Computer Network Laboratory

Assignment 7

Name: Gagan Kumre
Enrollment Number: 17114028
Class: 3rd year, B.Tech CSE
Course: CSN-361

GitHub link - https://github.com/gagankumre/CSN361/tree/master/Assignment

Two problems were given for this assignment. They are-

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:

1. Single Parity Check

2. Two-dimensional Parity Check

3. Checksum

4. Cyclic Redundancy Check (CRC)

Data structure used :

1. socket_fd and sockaddr_in for socket creation
2. bind function to bin the socket
3. listen function to wait for the client to approach the server
4. some basic data structures like int and arrays

Algorithms used :

1. Single Parity Check

ALGO: Number of ones are added and a bit with total%2 is added at end

2. 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

3. Checksum

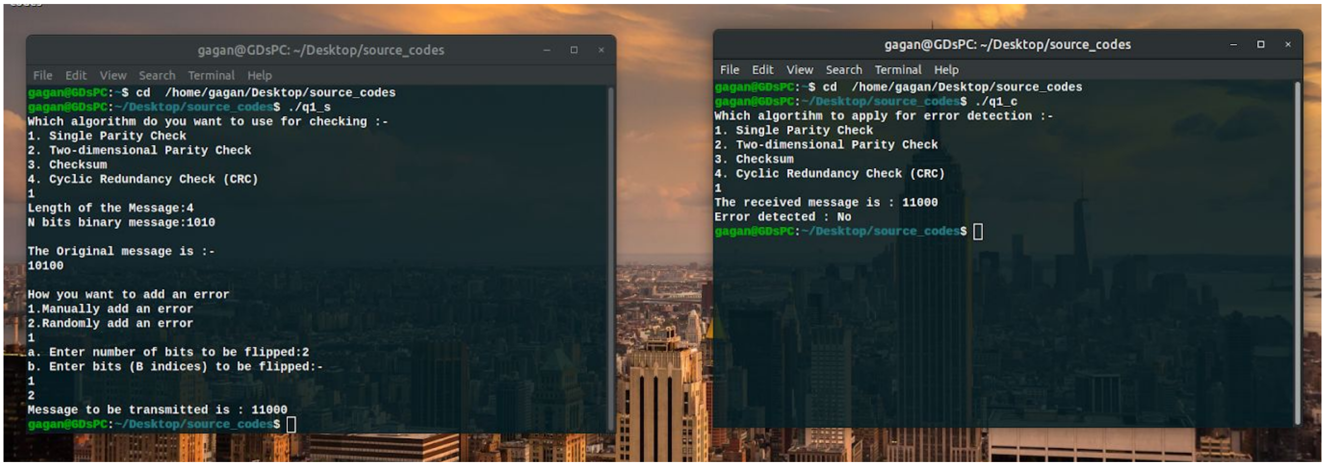
ALGO: Data is divided into segments and its checked sum is calculated using wrapped sum concept. It is checked at the receiver’s end and if the total comes to be zero then the data is correct.

4. 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.

Screenshots :

