Name: Shubham Chemate

Roll Number: 31118

Subject: CNS Lab Assignment 03 Writeup and performance screenshots

1

PICT, PUNE

Subject: Computer Networks & Security

Assignment : 03

Roll. No. 31118

11-Sept-2021

Title: Error detection

Problem statement:

Write a program for error detection & correction for 7/8 bits ASCII codes using Hamming codes (or) CRC.

Requirements

IDE: Eclipse (version: 2020-release)
compiler: gcc (version: 6.3.0)
Hardware: Intel i5-8265U CPU @ 1.60 GHz
8GB RAM & 512 GB SSD.

Description:

· Data link layer:

- In OSI model, data link layer is the 4th layer from top & 2nd from bottom.
- Data link layer gets the raw data from physical layer, try to make physical link reliable.
- It attempts to detect & recover the data that it gets from physical layer.
- Data link layer protocol defines the format of the packet exchanged across nodes, error detection, retransmission, flow control & random acess.

Services provided by data link layer:

i) Framing & Link Access:

i) data link layer takes packets from network layer & encapsulate them as frames

ii) It sends each frame bit-by-bit on the hardware.

iii) At receivers end, it assembles bit into frames.

2) Addressing

i) provides layer-2 hardware addressing mechanism

ii) Hardware address is unique & provided at the time of manufacturing.

3) Synchronization: machines should synchronized while sending message on link.

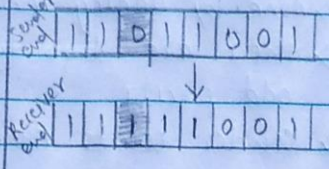
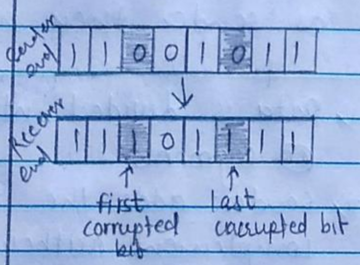
4) Error Control: Errors are detected & attempt is made to recover actual bits. provides error reporting mechanism to sender.

5) Flow control: machines on same link may have different speed/capacity. Data link layer ensures flow control.

6) Multi-Access: To avoid/reduce collisions data link layer provides mechanism such as CSMA/CD to equip capability of accessing a shared media among multiple systems.

Types of Errors:

There are three types of errors:

Error Type	Diagram	Description
i) Single bit Error	 Sender end 1 1 0 1 1 0 0 1 Receiver end 1 1 1 1 1 0 0 1	i) Only one -bits of given data-unit is corrapted. ii) least-likely in serial data transmission. iii) happen in parallel data transmission
2) Burst Error	 Sender end 1 1 0 0 1 0 1 1 Receiver end 1 1 1 0 1 1 1 1 first corrupted bit last corrupted bit length = 4	i) Two/more bits in data unit are corrupted ii) length is measured from first corrupted bit to last. iii) generally happen in serial data transmission.

Methods of error detection & correction:

error detection: Involves checking wheather any error has occured or not.

error correction: Involves ascertaining the exact number of bits that has been corrupted & location of corrupted bits.

Error detection techniques:

Name	description
1> parity check.	i> done by adding extra bit called parity bit ii> there are two types even parity & odd parity iii> Sender counts no.. of ones & make parity either even/odd by adding extra 1. iv> On receiver end, is parity is same as sender then message is accepted
2> checksum	i> Data is divided into fixed sized frames (or) segments. ii> Sender: adds the segment using 1's complement arithematic to get sum. Then it complements sum to get checksum & sends it along data frames iii> Receiver adds incoming segment along with the checksum using 1's complements & then complements it. iv> If result is zero, data is accepted.
3> Cyclic Redundancy Check (CRC)	i> It involves binary division of data bits by predetermined divisor which is generated using polynomials. ii> At sender side after division, remainder is appended at the end of data segment. iii> At receiver end, incoming data unit is divided by divisor. If there is 0 remainder data segment get's accepted

Error Correction Techniques

Technique	Description
1>Backward Error correction	1> Receiver request the sender to retransmit frame if he detects error. 11> It is simple technique which is efficient when retransmission is not expensive.
2>Forward Error ~~Detection~~ correction	1> Receiver executes error correcting code if he detects error. 11> This saves bandwidth requirement for retransmission. 111> If there are too-many errors retransmission is needed.

