Date:
7/8/25

PRACTICAL - 6

Aim:

Write a program to implement error detection and correction using HAMMING CODE concept. Make a test run to input data stream and 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 and correct the 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 the below features:

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 ~~recommen~~ 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 the input from channel file.

2. Apply hamming code on the binary data to check errors.

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

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

Student Observation

Code:

```
def cale-parity (data):
    C = [0, 0, 0, data[0], 0, data[1], data[2], data[3]]
    C[1] = C[3] ^ C[5] ^ C[7]
    C[2] = C[3] ^ C[6] ^ C[7]
    C[4] = C[5] ^ C[6] ^ C[7]
    return C[1:]

def detect_correct(r):
    r = [0] + r
    S1, S2, S4 = r[1] + r[3] ^ r[5] ^ r[7], r[2] ^ r[3]
            ^ r[6] ^ r[7], r[4] ^ r[5] ^ r[6] ^ r[7]
    crr_pos = S4 * 4 + S2 * r + S1
if err_pos:
        print(f'Error at bit {err_pos}')
        r[err_pos] ^ = 1
            print("corrected = ", r[1:])
else
```

```
    print("No error detected")
    return r[1:]
def extract_data(c):
    return [c[2], c[4], c[5], c[6]]
if __name__ == "__main__":
    data = input("Enter 4-bit data")
    if len(data) != 4 or any(b not in "01" for b in data):
        exit("invalid input")
    data = [int(b) for b in data]
    enc = calc_parity(data)
    print("Encoded:", enc)
    if input("Introduced error? (y/n):").lower()=='y':
        p = int(input("Enter position 1-7:"))
        if 1 <= p <= 7:
            enc[p-1] ^= 1
    print("Received:", enc)
    corr = detect_correct(enc)
    print("original data:", extract_data(corr))
```

output:

```
Enter 4-bit data: 1011
Encoded: [0, 1, 1, 0, 0, 1, 1]
Introduce error? (y/n): y
Enter position (1-7): 3
Recived: [0, 1, 0, 0, 0, 1, 1]
```

Error at bit 3

corrected = [0, 1, 1, 0, 0, 1, 1]

original data [1, 0, 1, 1]

Result:

Hence the code for Hamming problem was successful executed.

9/10/24