1. Single Parity Check

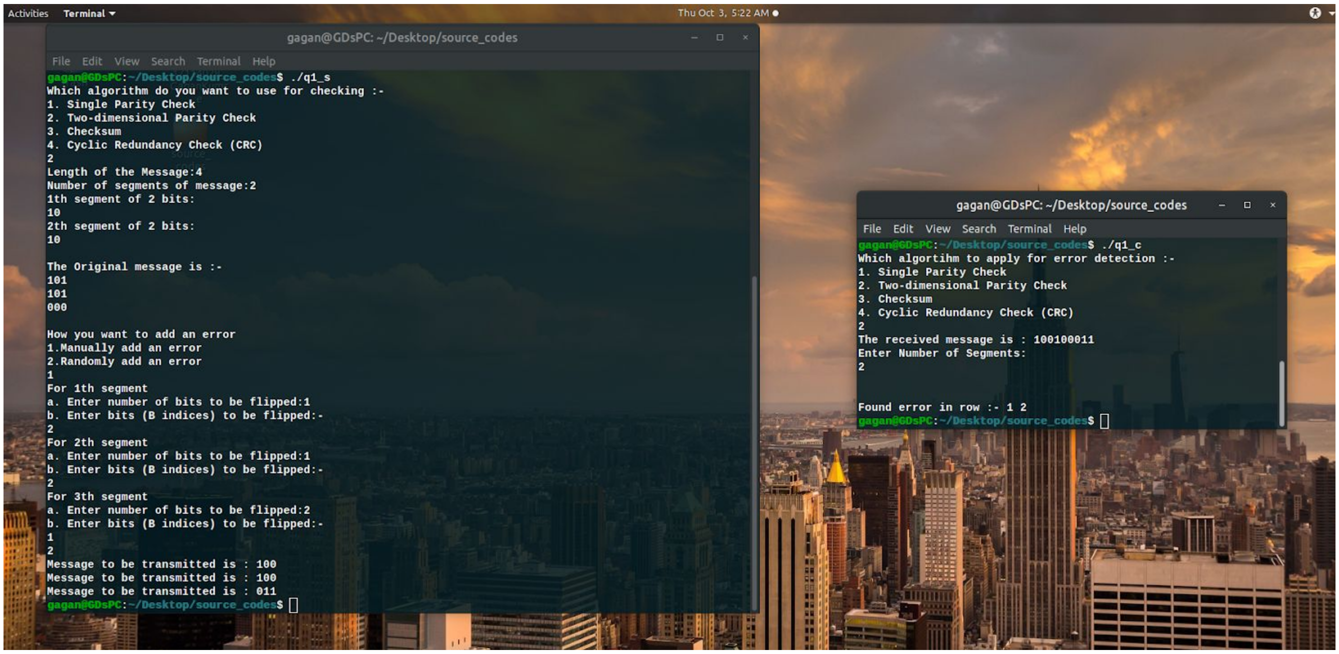

gagan@GDsPC: ~/Desktop/source_codes
File Edit View Search Terminal Help

```
gagan@GDsPC:~$ cd  /home/gagan/Desktop/source_codes
gagan@GDsPC:~/Desktop/source_codes$ ./q1_s
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:4
N bits binary message:1010

The Original message is :-
10100

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:2
b. Enter bits (B indices) to be flipped:-
1
2
Message to be transmitted is : 11000
gagan@GDsPC:~/Desktop/source_codes$ 
```

gagan@GDsPC: ~/Desktop/source_codes
File Edit View Search Terminal Help

```
gagan@GDsPC:~$ cd  /home/gagan/Desktop/source_codes
gagan@GDsPC:~/Desktop/source_codes$ ./q1_c
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 : 11000
Error detected : No
gagan@GDsPC:~/Desktop/source_codes$ 
```

