HAMMING CODE

AIM:

To write a program to implement error detection & correction using HAMMING CODE. Make test run to input data stream & verify error correction feature.

ERROR CORRECTION AT DATA LINK LAYER:

Hamming code is a set of error-correction codes that can be used to detect & correct the errors that can occur when the data is transmitted from the sender to receiver.

CREATE SENDER PROGRAM WITH BELOW FEATURES.

1) Input to sender file should be a text of any length.
2) Apply hamming code on binary data & add redun-dant bit to it
3) Serve this output in file called channel.

RECEIVER:

1) Receiver should read the input from channel
2) Apply hamming code on the binary to check for errors
3) If error, display position
4) Else, remove redundants & convert to ascii & display output.

STUDENT OBSERVATION:

```
def calc_parity_partitions(m):
    r = 0
    while (2**r) < (m+r+1)
        r+=1
    return r
def generate_hamming_code(data):
    n = len(data)
    r = calc_parity_partitions(m)
    n = m+r
    hamming = ['0'] - *(n+1)
    j = 0
    for j in range (1, n+1):
        if j & (i-1) == 0:
            continue
        hamming[i] = data[j]
        j+=1
for j in range (r):
    pos = 2** i
    parity = 0
    for j in range (1, n+1):
        if j & pos & j!= pos:
        parity ^= int (haroming[j])
        hamming[pos] = str(parity)
return .join (hamming[1:])
```

```
    r = calc_parity_positions(n + loop(n), count("1")
    error_pos = 0
    for j in range(r):
        pos = 2**i
        parity = 0
        for j in range (1, n+1):
            if(j & pos):
                parity ^= int(hamming[j])
            if parity != 0:
                error - pos += pos
    if error - pos != 0
        print(f"Error detected at bit position: {error - pos}")
        hamming[error_pos] = '1' if
        hamming[error_pos] == '0' else '0'
    else:
        print ("No error detected")

data = "1011"
print ("original data:", data)
hamming code = generate_hamming_code (data)
print("Hamming Encoded data:", hamming_code)
error_pos = 3
hamming_with_error = list (hamming_code)
hamming_with_error[error_pos-1] = '1' if
hamming_with_error [error_pos-1] == '0'
else '0'
```

```
corrected_code = detect_and_correct(hamming_with_error)
print("corrected hamming code:", corrected_code)
```

OUTPUT:

```
Original Data : 1011
Hamming Encoded data: 0110011
Hamming Code with error: 0100011
Error detected at bit position: 3
Corrected Hamming Code: 0110011
```

RESULT:

Hamming Code written & implemented successfully