Error correction codes:

1>Hamming Codes.
2>Binary convolution Code
3>Reed. Solomon Code
4> Low. density parity.check code.

here,

r= redundant bits

m= message length

3> Parity Bits:

Parity bits are appended to make parity of 1's either even (or) odd. For our analysis let's consider even parity from now.

Note that:

even parity: no. of 1sbits should be even. For odd 1's extra 1 should be added.

odd parity: no of 1's should be odd. For even 1's extra 1 should be added

4> Algorithm:

1> Calculate the value of r = redundent bits by formula $2^r \geq d+r+1$ & d = length of data word.

2> Effective length of codeword will be $n = r+d$

3> ~~where~~ Let parity bits be $P_1, P_2, P_3, .. P_r$. calculate $p_1, p_2,$ as follows:

p_1 = consider all the bits positions whose binary representation includes 1 in least pos. significant eg. (1, 3, 5, 7, 9, ..)

p_2 = consider all the bits positions whose binary representation includes 1 in second position from LSB.

⋮

P_r = . . .

4> As we are considering for even parity, set a parity bit to 1 if the total number of 1's in the positions it checks is odd.

5) Place these redundant bits at the positions of powers of 2 ie. 1, 2, 4, .. etc other positions will be occupied by data bits.

At receivers end:

1) Calculate the values of $P_1, P_2 \ldots P_r$.
if they are of opposite parity error exist.

2) Calculating the position of corrupted bit.
convert the value of $(P_r P_{r-1} \ldots P_1)$ to decimal.
This gives the ~~value~~ position of corrupted bit.

example:

Even parity

consider the message 1101011

here d = 7

using formula $2^r \geq d+r+1$

we get r = 4.

hence our message length will be r + d = 11

		1		1	0	1		0	1	0
P_1	P_2	d_3	P_4	d_5	d_6	d_7	P_8	d_9	d_{10}	d_{11}

consider:

$P_1 = d_3 d_5 d_7 d_9 d_{11} = \underbrace{1\phi100}_{} = 1$

$P_2 = d_3 d_6 d_7 d_{10} d_{11} = \underbrace{10110}_{\text{odd 1's}} = 1$

~~[illegible]~~ odd 1's ~~1010 = 1~~

$P_4 = d_5 d_6 d_7 = \underbrace{101}_{\text{even 1's}} = 0$

$P_8 = d_9 d_{10} d_{11} = 010 = 1$

our codeword to send is:

1	1	1	0	1	0	1	1	0	1	0

On receiver's end:

$p_1\ p_2\ d_3\ p_4\ d_5\ d_6\ d_7\ p_8\ d_9\ d_{10}\ d_{11}$

let we received: 11101011010

here $p_1 = \underbrace{111100}_{\text{even 1's}} = 0$

$p_2 = 110110 = 0$

$p_4 = 0101 = 0$

$p_8 = 1010 = 0$

As all parity bits satisfy parity condition, the received message is accepted.

$p_1\ p_2\ d_3\ p_4\ d_5\ d_6\ d_7\ p_8\ d_9\ d_{10}\ d_{11}$

let we received 11100011010

here $p_1 = 110100 = 1$

$p_2 = 110110 = 0$

$p_4 = 0001 = 1$

$p_8 = 1010 = 0$

As bits p_1 & P_2 violates parity condition, there must be an error in received message