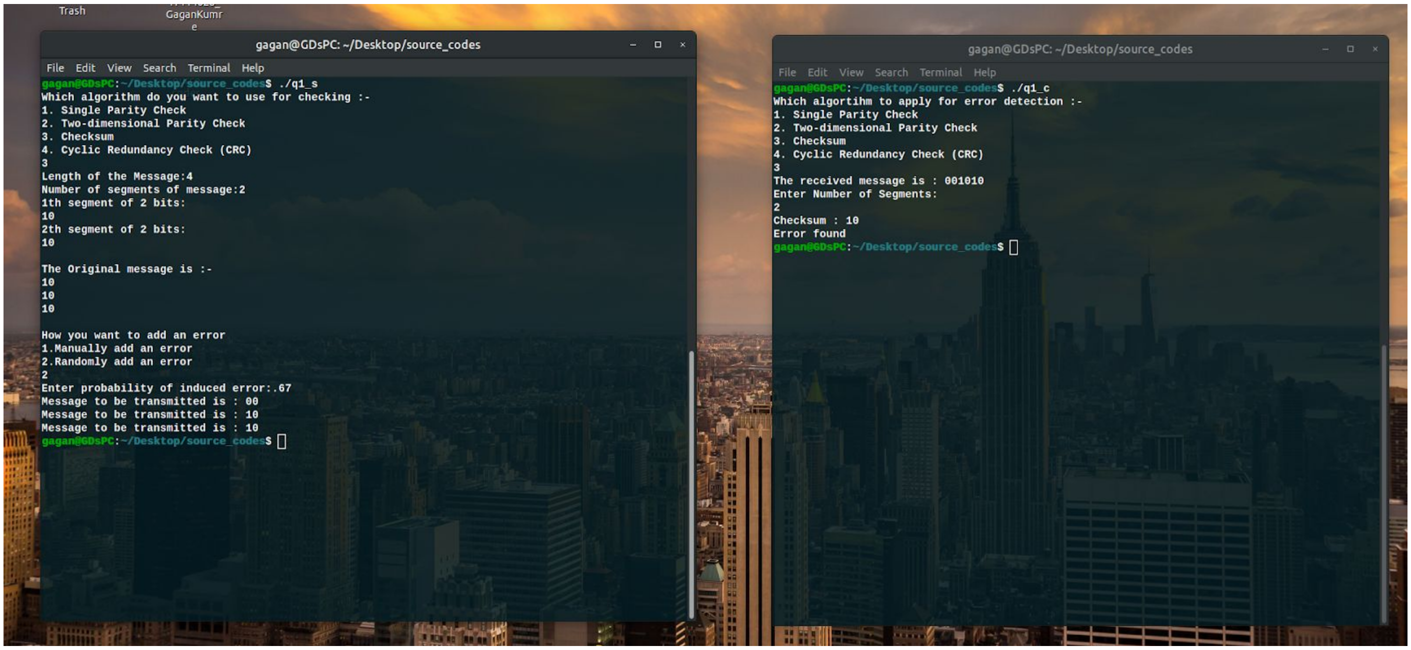
2. Two-dimensional Parity Check


Activities
Terminal
Thu Oct 3, 5:22 AM
gagan@GDsPC: ~/Desktop/source_codes
File Edit View Search Terminal Help
gagan@GDsPC: ~/Desktop/source_codes
File Edit View Search Terminal Help

```
gagan@GDsPC:~/Desktop/source_codes$ ./q1_s
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)
2
Length of the Message:4
Number of segments of message:2
1th segment of 2 bits:
10
2th segment of 2 bits:
10

The Original message is :-
101
101
000

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:1
b. Enter bits (B indices) to be flipped:-
2
For 2th segment
a. Enter number of bits to be flipped:1
b. Enter bits (B indices) to be flipped:-
2
For 3th segment
a. Enter number of bits to be flipped:2
b. Enter bits (B indices) to be flipped:-
1
2
Message to be transmitted is : 100
Message to be transmitted is : 100
Message to be transmitted is : 011
gagan@GDsPC:~/Desktop/source_codes$ 
```

```
gagan@GDsPC:~/Desktop/source_codes$ ./q1_c
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 : 100100011
Enter Number of Segments:
2


Found error in row :- 1 2
gagan@GDsPC:~/Desktop/source_codes$ 
```

