COMPUTER NETWORKS LABORATORY

ASSIGNMENT 7

NAME: Abhishek Rathod
ENR. NO. : 17114004
BATCH: CS1

Github link :- https://github.com/rathodabhishek36/CSN-361-Computer-Networks-Laboratory

Q.1) Transmit a binary message (from a sender to a receiver) using socket programming in C and report whether the received msg is correct or not; using the following error detection algorithms: 1. Single Parity Check 2. Two-dimensional Parity Check 3. Checksum 4. Cyclic Redundancy Check (CRC)

Algorithms :- One socket listens on a particular port at an IP(IPv4 type used here), while other socket reaches out to the other to form a connection with the first socket. Server corresponds to the listener socket while client reaches out to the server.
Created two C files. One for the server and another for the client socket. Connected them using TCP protocol and IPv4 address.

In the server, we ask the user about the preference of algorithm to be used and then set the error in the message to send to the client.

1. Single Parity Check
Blocks of data from the source are subjected to a check bit or parity bit generator form, where a parity of :

- 1 is added to the block if it contains odd number of 1’s, and
- 0 is added if it contains even number of 1’s

This scheme makes the total number of 1’s even, that is why it is called even parity checking.

2. Two-dimensional Parity check
Parity check bits are calculated for each row, which is equivalent to a simple parity check bit. Parity check bits are also calculated for all columns, then both are sent along with the data. At the receiving end these are compared with the parity bits calculated on the received data.

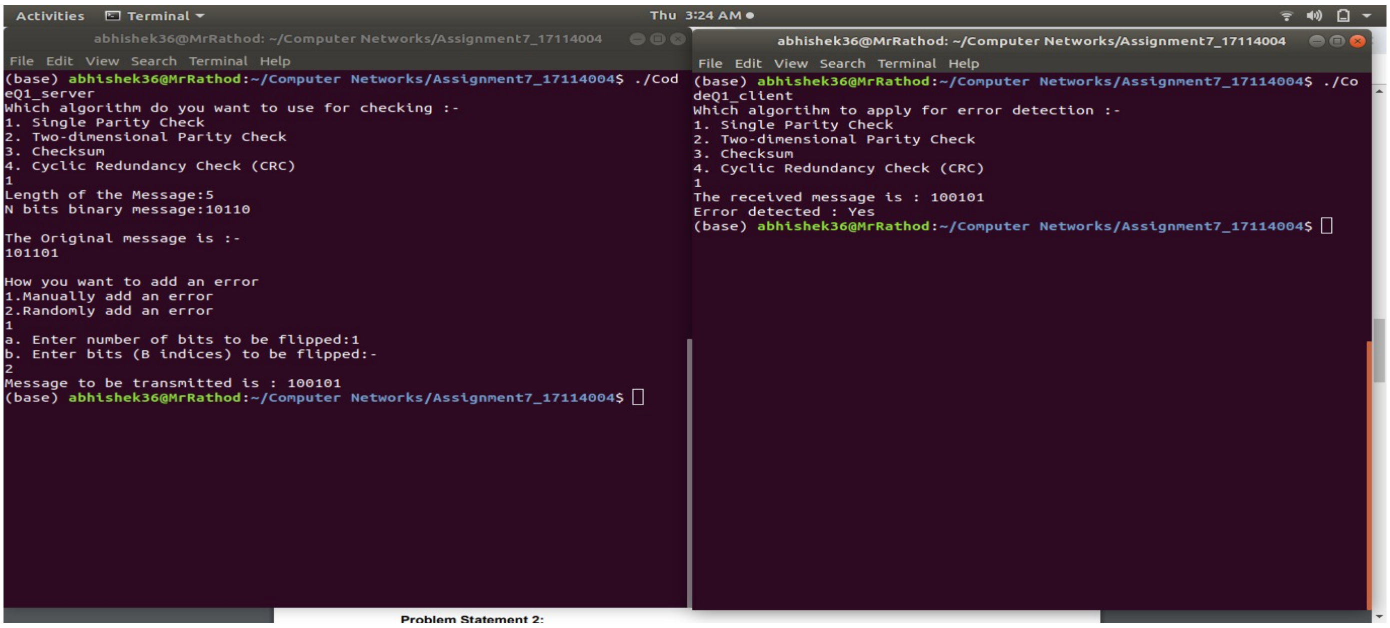
3. Checksum

In checksum error detection scheme, the data is divided into k segments each of m bits. In the sender's end the segments are added using 1's complement arithmetic to get the sum. The sum is complemented to get the checksum.The checksum segment is sent along with the data segments. At the receiver's end, all received segments are added using 1's complement arithmetic to get the sum. The sum is complemented.If the result is zero, the received data is accepted; otherwise discarded.