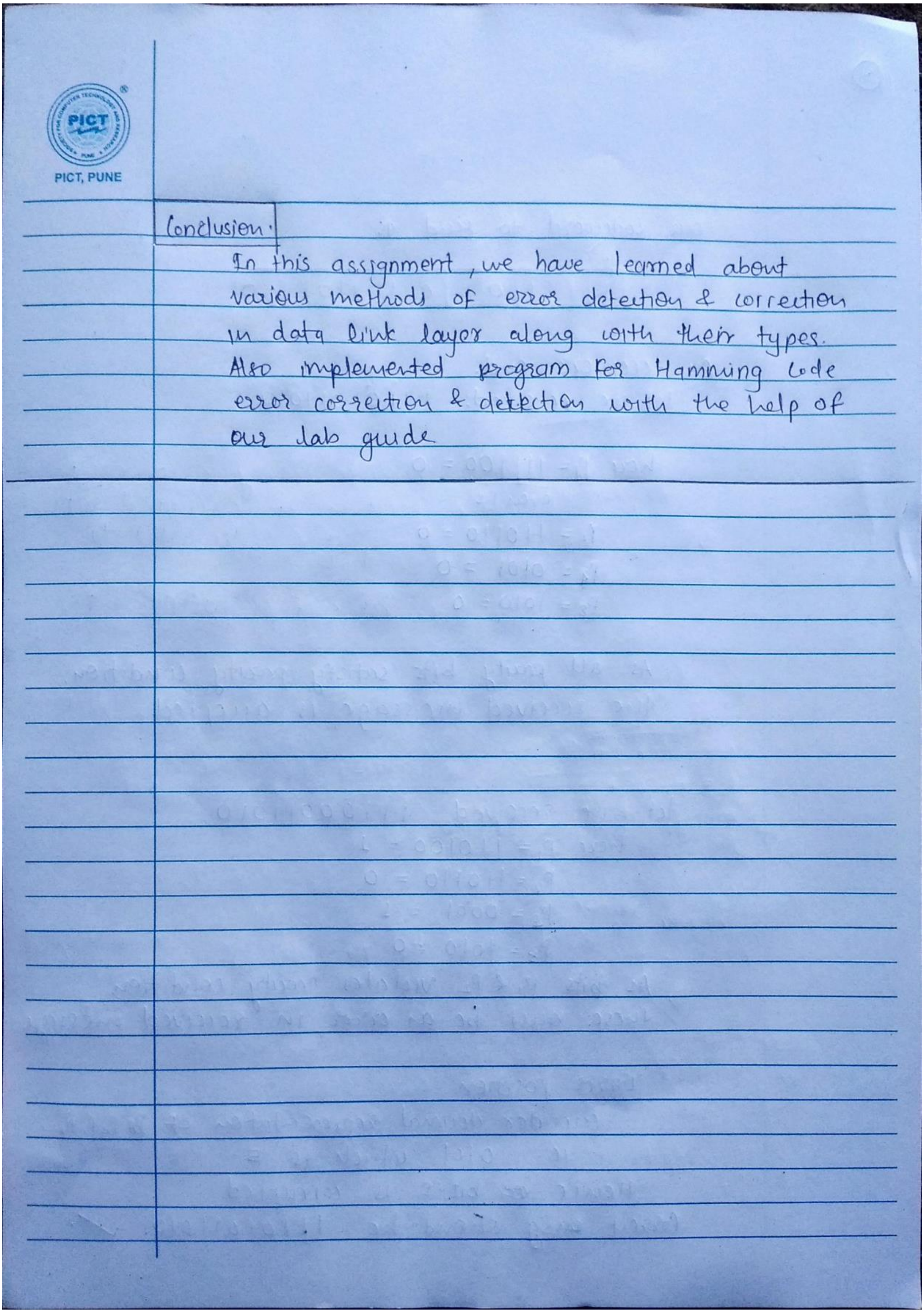
Error position:

consider decimal representation of $P_8 P_4 P_2 P_1$

ie. 0101 which is 5.

Hence bit-5 is corrupted.

Correct msg. should be 11101011010. ✓

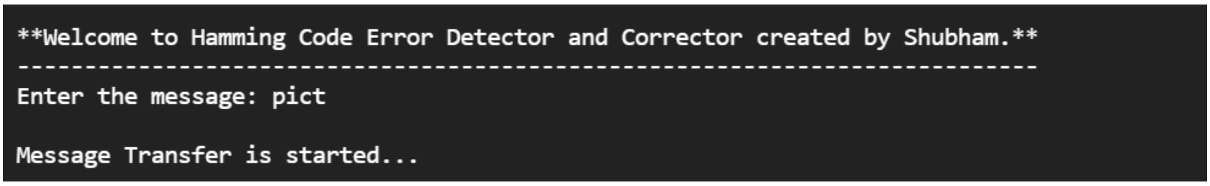


PICT, PUNE

Conclusion:

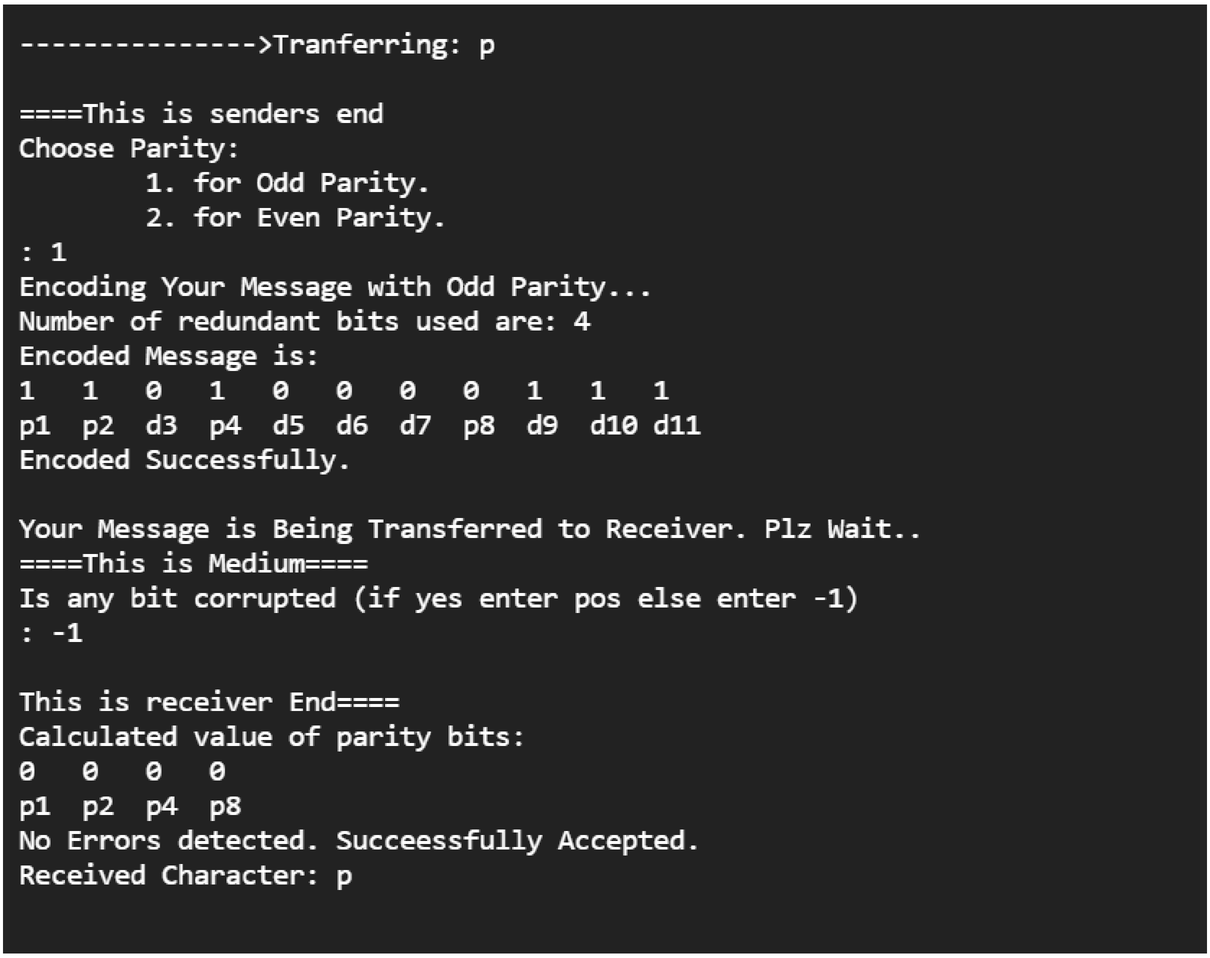
In this assignment, we have learned about various methods of error detection & correction in data link layer along with their types. Also implemented program for Hamming code error correction & detection with the help of our lab guide

