

Computer Network Laboratory

Assignment 7

Name: Hemant Singh

Enrollment Number: 17114038

Class: 3rd year, B.Tech CSE

Course: CSN-361



Github Link: <https://github.com/hemant84/CSN-361>

Problem Statements:

Problem 1 :

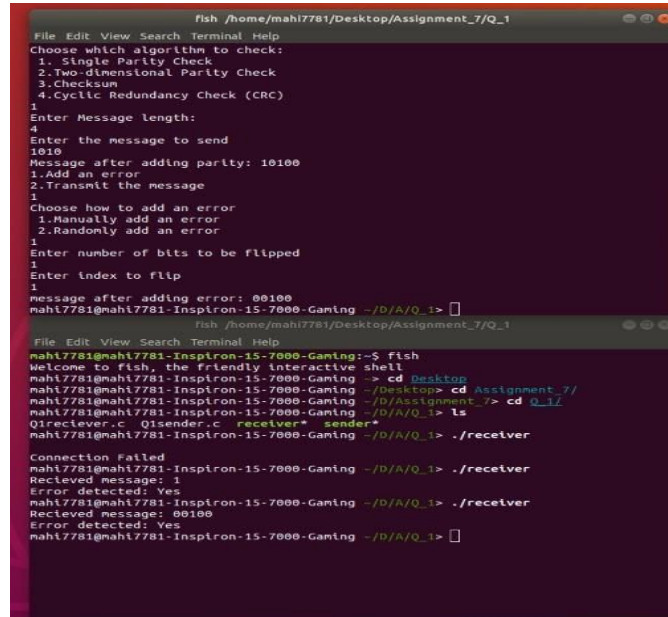
Transmit a binary message (from a sender to a receiver) using socket programming in C and report whether the received msg is correct or not; using the following error detection algorithms:

Algorithms used :

- socket_fd and sockaddr_in for socket creation
- bind function to bin the socket
- listen function to wait for the client to approach the server
- some basic data structures like int and arrays

Screenshot : Single Parity Check

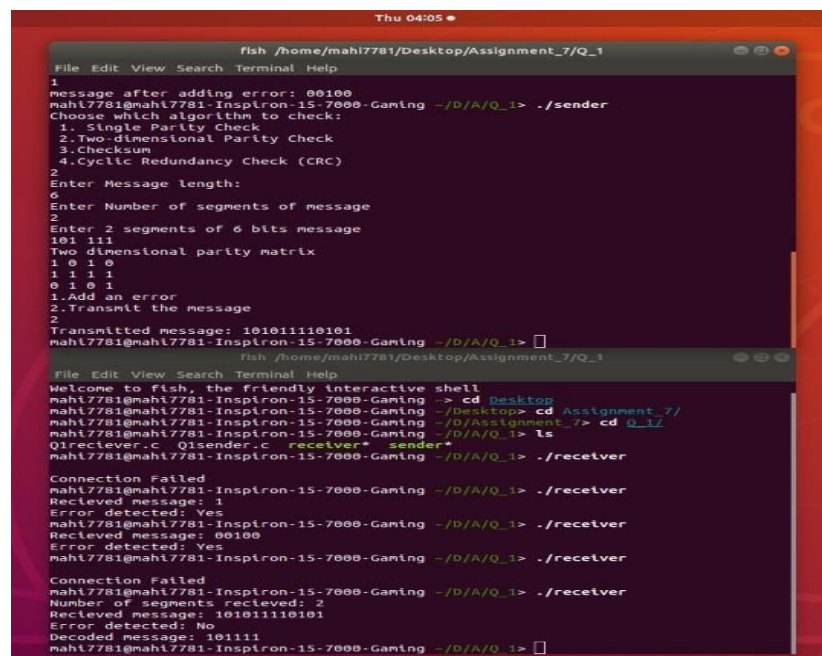
ALGO: Number of ones are added and a bit with $\text{total} \% 2$ is added at the end



```
fish /home/mahi7781/Desktop/Assignment_7/Q_1
File Edit View Search Terminal Help
Choose which algorithm to check:
1. Single Parity Check
2. Two-dimensional Parity Check
3. Checksum
4. Cyclic Redundancy Check (CRC)
1
Enter Message length:
4
Enter the message to send
1010
Message after adding parity: 10100
1. Add an error
2. Transmit the message
1
Choose how to add an error
1. Manually add an error
2. Randomly add an error
1
Enter number of bits to be flipped
1
Enter index to flip
1
Message after adding error: 00100
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1
fish /home/mahi7781/Desktop/Assignment_7/Q_1
File Edit View Search Terminal Help
mahi7781@mahi7781-Inspiron-15-7000-Gaming:~$ fish
Welcome to fish, the friendly interactive shell
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~-> cd Desktop
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop> cd Assignment_7/
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/Assignment_7> cd Q_1/
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ls
Q1receiver.c Q1sender.c receiver* sender*
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Connection Failed
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Received message: 1
Error detected: Yes
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Received message: 00100
Error detected: Yes
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1>
```