4. Cyclic redundancy check (CRC)

In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.

Screenshots :-



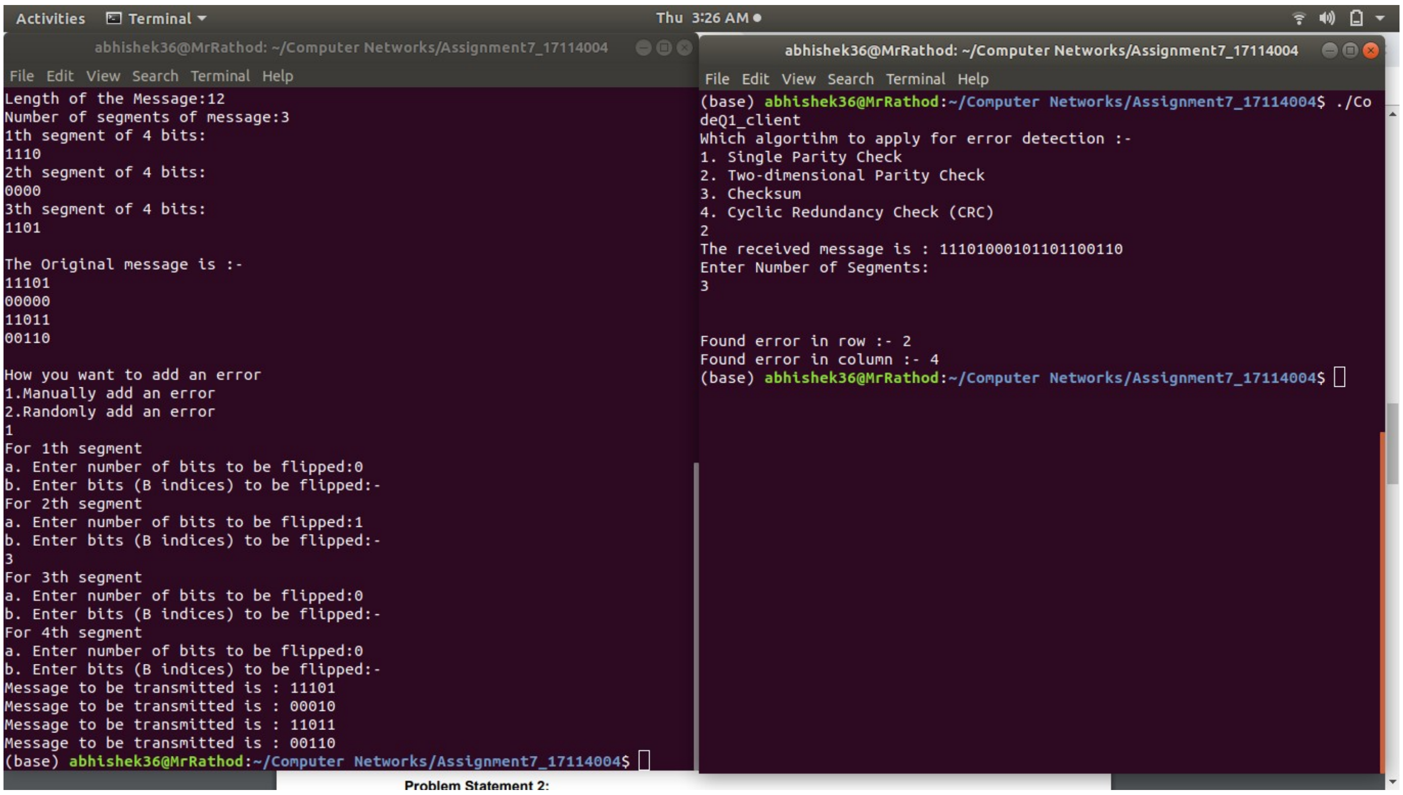
```
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$ ./Cod
eQ1_server
Which algorithm do you want to use for checking :-
1. Single Parity Check
2. Two-dimensional Parity Check
3. Checksum
4. Cyclic Redundancy Check (CRC)
1
Length of the Message:5
N bits binary message:10110

The Original message is :-
101101

How you want to add an error
1.Manually add an error
2.Randomly add an error
1
a. Enter number of bits to be flipped:1
b. Enter bits (B indices) to be flipped:-
2
Message to be transmitted is : 100101
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$
```

```
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$ ./Co
deQ1_client
Which algortihm to apply for error detection :-
1. Single Parity Check
2. Two-dimensional Parity Check
3. Checksum
4. Cyclic Redundancy Check (CRC)
1
The received message is : 100101
Error detected : Yes
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$
```