Screenshots of performance:



```
**Welcome to Hamming Code Error Detector and Corrector created by Shubham.**
----------------------------------------------------------------------------
Enter the message: pict

Message Transfer is started...
```



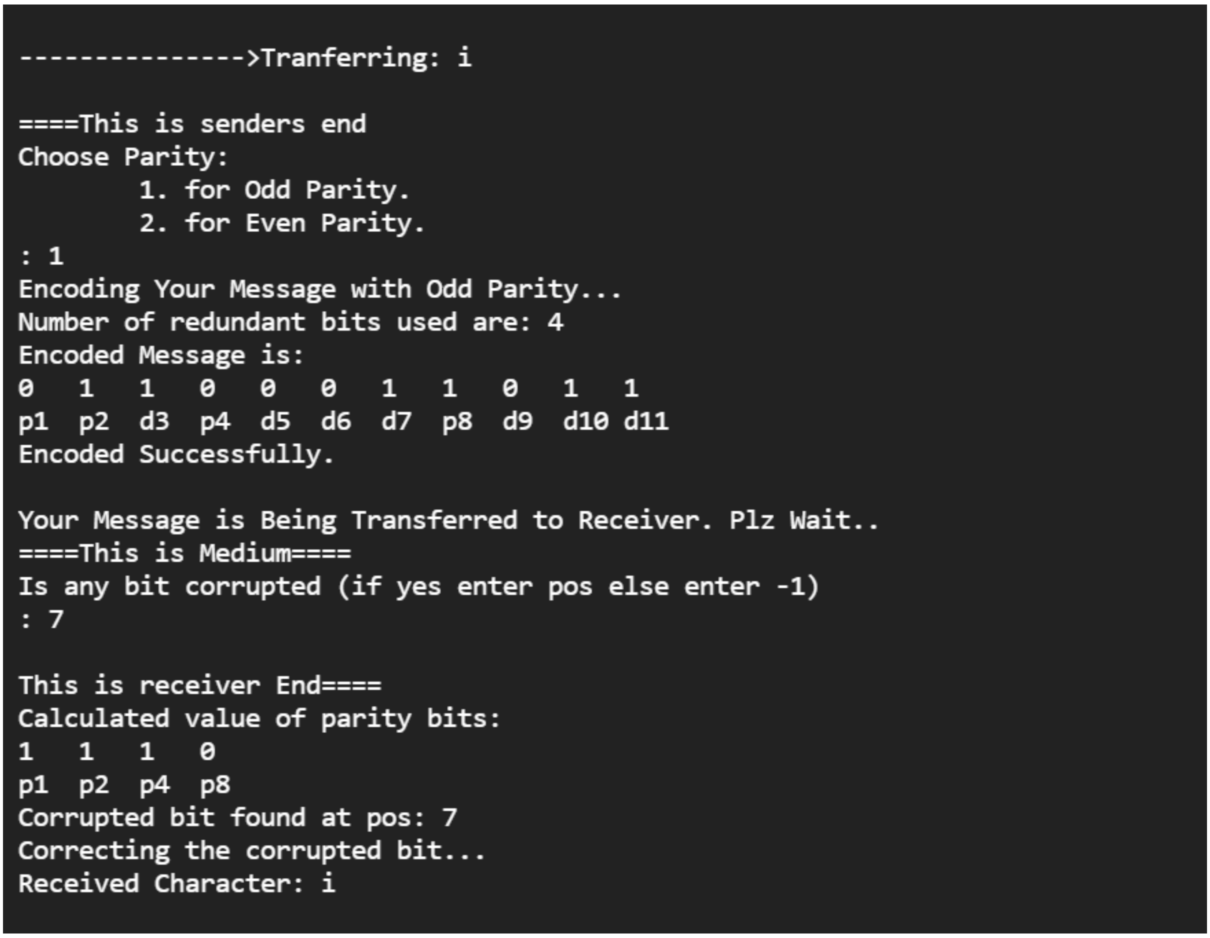
```
--------------->Tranferring: p

====This is senders end
Choose Parity:
        1. for Odd Parity.
        2. for Even Parity.
: 1
Encoding Your Message with Odd Parity...
Number of redundant bits used are: 4
Encoded Message is:
1   1   0   1   0   0   0   0   1   1   1
p1  p2  d3  p4  d5  d6  d7  p8  d9  d10 d11
Encoded Successfully.

Your Message is Being Transferred to Receiver. Plz Wait..
====This is Medium====
Is any bit corrupted (if yes enter pos else enter -1)
: -1

This is receiver End====
Calculated value of parity bits:
0   0   0   0
p1  p2  p4  p8
No Errors detected. Succeessfully Accepted.
Received Character: p
```