Two-dimensional Parity Check

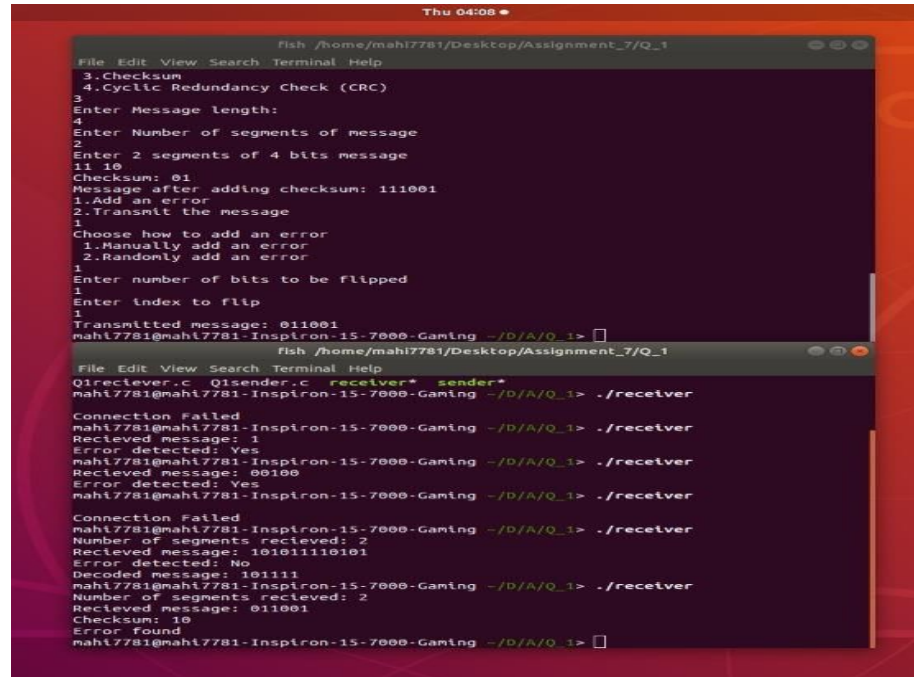
ALGO: Single parity is calculated for each row and each column and an extra bit row and column are added respectively



```
Thu 04:05
fish /home/mahi7781/Desktop/Assignment_7/Q_1
File Edit View Search Terminal Help
1
Message after adding error: 00100
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./sender
Choose which algorithm to check:
1. Single Parity Check
2. Two-dimensional Parity Check
3. Checksum
4. Cyclic Redundancy Check (CRC)
2
Enter Message length:
6
Enter Number of segments of message
2
Enter 2 segments of 6 bits message
101 111
Two dimensional parity matrix
1 0 1 0
1 1 1 1
0 1 0 1
1. Add an error
2. Transmit the message
2
Transmitted message: 101011110101
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1>
fish /home/mahi7781/Desktop/Assignment_7/Q_1
File Edit View Search Terminal Help
Welcome to fish, the friendly interactive shell
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~-> cd Desktop
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop> cd Assignment_7/
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/Assignment_7> cd Q_1/
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ls
Q1receiver.c Q1sender.c receiver* sender*
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Connection Failed
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Received message: 1
Error detected: Yes
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Received message: 00100
Error detected: Yes
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Connection Failed
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Number of segments received: 2
Received message: 101011110101
Error detected: No
Decoded message: 101111
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1>
```

Checksum

ALGO: Data is divided into segments and its checked sum is calculated using the wrapped sum concept. It is checked at the receiver's end and if the total comes to be zero then the data is correct.



```
Thu 04:08
fish /home/mahi7781/Desktop/Assignment_7/Q_1
File Edit View Search Terminal Help
3.Checksum
4.Cyclic Redundancy Check (CRC)
3
Enter Message Length:
4
Enter Number of segments of message
2
Enter 2 segments of 4 bits message
11 10
Checksum: 01
Message after adding checksum: 111001
1.Add an error
2.Transmit the message
1
Choose how to add an error
1.Manually add an error
2.Randomly add an error
1
Enter number of bits to be flipped
1
Enter index to flip
1
Transmitted message: 011001
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1
fish /home/mahi7781/Desktop/Assignment_7/Q_1
File Edit View Search Terminal Help
Q1receiver.c Q1sender.c receiver* sender*
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Connection Failed
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Received message: 1
Error detected: Yes
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Received message: 00100
Error detected: Yes
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Connection Failed
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Number of segments received: 2
Received message: 10101110101
Error detected: No
Decoded message: 101111
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Number of segments received: 2
Received message: 011001
Checksum: 10
Error Found
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1>
```

