EX-6 EX NO 06 ERROR CORRECTION AT DATA LINK LAYER

DATE - 20.8.24

AIM -

Write a program to implement error detection & correction using hamming code concept make a test to input data stream & verify error correction feature

Error correction of data link uses

Hamming code is a set of error correction ~~code~~ codes that can be used to detect correct ~~error that~~ of errors that can occur when the data is transmitted from the sender to the receiver. It is a technique developed by R.W. Hamming for error correction

Create sender program with below feature:

1) Input to sender file should be a text of any length

2) Apply hamming code concept on the binary data & add redundant bits of it

3) Save this output in a file called channel

Create a receiver program with below feature

1) Receiver program should read the input from channel file

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

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

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

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

student is observation

code:-

```
# 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 bit (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 j in range(1, n+1)
  if j & (2**i) == (2**i)
   val = val ^ int(arr[-j])
  arr = arr[:n-(2**i)] +
              str(val) + arr[n-(2**
   return arr

def flip(data, pos):
    if pos < 1 or pos > len(data):
        print("invalid position")
        return data

    data_list = list(data)
    data_list[pos-1] = '1' if data_list[pos-1]
                                         == '0' else '0'
      return ''.join(data_list)

    def bin_to_dec(b):
        return int(b, 2)
        s = input("Enter a string to encode:")

  bin_val = ''.join([bin(ord(c))[2:]
                                 .zfill(8) for c
                                          in s])
  print(f"Binary representation of {s}
                                    : {bin_val}")
    l = len(bin_val)
    r = calc_r(l)
    print(f"Number of redundant
                                    bits: {r}
```

```
        pos = pos_red_bit(bin_val, n)
        enc_data = calc_parity(pos, n)
        print(f"Data with redundant
                bit: {enc_data}")

    while True
        err_pos = int(input(f"Enter the
                position of the bit to flip
                (1-based index, 1 to {len(enc_data)}): "))

        if err_pos in [2**i for i in range(n)]:
            print("Cannot flip a redundant
                bit position. Please enter
                a valid position")
            continue
        else
            enc_data_err = flip(enc_data,
                                err_pos)
            print(f"Data with error introduced
                : {enc_data_err}")
            break
    err_detected = detect_err(enc_data_err, n)
    if err_detected == 0:
        print("No error detected in
                received data.")

    else:
        err_pos_detected = err_detected
        err_pos_left = len(enc_data_err) -
        bin_err_pos = bin(err_pos_detected,
                          pos_left)[2:].zfill(0
        dec_err_pos = bin_to_dec(bin_err_pos)
        print(f"Error detected at position:
                {err_pos_left}")

    print(f"Binary error position: {bin_err_pos}
            Decimal position: {dec_err_pos}")
```

correct = input ("Do you want to correct the error ? (Yes/no): ").strip().lower()

```
if correct == 'yes':
    corrected_data = flip(enc_data, err_pos_left)
    print(f"corrected data : {corrected_data}")
else:
    print("error was not corrected.")
```

OUTPUT

Enter a string to encode: Hi

Binary representation of 'Hi':

0100100001101001

Number of redundant bits : 5

Data with redundant bit :

010010000110010000100

Enter a position of the bit to flip (1-based index (1 to 21): 9

cannot flip a redundant bit position. please enter a valid position

Enter the position of the bit to flip (-1 based index (1 to 21): 6

Data with error introduced :

010011000011001000100

Error detected at position 6

Binary error position: 00110, Decimal position: 6

Do you want to correct the error ? (Yes/no): yes

corrected data : 010010000110010000100

Result

Thus the Hamming error correction of Data link ...