Activities
Terminal
Thu 3:24 AM
abhishek36@MrRathod: ~/Computer Networks/Assignment7_17114004
File Edit View Search Terminal Help
Problem Statement 2:


Activities
Terminal
Thu 3:26 AM
abhishek36@MrRathod: ~/Computer Networks/Assignment7_17114004
File Edit View Search Terminal Help
abhishek36@MrRathod: ~/Computer Networks/Assignment7_17114004
File Edit View Search Terminal Help
Problem Statement 2:

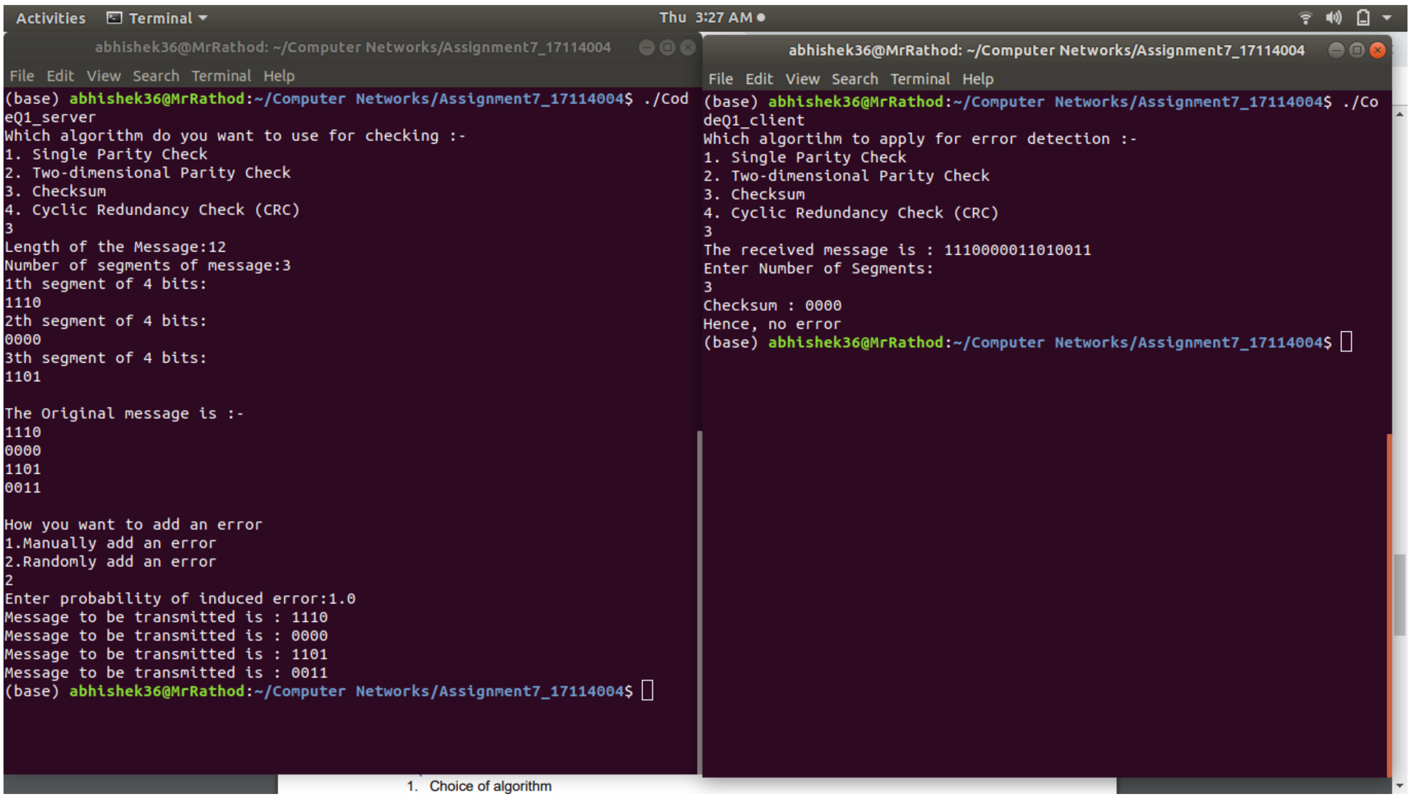
```
Length of the Message:12
Number of segments of message:3
1th segment of 4 bits:
1110
2th segment of 4 bits:
0000
3th segment of 4 bits:
1101

The Original message is :-
11101
00000
11011
00110

How you want to add an error
1.Manually add an error
2.Randomly add an error
1
For 1th segment
a. Enter number of bits to be flipped:0
b. Enter bits (B indices) to be flipped:-
For 2th segment
a. Enter number of bits to be flipped:1
b. Enter bits (B indices) to be flipped:-
3
For 3th segment
a. Enter number of bits to be flipped:0
b. Enter bits (B indices) to be flipped:-
For 4th segment
a. Enter number of bits to be flipped:0
b. Enter bits (B indices) to be flipped:-
Message to be transmitted is : 11101
Message to be transmitted is : 00010
Message to be transmitted is : 11011
Message to be transmitted is : 00110
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$
```

```
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$ ./Co
deQ1_client
Which algortihm to apply for error detection :-
1. Single Parity Check
2. Two-dimensional Parity Check
3. Checksum
4. Cyclic Redundancy Check (CRC)
2
The received message is : 11101000101101100110
Enter Number of Segments:
3


Found error in row :- 2
Found error in column :- 4
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$
```