Cyclic Redundancy Check (CRC)

ALGO: Based on binary-xor division CRC bits are calculated and are appended at the end and the receivers end its division is performed again. If the result comes to be zero, then the data is correct.

```

Thu 04:10
fish /home/mahi7781/Desktop/Assignment_7/Q_1
File Edit View Search Terminal Help
3.Checksum
4.Cyclic Redundancy Check (CRC)
4
Enter length of Divisor:
3
Enter Divisor:
101
Enter Message length:
4
Enter the message to send
1010
Remainder: 00
Message after CRC: 101000
1.Add an error
2.Transmit the message
1
Choose how to add an error
1.Manually add an error
2.Randomly add an error
2
Enter probability of induced error
50
Transmitted message: 101000
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Error detected: Yes
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Error detected: Yes
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Connection Failed
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Number of segments recieved: 2
Recieved message: 10101110101
Error detected: No
Decoded message: 101111
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Number of segments recieved: 2
Recieved message: 011001
Checksum: 10
Error found
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1> ./receiver
Divisor length recieved: 3
Divisor recieved:
Recieved message: 101
Remainder: 11
Error found
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/A/Q_1>


```

Problem 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, if any.

Algorithm used :

1. Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc).
2. All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc).
3. All the other bit positions are marked as data bits.
4. Each data bit is included in a unique set of parity bits, as determined by its bit position in binary form.
 - a. Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7, 9, 11, etc).
 - b. Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7, 10, 11, etc).



c. Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4–7, 12–15, 20–23, etc).

d. Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit bits (8–15, 24–31, 40–47, etc).

e. In general, each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.

5. Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks is odd.

6. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Data Structures:

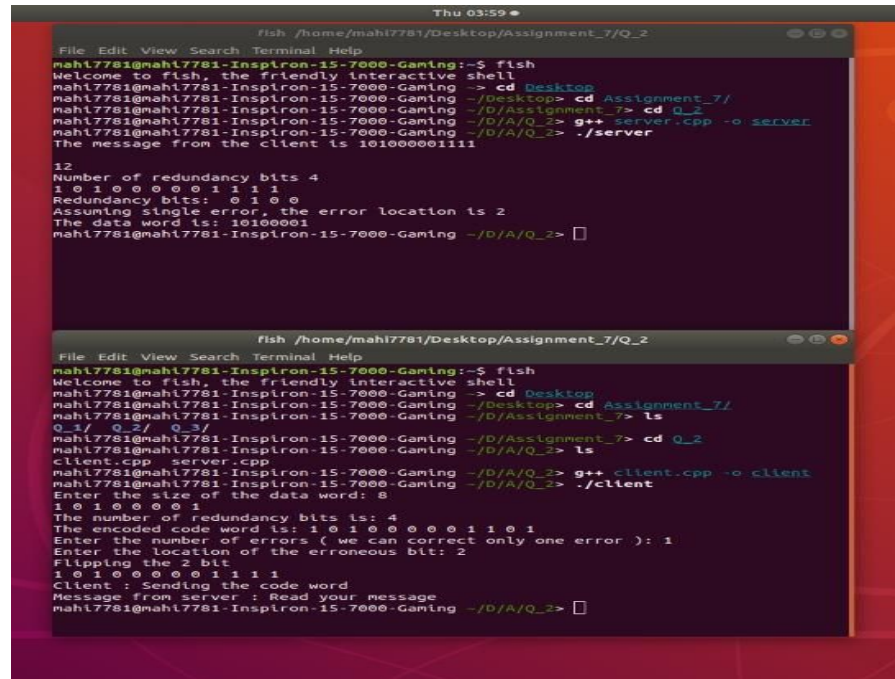
- bind function binds the socket to the address and port number specified in addr and socket creation as mentioned in the above question

- accept function to extract 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.

- Socket connection: same as that of the server's socket creation.

- connect function to establish a connection.

Screenshots:



The image displays two screenshots of a terminal window running a C++ program. The terminal title is "fish /home/mahi7781/Desktop/Assignment_7/Q_2".