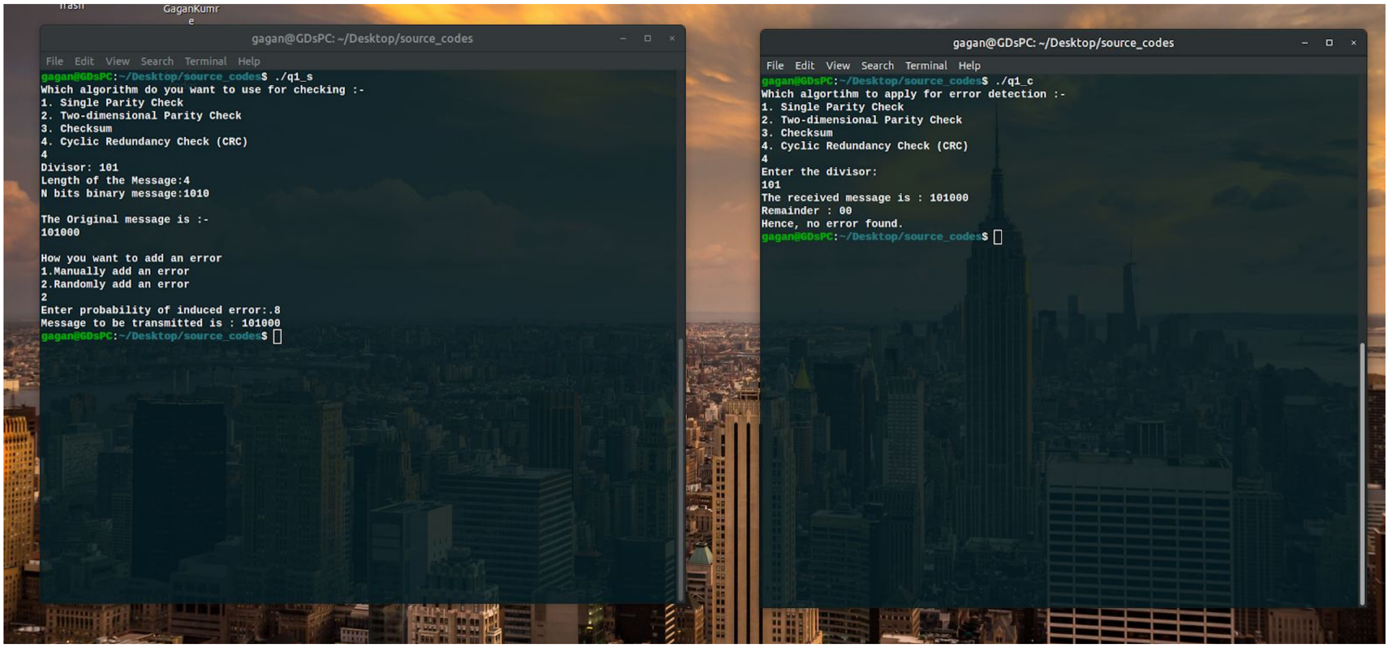
3. Checksum


Trash
GaganKumr
e
gagan@GDsPC: ~/Desktop/source_codes
File Edit View Search Terminal Help
gagan@GDsPC: ~/Desktop/source_codes
File Edit View Search Terminal Help

```
gagan@GDsPC:~/Desktop/source_codes$ ./q1_s
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:4
Number of segments of message:2
1th segment of 2 bits:
10
2th segment of 2 bits:
10

The Original message is :-
10
10
10

How you want to add an error
1.Manually add an error
2.Randomly add an error
2
Enter probability of induced error:.67
Message to be transmitted is : 00
Message to be transmitted is : 10
Message to be transmitted is : 10
gagan@GDsPC:~/Desktop/source_codes$ 
```

```
gagan@GDsPC:~/Desktop/source_codes$ ./q1_c
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 : 001010
Enter Number of Segments:
2
Checksum : 10
Error found
gagan@GDsPC:~/Desktop/source_codes$ 
```

4. Cyclic Redundancy Check (CRC)


Trash
GaganKumr
e
gagan@GDsPC: ~/Desktop/source_codes
File Edit View Search Terminal Help
gagan@GDsPC: ~/Desktop/source_codes
File Edit View Search Terminal Help

```
gagan@GDsPC:~/Desktop/source_codes$ ./q1_s
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: 101
Length of the Message:4
N bits binary message:1010

The Original message is :-
101000

How you want to add an error
1.Manually add an error
2.Randomly add an error
2
Enter probability of induced error:.8
Message to be transmitted is : 101000
gagan@GDsPC:~/Desktop/source_codes$ 
```

```
gagan@GDsPC:~/Desktop/source_codes$ ./q1_c
Which algortihm 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:
101
The received message is : 101000
Remainder : 00
Hence, no error found.
gagan@GDsPC:~/Desktop/source_codes$ 
```

Problem 2 :

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

Algorithms 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 its bit position in binary form.