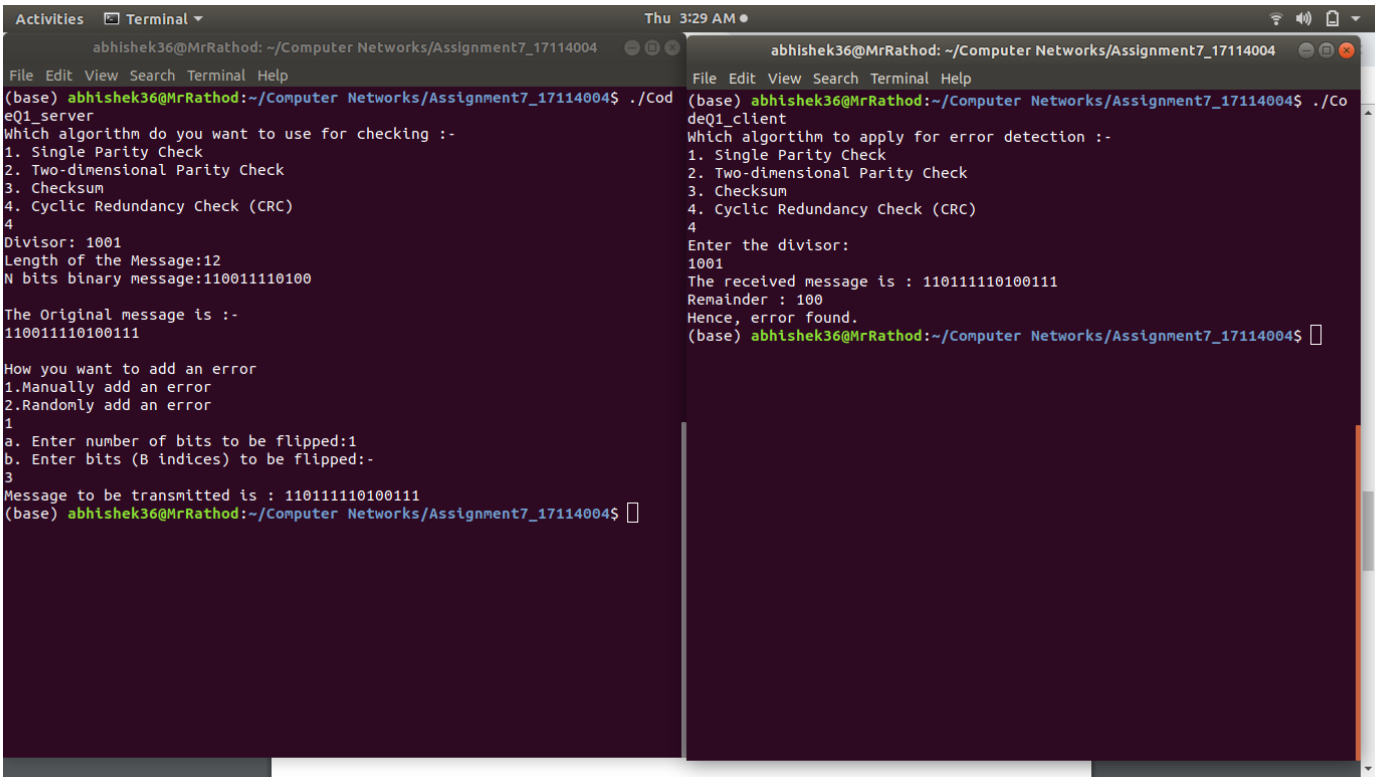

Activities
Terminal
Thu 3:27 AM
abhishek36@MrRathod: ~/Computer Networks/Assignment7_17114004
File Edit View Search Terminal Help
abhishek36@MrRathod: ~/Computer Networks/Assignment7_17114004
File Edit View Search Terminal Help
1. Choice of algorithm

```
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$ ./Cod
eQ1_server
Which algorithm do you want to use for checking :-
1. Single Parity Check
2. Two-dimensional Parity Check
3. Checksum
4. Cyclic Redundancy Check (CRC)
3
Length of the Message:12
Number of segments of message:3
1th segment of 4 bits:
1110
2th segment of 4 bits:
0000
3th segment of 4 bits:
1101

The Original message is :-
1110
0000
1101
0011

How you want to add an error
1.Manually add an error
2.Randomly add an error
2
Enter probability of induced error:1.0
Message to be transmitted is : 1110
Message to be transmitted is : 0000
Message to be transmitted is : 1101
Message to be transmitted is : 0011
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$
```

```
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$ ./Co
deQ1_client
Which algortihm to apply for error detection :-
1. Single Parity Check
2. Two-dimensional Parity Check
3. Checksum
4. Cyclic Redundancy Check (CRC)
3
The received message is : 1110000011010011
Enter Number of Segments:
3
Checksum : 0000
Hence, no error
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$
```


Activities
Terminal
Thu 3:29 AM
abhishek36@MrRathod: ~/Computer Networks/Assignment7_17114004
File Edit View Search Terminal Help
abhishek36@MrRathod: ~/Computer Networks/Assignment7_17114004
File Edit View Search Terminal Help

```
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$ ./Cod
eQ1_server
Which algorithm do you want to use for checking :-
1. Single Parity Check
2. Two-dimensional Parity Check
3. Checksum
4. Cyclic Redundancy Check (CRC)
4
Divisor: 1001