Top Screenshot:

```
fish /home/mahi7781/Desktop/Assignment_7/Q_2
File Edit View Search Terminal Help
mahi7781@mahi7781-Inspiron-15-7000-Gaming:~$ fish
Welcome to fish, the friendly interactive shell
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~$ cd Desktop
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop$ cd Assignment_7/
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/Assignment_7$ cd Q_2
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/Assignment_7/Q_2$ g++ server.cpp -o server
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/Assignment_7/Q_2$ ./server
The message from the client is 101000001111
12
Number of redundancy bits 4
1 0 1 0 0 0 0 1 1 1 1
Redundancy bits: 0 1 0 0
Assuming single error, the error location is 2
The data word is: 10100001
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/Assignment_7/Q_2$
```

Bottom Screenshot:

```
fish /home/mahi7781/Desktop/Assignment_7/Q_2
File Edit View Search Terminal Help
mahi7781@mahi7781-Inspiron-15-7000-Gaming:~$ fish
Welcome to fish, the friendly interactive shell
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~$ cd Desktop
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop$ cd Assignment_7/
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/Assignment_7$ ls
Q_1/  Q_2/  Q_3/
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/Assignment_7$ cd Q_2
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/Assignment_7/Q_2$ ls
client.cpp  server.cpp
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/Assignment_7/Q_2$ g++ client.cpp -o client
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/Assignment_7/Q_2$ ./client
Enter the size of the data word: 8
1 0 1 0 0 0 0 1
The number of redundancy bits is: 4
The encoded code word is: 1 0 1 0 0 0 0 1 1 0 1
Enter the number of errors ( we can correct only one error ): 1
Enter the location of the erroneous bit: 2
Flipping the 2 bit
1 0 1 0 0 0 0 1 1 1
Client : Sending the code word
Message from server : Read your message
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/Assignment_7/Q_2$
```

Problem 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:

a) Huffman b) Shannon-Fano

Data Structures Used:

- Node- a struct tree node denotes a char and its properties
- comp-Comparison object to be used to order the heap
- encode-traverse the Huffman Tree and store Huffman Codes in a map.
- inbuilt sort function to sort according to probabilities
- decode-traverse the Huffman Tree and decode the encoded string
- Huffman Tree Builder
- Probability table storing probabilities
- ifstream and ofstream : Stream class to read from files
- fstream: Stream class to both read and write from/to files.

Screenshots:

Huffman Algorithm:

There are two major parts in Huffman Coding

1) Build a Huffman Tree from input characters. Steps to build Huffman Tree

Input is an array of unique characters along with their frequency of occurrences and output is Huffman Tree.

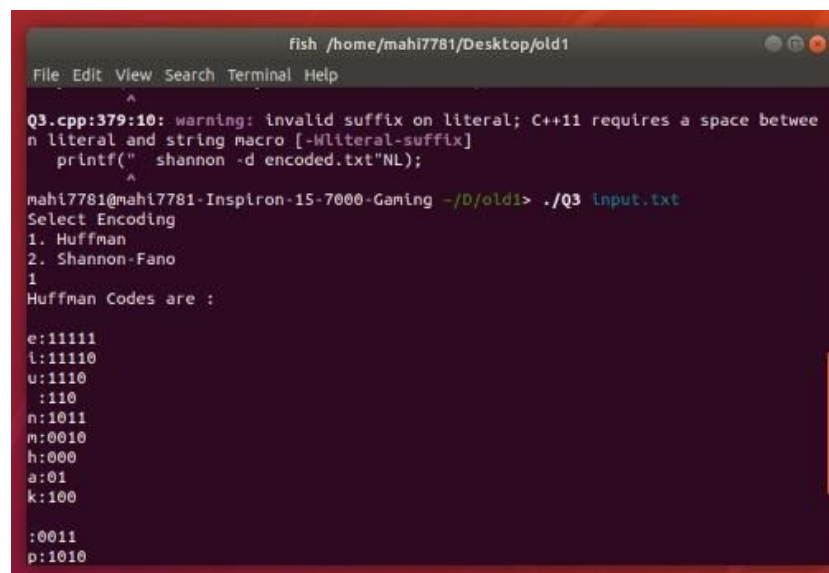
Create a leaf node for each unique character and build a min-heap of all leaf nodes (Min Heap is used as a priority queue. The value of frequency field is used to compare two nodes in min heap. Initially, the least frequent character is at root)

Extract two nodes with the minimum frequency from the min-heap.

