Exp No: 6. Hamming Code programs for error detection and correction:

Aim:-
write a program to implement error detection and correction using HAMMING code concept make a test sum to input data stream and vality error correction feature.

Error Correction at Data link layer:
Hamming code is a set of error Correction Code that can be used to detect and correct that error that can occur when the data is transmitted from the sender to the reciever. 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 program should convert the text to binary.

2, Apply hamming code concept on the binary data and add redundant bits to it.

3, Save this output in a file called channel.

create a receiver program with below features.

1, Receiver program should Read input from Channel file.

2. Apply hamming Code on binary data to check for error.

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

4. Else remote redudant bits and convert the binary data to ascii and display the o/p.

code,

```
Sender.py.
def char_to_binary (ch).

    """convert a character to
    8-bit binary string""".
    return format (ord(ch), '08b').
def hamming_encode (data 4):
    d1, d2, d3, d4 = [int [bit] for.
                        bit in data4].
    p1 = d1 ^ d2 ^ d4.
    p2 = d1 ^ d3 ^ d4.
    p4 = d2 ^ d3 ^ d4.
return f" {p1} {p2} {d1} {p4} {d2}.
                        {d3} {d4}.
```

```
text = input ("Enter text:")
with open ("channel.txt", "w") as
    for ch in text ~~ef~~ :
        bin_ch = char_to_binary
                              (ch).
    for i in range (0, 8)
        code = hamming_encode
              [bin_ch [i : i4]
            f. write (code)
    print ("data with to channel.
              txt with hammingcode
receiver.py .
    def hamming_decode (code_s):
      b = [0] + [int (bit) for bit in
                          code7 ]
      p1 = b[1] ^ b [3] ^ b [5] ^ b [7] .
      p2 = b[2 ] ^ b[3] ^ b[6] ^ b[7]
      p4 = b [4] ^ b[5] ^ b[6] ^ b[7].
    error_pos = p1 * 1 + p2 * 2 + p4 * 4 .
   if error_pos! = 0
        print (f" error detected at
            position {error_ias}
                correcting ...").
    b[error-pos] ^= 1
    d1,d2,d3,d4 = b[3], b[5], b[6], b[7]
    return f"{d1}{d2}{d3}{d4}".

binary result = "".
withopen("channel.text", "r") as f:
    code = f.read()
for i in range (0, len (codes), 7):
    binary-result += hamming-decode (code[i:i+7])
text = "".
for i in range (0, len (binary-result), 8):
    byte = binary-result [i:i+8]
    text += chr(int (byte,2)).
print ("receive text after error correction :", text).
```

output:-

```
Enter 4-bit data : 1011
  sender side :- 0010011
Enter bit position to introduce error : 0
  Receiver side : 0010011
No error detected.
original Data bit extracted : 1011
```

Result:-

Hence the code is sucessfully executed and completed.