Length of the Message:12
N bits binary message:110011110100

The Original message is :-
110011110100111

How you want to add an error
1.Manually add an error
2.Randomly add an error
1
a. Enter number of bits to be flipped:1
b. Enter bits (B indices) to be flipped:-
3
Message to be transmitted is : 110111110100111
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$
```

```
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$ ./Co
deQ1_client
Which algorithm to apply for error detection :-
1. Single Parity Check
2. Two-dimensional Parity Check
3. Checksum
4. Cyclic Redundancy Check (CRC)
4
Enter the divisor:
1001
The received message is : 110111110100111
Remainder : 100
Hence, error found.
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$
```

Data structures used :-

1) Socket -

Socket creation:

sockfd: socket descriptor, an integer

struct sockaddr_in : structure to store internet addresses like IP address, port.

Bind:

int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

bind function binds the socket to the address and port number specified in addr.

Listen:

int listen(int sockfd, int backlog);

It puts the server socket in a passive mode, where it waits for the client to approach the server to make a connection. The backlog, defines the maximum length to which the queue of pending connections for sockfd may grow.

Accept:
int new_socket= accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
It extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket.

2) Client -

Socket connection:
same as that of server’s socket creation

Connect:
int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
The connect() system call connects the socket referred to by the file descriptor sockfd to the address specified by addr. Server’s address and port is specified in addr.

-> Also, character arrays and pointers were used to manipulate the data at the sender and client sites.