- 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).

- 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).

- 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).

- 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).

- 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.

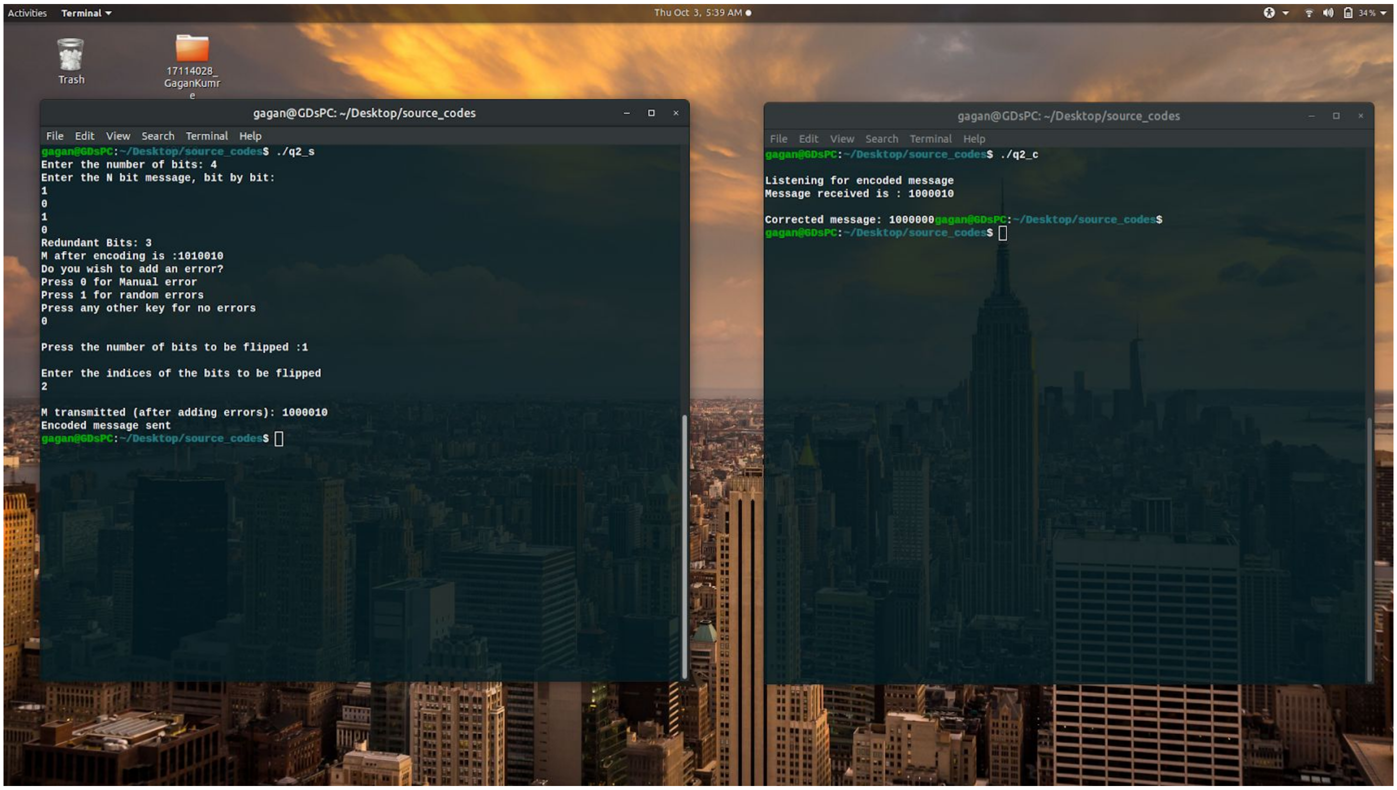
6. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Data structures used :