Screenshot-1: Odd Parity with no error



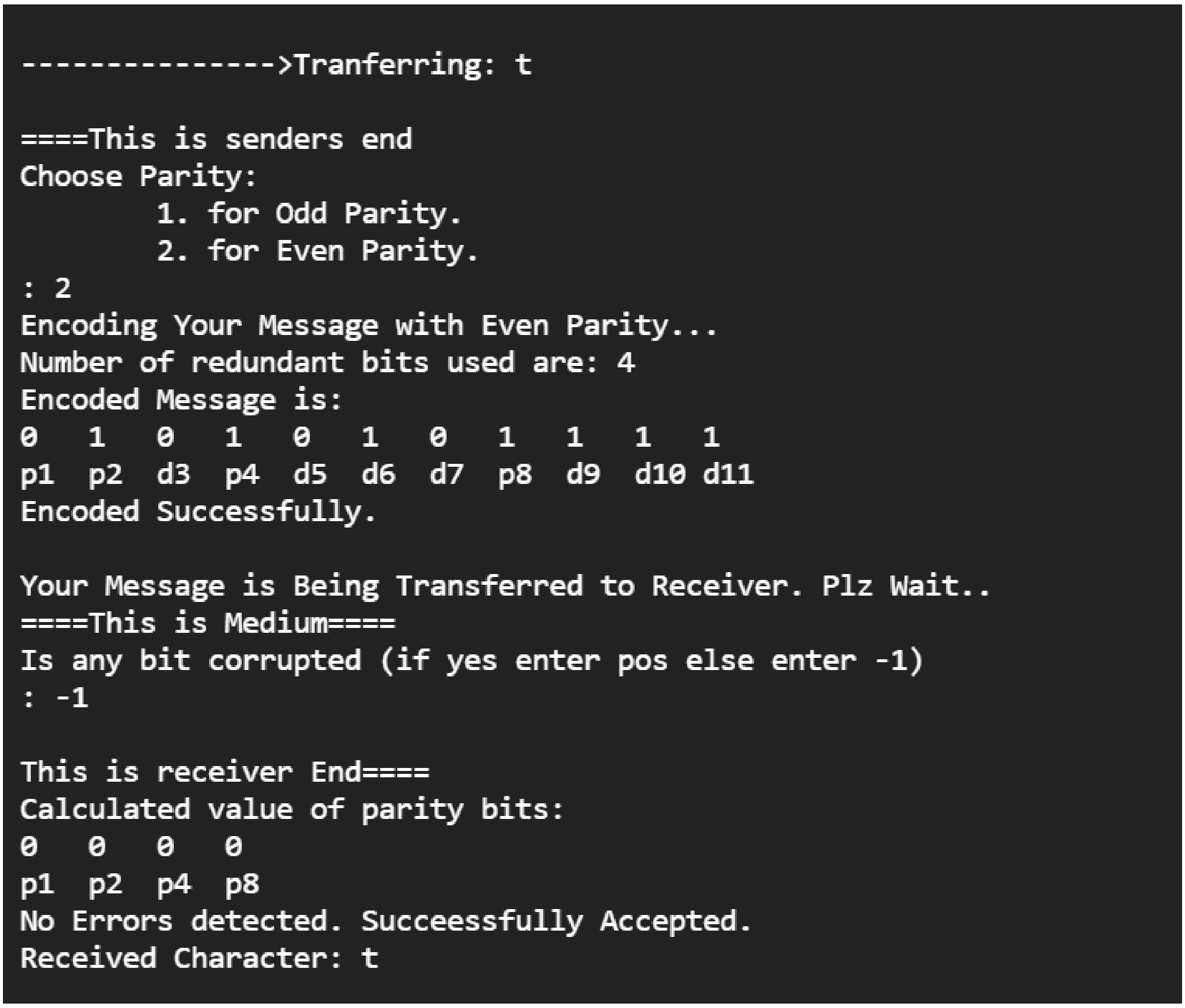
```
--------------->Tranferring: i

====This is senders end
Choose Parity:
        1. for Odd Parity.
        2. for Even Parity.
: 1
Encoding Your Message with Odd Parity...
Number of redundant bits used are: 4
Encoded Message is:
0   1   1   0   0   0   1   1   0   1   1
p1  p2  d3  p4  d5  d6  d7  p8  d9  d10 d11
Encoded Successfully.

Your Message is Being Transferred to Receiver. Plz Wait..
====This is Medium====
Is any bit corrupted (if yes enter pos else enter -1)
: 7

This is receiver End====
Calculated value of parity bits:
1   1   1   0
p1  p2  p4  p8
Corrupted bit found at pos: 7
Correcting the corrupted bit...
Received Character: i
```