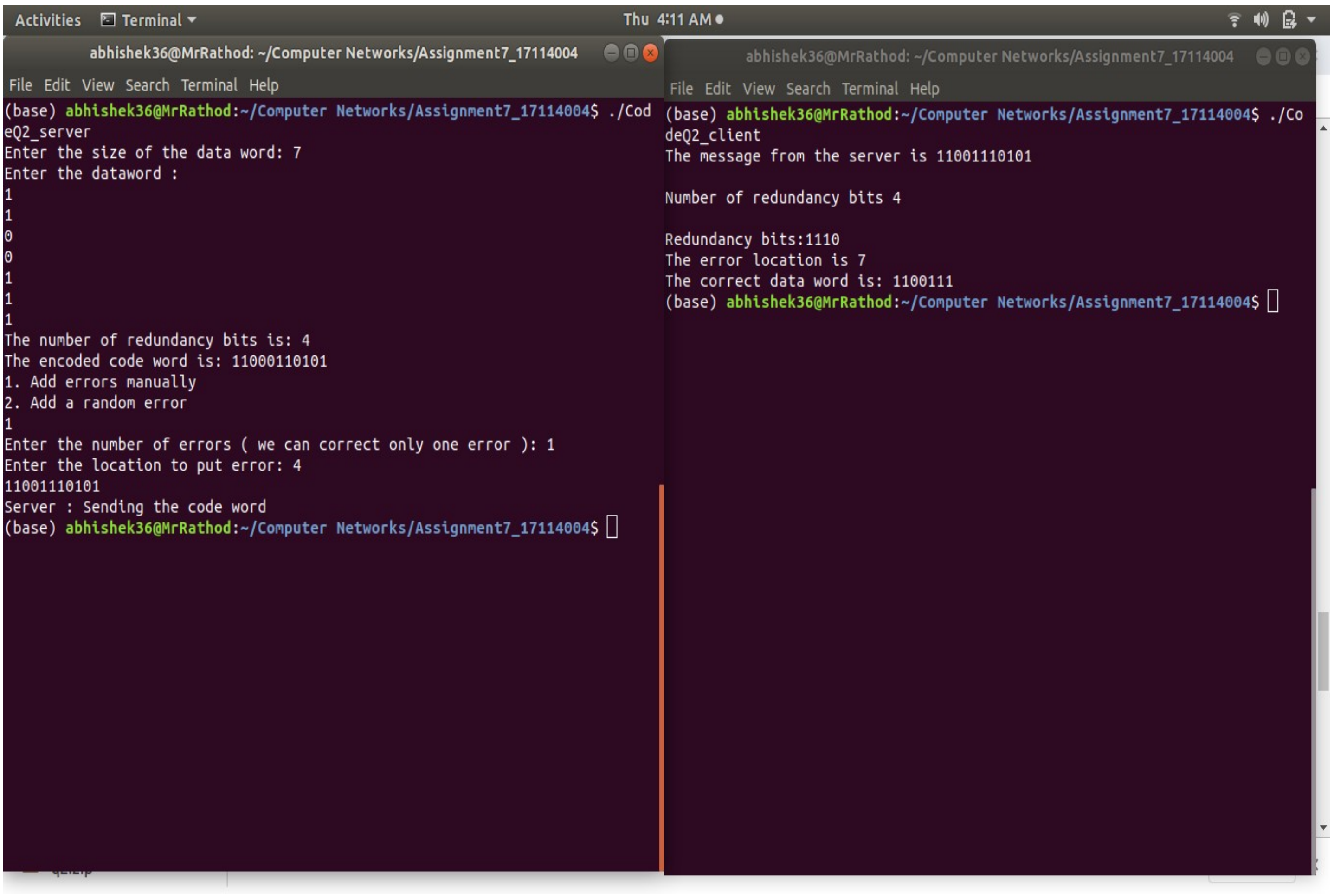
Question 2: Transmit a binary message (from a sender to a receiver) using socket programming in C. Using Hamming code detect and correct errors in the transmitted message.

Algorithm used:- One socket listens on a particular port at an IP(IPv4 type used here), while other socket reaches out to the other to form a connection with the first socket. Server

corresponds to the listener socket while client reaches out to the server.
Created two C files. One for the server and another for the client socket. Connected them using TCP protocol and IPv4 address.

Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver.
Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer.
A parity bit is a bit appended to a data of binary bits to ensure that the total number of 1’s in the data are even or odd. Parity bits are used for error detection.

Screenshots-


Activities
Terminal
Thu 4:11 AM
abhishek36@MrRathod: ~/Computer Networks/Assignment7_17114004
File Edit View Search Terminal Help
abhishek36@MrRathod: ~/Computer Networks/Assignment7_17114004
File Edit View Search Terminal Help

```
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$ ./Cod
eQ2_server
Enter the size of the data word: 7
Enter the dataword :
1
1
0
0
1
1
1
The number of redundancy bits is: 4
The encoded code word is: 11000110101
1. Add errors manually
2. Add a random error
1
Enter the number of errors ( we can correct only one error ): 1
Enter the location to put error: 4
11001110101
Server : Sending the code word
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$
```

```
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$ ./Co
deQ2_client
The message from the server is 11001110101

Number of redundancy bits 4

Redundancy bits:1110
The error location is 7
The correct data word is: 1100111
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$
```

Data structures used :

Character arrays and pointers were used to manipulate the data at the sender and client sites.

Socket creation:
sockfd: socket descriptor, an integer
struct sockaddr_in : structure to store internet addresses like IP address, port.