Create a new internal node with a frequency equal to the sum of the two nodes frequencies. Make the first extracted node as its left child and the other extracted node as its right child. Add this node to the min-heap.

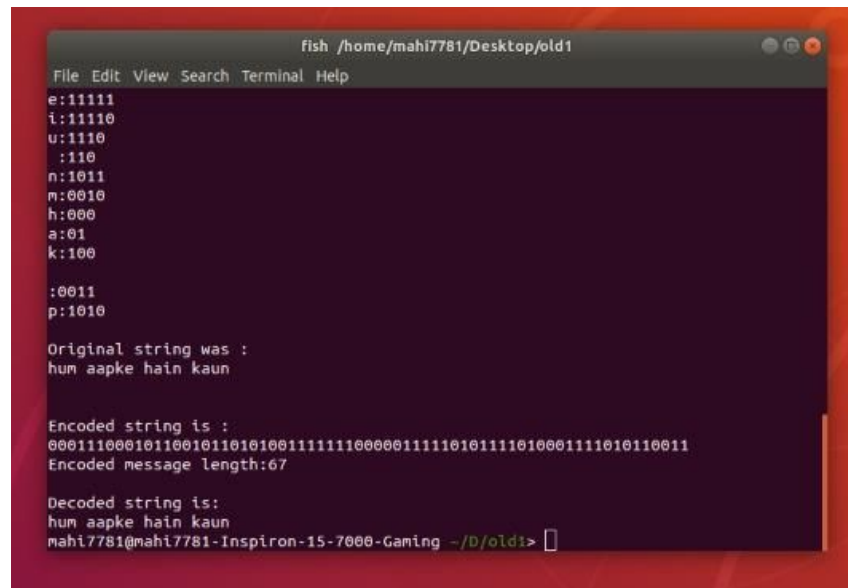
Repeat steps#2 and #3 until the heap contains only one node. The remaining node is the root node and the tree is complete.

2) Traverse the Huffman Tree and assign codes to characters.



```
fish /home/mahi7781/Desktop/old1
File Edit View Search Terminal Help
Q3.cpp:379:10: warning: invalid suffix on literal; C++11 requires a space between
n literal and string macro [-Wliteral-suffix]
    printf(" shannon -d encoded.txt"NL);
    ^
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/old1> ./Q3 input.txt
Select Encoding
1. Huffman
2. Shannon-Fano
1
Huffman Codes are :

e:11111
l:11110
u:1110
i:110
n:1011
m:0010
h:000
a:01
k:100
o:0011
p:1010
```

A terminal window titled 'fish /home/mahi7781/Desktop/old1' with a menu bar (File, Edit, View, Search, Terminal, Help). It displays the frequency of characters in the string 'hum aapke hain kaun', the original string, the encoded binary string, the encoded message length (67), and the decoded string. The encoding process is shown as a series of binary splits based on frequency.

```
fish /home/mahi7781/Desktop/old1
File Edit View Search Terminal Help
e:11111
i:11110
u:1110
:110
n:1011
m:0010
h:000
a:01
k:100

:0011
p:1010

Original string was :
hum aapke hain kaun

Encoded string is :
000111000101100101101001111110000011110101111010001111010110011
Encoded message length:67

Decoded string is:
hum aapke hain kaun
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/Desktop/old1>
```

Shannon-Fano Algorithm:

The steps of the algorithm are as follows:

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

```
fish /home/mahi7781/Desktop/old1
File Edit View Search Terminal Help
mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/old1> ./Q3 input.txt
Select Encoding
1. Huffman
2. Shannon-Fano
2

11
a      0.200000      00
      0.150000      010
h      0.100000      011
k      0.100000      100
n      0.100000      101
u      0.100000      1100
      0.050000      1101
e      0.050000      11100
i      0.050000      11101
m      0.050000      11110
p      0.050000      11111

0111100111100100000111111001110001001100111011010101000011001011101

Length of encoded message : 68
```

```
fish /home/mahi7781/Desktop/old1
File Edit View Search Terminal Help
k      0.100000      100
n      0.100000      101
u      0.100000      1100
      0.050000      1101
e      0.050000      11100
i      0.050000      11101
m      0.050000      11110
p      0.050000      11111

0111100111100100000111111001110001001100111011010101000011001011101

Length of encoded message : 245

mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/old1> ./Q3 -d encoded.txt
Select Encoding
1. Huffman
2. Shannon-Fano
2

hum aapke hain kaun

mahi7781@mahi7781-Inspiron-15-7000-Gaming ~/D/old1> 
```