1. bind function binds the socket to the address and port number specified in addr and socket creation as mentioned in the above question
2. 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.
3. Socket connection: same as that of server’s socket creation

4. connect function to establish a connection

Screenshots :


Activities
Terminal
Thu Oct 3, 5:39 AM
34 %
Trash
17114028_
GaganKumr
e
gagan@GDsPC: ~/Desktop/source_codes
File Edit View Search Terminal Help

```
gagan@GDsPC:~/Desktop/source_codes$ ./q2_s
Enter the number of bits: 4
Enter the N bit message, bit by bit:
1
0
1
0
Redundant Bits: 3
M after encoding is :1010010
Do you wish to add an error?
Press 0 for Manual error
Press 1 for random errors
Press any other key for no errors
0

Press the number of bits to be flipped :1

Enter the indices of the bits to be flipped
2

M transmitted (after adding errors): 1000010
Encoded message sent
gagan@GDsPC:~/Desktop/source_codes$ 
```

gagan@GDsPC: ~/Desktop/source_codes
File Edit View Search Terminal Help

```
gagan@GDsPC:~/Desktop/source_codes$ ./q2_c

Listening for encoded message
Message received is : 1000010

Corrected message: 1000000gagan@GDsPC:~/Desktop/source_codes$
gagan@GDsPC:~/Desktop/source_codes$ 
```

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:

1. Huffman

2. Shannon-Fano

Algorithms used :

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

Data structures used :

Huffman Tree:

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.

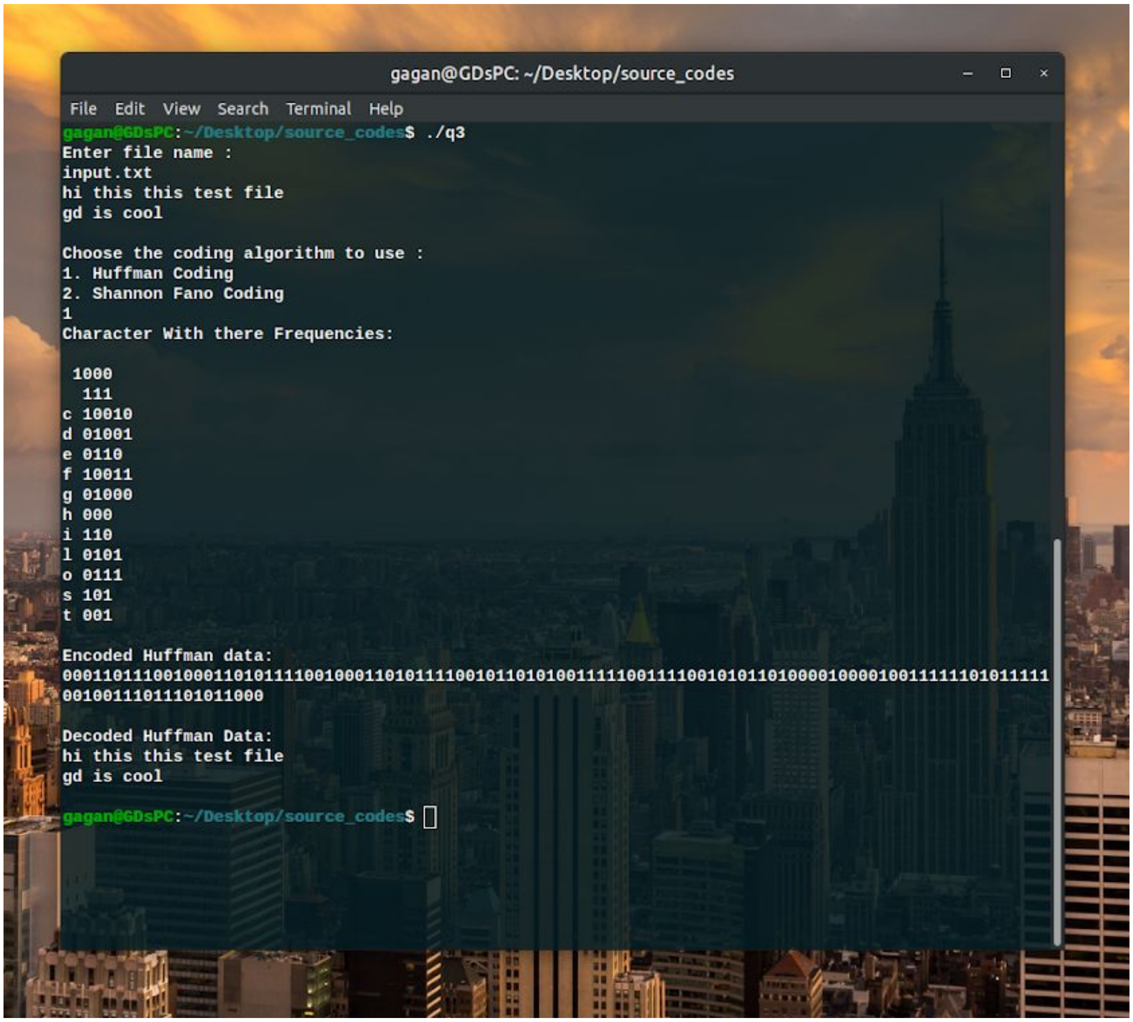
Shannon-Fano Algorithm:

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 :

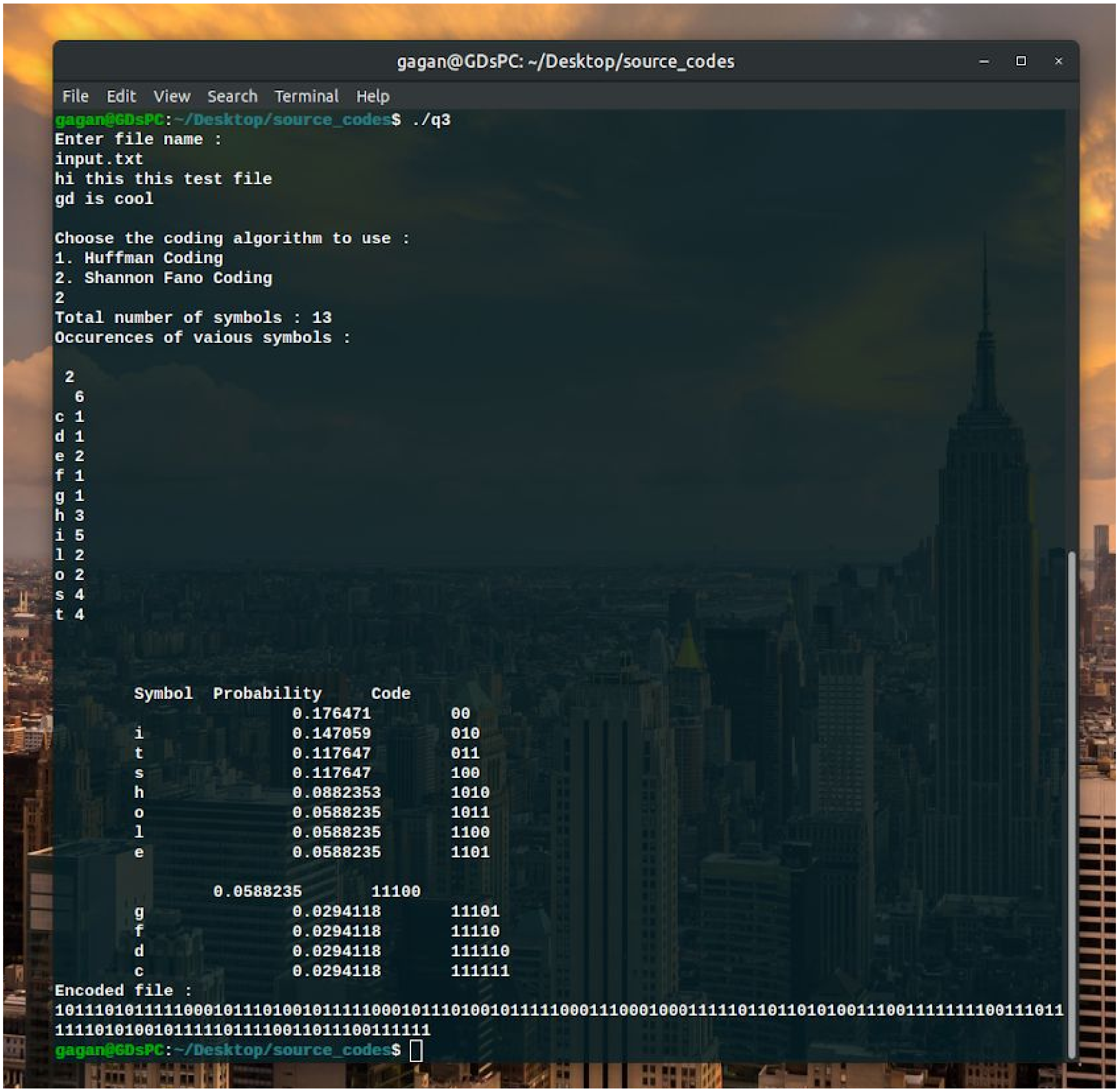
1. Huffman Tree



gagan@GDsPC: ~/Desktop/source_codes
File Edit View Search Terminal Help

```
gagan@GDsPC:~/Desktop/source_codes$ ./q3
Enter file name :
input.txt
hi this this test file
gd is cool

Choose the coding algorithm to use :
1. Huffman Coding
2. Shannon Fano Coding
1
Character With there Frequencies:

 1000
  111
c 10010
d 01001
e 0110
f 10011
g 01000
h 000
i 110
l 0101
o 0111
s 101
t 001

Encoded Huffman data:
00011011100100011010111100100011010111100101101010011111001111001010110100001000010011111101011111
00100111011101011000

Decoded Huffman Data:
hi this this test file
gd is cool

gagan@GDsPC:~/Desktop/source_codes$ 
```

2. Shannon-Fano Algorithm


gagan@GDsPC: ~/Desktop/source_codes
File Edit View Search Terminal Help

```
gagan@GDsPC:~/Desktop/source_codes$ ./q3
Enter file name :
input.txt
hi this this test file
gd is cool

Choose the coding algorithm to use :
1. Huffman Coding
2. Shannon Fano Coding
2
Total number of symbols : 13
Occurences of vaious symbols :

 2
  6
c 1
d 1
e 2
f 1
g 1
h 3
i 5
l 2
o 2
s 4
t 4



        Symbol  Probability     Code
                        0.176471        00
        i               0.147059        010
        t               0.117647        011
        s               0.117647        100
        h               0.0882353       1010
        o               0.0588235       1011
        l               0.0588235       1100
        e               0.0588235       1101

                0.0588235       11100
        g               0.0294118       11101
        f               0.0294118       11110
        d               0.0294118       111110
        c               0.0294118       111111
Encoded file :
101110101111100010111010010111110001011101001011111000111000100011111011011010100111001111111100111011
11110101001011111011110011011100111111
gagan@GDsPC:~/Desktop/source_codes$ 
```