EXP NO : 06

DATE : 20.8.24

HAMMING CODE

AIM :

Write a program to check error detection and correction using hamming code. Make a test run to input data stream & verify error correction feature.

PROCEDURE : SENDER

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 redudant bits to it
3. Save this output in a file called channel

RECIEVER :

1. Reciever program should read input from Channel file
2. Apply hamming code :

CODE:

```
def string_to_binary(input_string):
    return ''.join(format(ord(char), '08b') for char in input_string)

def binary_to_string(binary_data):
    chars = []
    for i in range(0, len(binary_data), 8):
        byte = binary_data[i:i+8]
        chars.append(chr(int(byte, 2)))
    return ''.join(chars)

def calculate_parity_bits(data):
    n = len(data)
    r = 0
    while (2**r) < (n + r + 1):
        r += 1
    return r

def insert_parity_bits(data, r):
    n = len(data)
    j = 0
    k = 0
    m = n + r
    hamming_code = []
    for i in range(1, m + 1):
        if i == 2**k:
            hamming_code.append('0')
```

```
            j += 1
        else:
            hamming_code.append(int(data[k]))
            k += 1
    return hamming_code

def detect_and_correct_error(hamming_code, r):
    n = len(hamming_code)
    error_position = 0
    for i in range(r):
        parity_pos = 2**i
        parity_val = 0
        for j in range(1, n+1):
            if j & parity_pos:
                parity_val ^= hamming_code[j-1]
        if parity_val != 0:
            error_position += parity_pos
    if error_position
        print(f"Error at: {error_position}")
        hamming_code[error_position-1] ^= 1
        print("Corrected Hamming code:
                {hamming_code}")
    else:
        print("No error Detected")
```

(0) buggo.

```
written hamming code

def main():
    input_string = input("enter a string")
    binary_data = string_to_binary(input_
                                   string)
    print(f"Binary is '{input_string}':
                {binary_data}")
    r = calculate_parity_bits(binary_data)
    hamming_code = insert_parity_bits(binary_data,
                                      r)
    hamming_code = calculate_parity_values(hamming_
                                           code, r)
    print(f"hamming code: {hamming_code}")
    print("\n introduce error")
    error_bit = int(input(f"error the bit
                position (1- {len(hamming_code)}):
    hamming_code[error_bit-1] ^= 1 #interna
    print(f"hamming code with error:
                {hamming_code}")
```

```
hamming_code = detect_and_error(hamming_code, r)
corrected_string = binary_to_string(corrected_binary_data)
print(f"final output : {corrected_string}")

if __name__ == "__main__":
    main()
```

OUTPUT:

```
Enter a string : hi
Binary Representation : 0110100001101001
hamming code with parity : [0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1]
Enter the bit for error : 2
hamming code with error :
[0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1]
corrected_hamming_code :
: array([0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1])
final output : hi
```

Result

Thus the program is successfully executed & the output is verified