Screenshot-2: Odd Parity with error



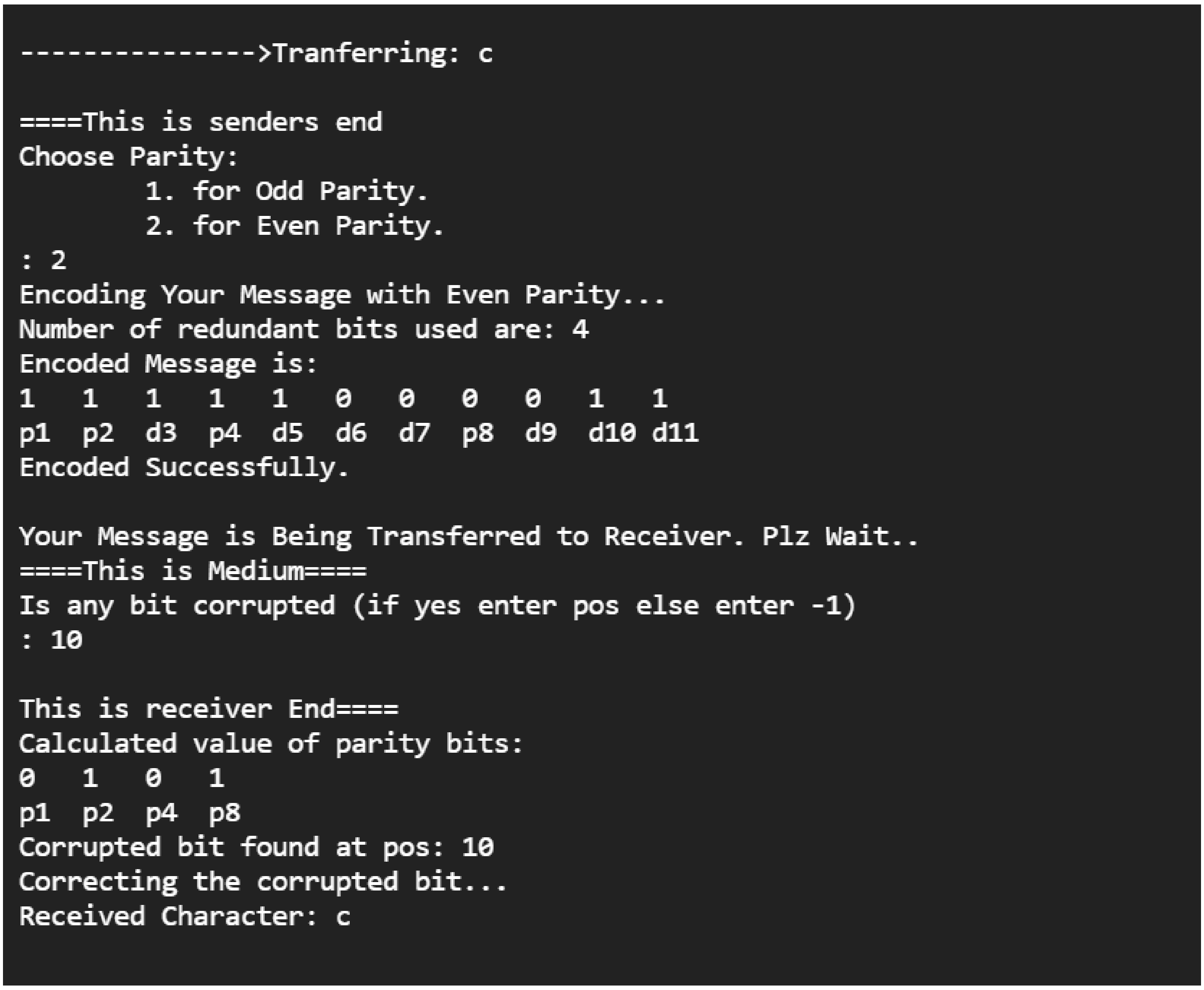
```
--------------->Tranferring: t

====This is senders end
Choose Parity:
        1. for Odd Parity.
        2. for Even Parity.
: 2
Encoding Your Message with Even Parity...
Number of redundant bits used are: 4
Encoded Message is:
0   1   0   1   0   1   0   1   1   1   1
p1  p2  d3  p4  d5  d6  d7  p8  d9  d10 d11
Encoded Successfully.

Your Message is Being Transferred to Receiver. Plz Wait..
====This is Medium====
Is any bit corrupted (if yes enter pos else enter -1)
: -1

This is receiver End====
Calculated value of parity bits:
0   0   0   0
p1  p2  p4  p8
No Errors detected. Succeessfully Accepted.
Received Character: t
```

