$
```

THANK YOU!