Bind:
int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
bind function binds the socket to the address and port number specified in addr.

Recvfrom:
int recvfrom(int socket, char *buffer, int length, int flags, struct sockaddr *address, int *address_length) ;
The recvfrom() function recieves data on a socket named by descriptor socket and stores it in a buffer. The recvfrom() function applies to any datagram socket, whether connected or unconnected.

Sendto:
int sendto(int socket, char *buffer, int length, int flags, struct sockaddr *address, int adddress_len) ;
The sendto() function sends data on the socket with descriptor socket. The sendto() call applies to either connected or unconnected sockets.

Question 3: Write a C++ program to compress a message (non-binary, can be anything like a text message or a code like hexadecimal, etc.) using the following data compression algorithm: 1. Huffman 2. Shannon-Fano .

Algorithm used:-

Huffman Coding also called as Huffman Encoding is a famous greedy algorithm that is used for the lossless compression of data.

- It uses variable length encoding where variable length codes are assigned to all the characters depending on how frequently they occur in the given text.
- The character which occurs most frequently gets the smallest code and the character which occurs least frequently gets the largest code.

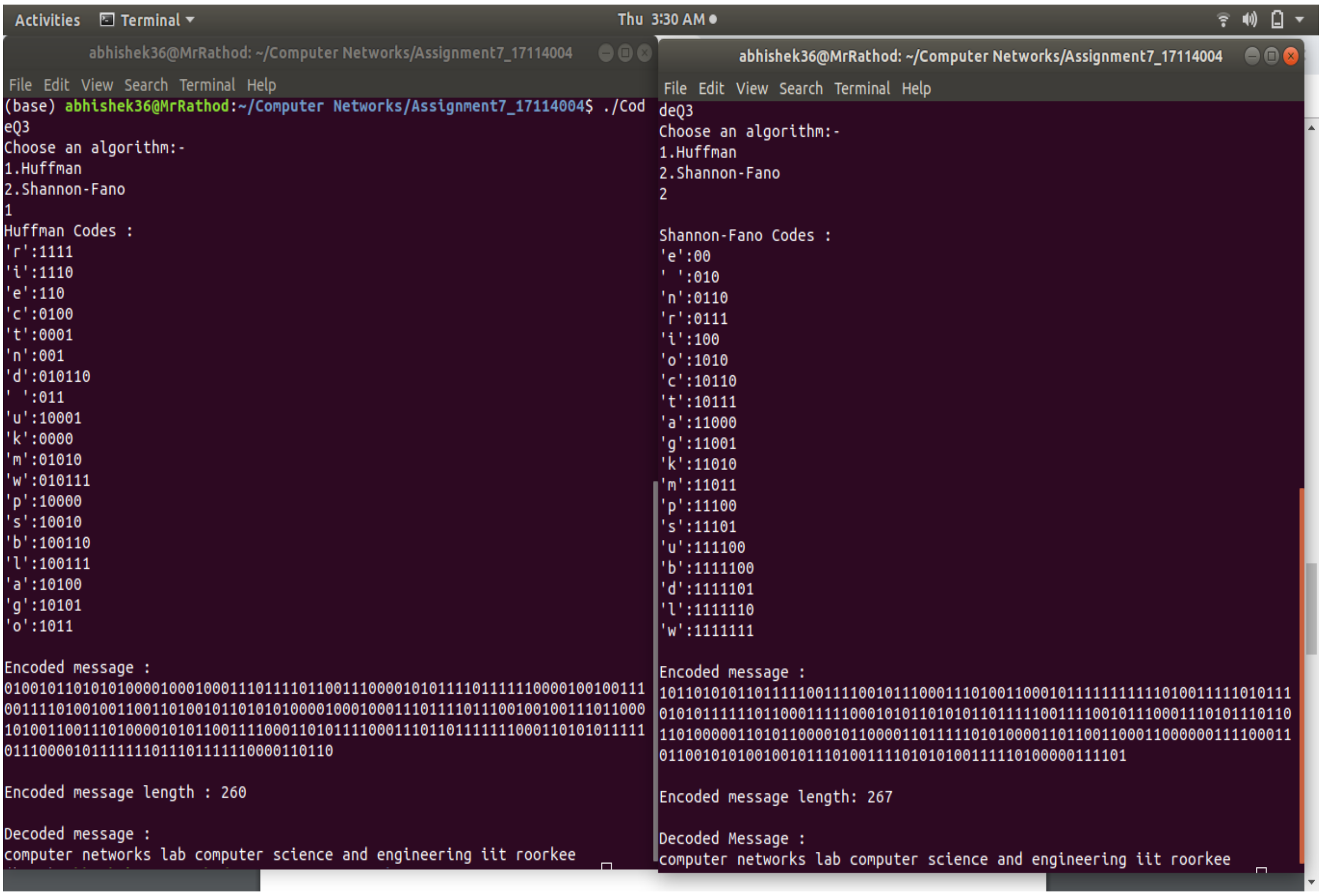
Shannon Fano Algorithm is an entropy encoding technique for lossless data compression of multimedia. Named after Claude Shannon and Robert Fano, it assigns a code to each symbol based on their probabilities of occurrence. It is a variable length encoding scheme, that is, the codes assigned to the symbols will be of varying length.