Screenshot-3: Even Parity with no error



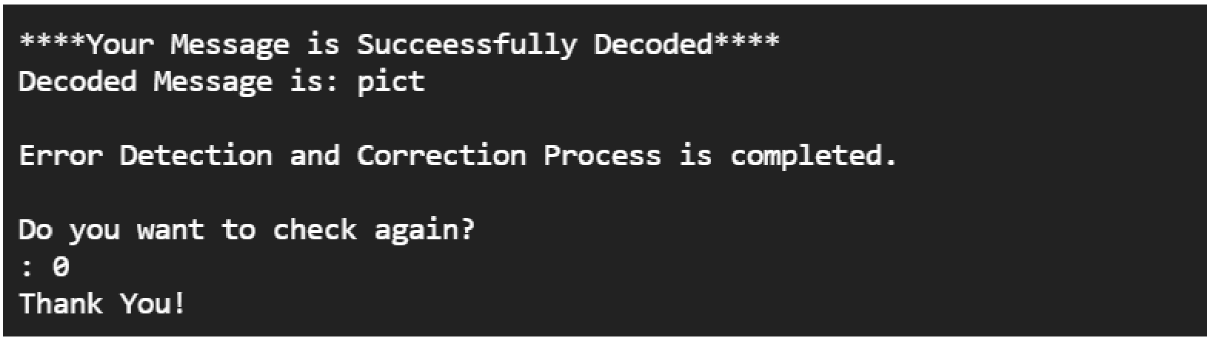
```
--------------->Tranferring: c

====This is senders end
Choose Parity:
        1. for Odd Parity.
        2. for Even Parity.
: 2
Encoding Your Message with Even Parity...
Number of redundant bits used are: 4
Encoded Message is:
1   1   1   1   1   0   0   0   0   1   1
p1  p2  d3  p4  d5  d6  d7  p8  d9  d10 d11
Encoded Successfully.

Your Message is Being Transferred to Receiver. Plz Wait..
====This is Medium====
Is any bit corrupted (if yes enter pos else enter -1)
: 10

This is receiver End====
Calculated value of parity bits:
0   1   0   1
p1  p2  p4  p8
Corrupted bit found at pos: 10
Correcting the corrupted bit...
Received Character: c
```

Screenshot-4: Even Parity with error



```
****Your Message is Succeessfully Decoded****
Decoded Message is: pict

Error Detection and Correction Process is completed.

Do you want to check again?
: 0
Thank You!
```

Thank You!