Exp. 6
16/8/24

Practical - 6

Error Correction and Data link layer

Aim:
Write a program to implement error detection & correction using Hamming code concept. Make a test run to input data stream & verify error correction features

Error correction at data link layer
Hamming code is a set of link error detection codes that can be used to detect & correct the errors than can occur when the data is transmitted from the sender to receiver. Developed by R.W hamming error detections

Create Sender Program:

1) Input to sender file should be a text of any length, Program should convert text to binary.

2) Apply hamming code on the binary data to check for errors

3) If there is an error convert
output is a file called channel

Create receiver program

1) receiver program should read
the input from channel file

2) Apply hamming code on the
binary data to check for errors

3) If there is an error, display
the position of the error

4) Else remove the redundant
bits and convert binary data to
ascii & display the output

students observation

Write the code here

```
# Sender py (filename)
import os
def text to binary (text)
return join ( format (ord(char), '08b')
for char in text)
```

```
def calculate redundant bits(m)
r=0
while (2**r) < (m+r+1)
r+=1
return r
def position redundant bits(data, r)
j=0
k=1
m=len (data)
res='';
for i in range (1, m+r+1)
if i==2**j:
res+='0'
j+=1
else
res+=data[k]
k+=1
return res[::-1]
def calculate parity bits(arr, r);
n = len(arr)
for i in range(r):
val=0
```

```
for i in range (n+r+1).
if j & (2**i) == (2**i);
val = val ^ int (arr[-j])
res = res + val * (10**i)
return int (str(res), 2)
def correct_error (arr pos);
if pos != 0:
arr = arr [: len[arr] - pos] +
str (1 - int (arr [len (arr) - pos])
return arr
def remove_redundant bits
n = len (arr)
j = 0
res = ''
for i in range (1, n+1)
if (i != 2**j)
res += arr[i]
else
j += 1
return res[:: -1]
```

```
def binary to text (binary data)
text = ''
for i in range (0, len (binary.data), 8);
byte = binary . data [i : i+8]
text += chr (int (byte, 2))
return text
```

output

Data transferred is 1010100110

Error Data is 11101001110.

The position of error is 2 from the left

[illegible] 16/8/24

Result: Thus the output is verified successfully.