The steps of the algorithm are as follows:

1. Create a list of probabilities or frequency counts for the given set of symbols so that the relative frequency of occurrence of each symbol is known.

2. Sort the list of symbols in decreasing order of probability, the most probable ones to the left and least probable to the right.
3. Split the list into two parts, with the total probability of both the parts being as close to each other as possible.
4. Assign the value 0 to the left part and 1 to the right part.
5. Repeat the steps 3 and 4 for each part, until all the symbols are split into individual subgroups.

Screenshots-


Activities
Terminal
Thu 3:30 AM
abhishek36@MrRathod: ~/Computer Networks/Assignment7_17114004
File Edit View Search Terminal Help
abhishek36@MrRathod: ~/Computer Networks/Assignment7_17114004
File Edit View Search Terminal Help

```
(base) abhishek36@MrRathod:~/Computer Networks/Assignment7_17114004$ ./Cod
eQ3
Choose an algorithm:-
1.Huffman
2.Shannon-Fano
1
Huffman Codes :
'r':1111
'i':1110
'e':110
'c':0100
't':0001
'n':001
'd':010110
' ':011
'u':10001
'k':0000
'm':01010
'w':010111
'p':10000
's':10010
'b':100110
'l':100111
'a':10100
'g':10101
'o':1011

Encoded message :
0100101101010100001000100011101111011001110000101011110111111000010010011l
0011110100100110011010010110101010000100010001110111101110010010011011000
1010011001110100001010110011110001101011110001110110111111100011010101111l
0111000010111111101110111111000011011O

Encoded message length : 260

Decoded message :
computer networks lab computer science and engineering iit roorkee
```

```
deQ3
Choose an algorithm:-
1.Huffman
2.Shannon-Fano
2

Shannon-Fano Codes :
'e':00
' ':010
'n':0110
'r':0111
'i':100
'o':1010
'c':10110
't':10111
'a':11000
'g':11001
'k':11010
'm':11011
'p':11100
's':11101
'u':111100
'b':1111100
'd':1111101
'l':1111110
'w':1111111

Encoded message :
10110101011011111001111001011100011101001100010111111111101001111010111
01010111111011000111110001010110101011011111001111001011100011101011101l0
11010000011010110000101100001101111101010000110110011000110000000111100011
0110010101001001011101001111010101001111101000000111101

Encoded message length: 267

Decoded Message :
computer networks lab computer science and engineering iit roorkee
```

Data structures used :

1. Queue :- A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO). A good example of a queue is any queue of consumers for a resource where the consumer that came first is served first.

2. unordered_map :- unordered_map is an associated container that stores elements formed by combination of key value and a mapped value. The key value is used to uniquely identify the element and mapped value is the content associated with the key. Both key and value can be of any type predefined or user-defined. Internally unordered_map is implemented using Hash Table, the key provided to map are hashed into indices of hash table that is why performance of data structure depends on hash function a lot but on an average the cost of search, insert and delete from hash table is O(1).

3. File-Handling data structures :-
In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.
Ofstream: Stream class to write on files.
Ifstream: Stream class to read from files.
fstream: Stream class to both read and write from/to files.

4. Vector :- Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container. Vector elements are placed in contiguous storage so that they can be accessed and traversed using iterators. In vectors, data is inserted at the end. Inserting at the end takes differential time, as sometimes there may be a need of extending the array. Removing the last element takes only constant time because no resizing happens. Inserting and erasing at the beginning or in the middle is linear in time.

5. Map :- Maps are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. No two mapped values can have same key values.

6. cassert :- Assertions are statements used to test assumptions made by programmer. For example, we may use assertion to check if pointer returned by malloc() is NULL or not. If expression evaluates to 0 (false), then the expression, sourcecode filename, and line number are sent to the standard error, and then abort() function is called.