21/8/24

Practical - 6

Aim :

To Write a Program to implement error detection and correction using Hamming code concept.

Error Correction at data link Layer:

Code:

```
import os
def text_to_binary (text):
"""convert text to binary representation."""
return ''.join (format (ord (c), '08b') for c in text
def hamming_code (data):
""" Encode binary data using hamming code"""
n = len (data)
r = 0
while (2**r < n + r + 1):
    r += 1
hamming code = ['0'] * (n+r)
j = 0
for i in range (n+r):
if (i+1) & (i) == 0:
    continue
hamming_code [i] = data [j]
j += 1
```

```
    for i in range(r):
        parity_bit_position = 2**i
        parity_bit = 0
        for j in range(1, len(hamming_code)+1):
            if (j & parity_bit_position) == parity_bit_p
                parity_bit ^= int(hamming_code[j-1])
        hamming_code[parity_bit-1] = str(parity_
    return ''.join(hamming_code)

def hamming_decode(data):
    """
    decode hamming code & check for error
    n = len(data)
    r = 0
    while(2**r < n+1):
        r += 1
    error_position = 0
    for i in range(r):
        parity_bit_position = 2**i
        parity_bit = 0
    for j in range(1, n+1):
        if (j & parity_bit_position) == parity_
                                        position
        parity_bit ^= int(data[j-1])
```

```
if paritybit != 0:
error_position += parity_bit position
if error position != 0:
    print (f"error detected at position:
          {error_position}")
    data = list (data)
    data [error_position - 1] = '1' if data[error
                          position - 1] == '0' else '0'
data = ''.join (data)

else:
   print ("No errors detected.")
   result = []
  for i in range (n):
    if (i+1) & (i) != 0:
       result.append (data[i])
   return ''.join (result)

def binary_to_text (binary data):
""" Convert binary to data to text. """
     text = ""
     for i in range (0, len (binary_data, 8):
Return !
```

Result:

Thus Hamming Code to implement error detection is studied

28/8/24