21/8/24 PRACTICAL - 6.

# HAMMING CODE.

AIM: Write a hamming code, program in python to detect the error from the recieved data and the position.

PROGRAM:

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

def calcRedundantBits(m):
    for i in range(m):
        if 2**i >= m+i+1:
            return i

def posRedundantBits(data, r):
    j = 0
    k = 1
    m = len(data)
    res = ''
    for i in range(1, m+r+1):
        if i == 2**j:
            res = res + '0'
            j += 1
        else:
            res = res + data[-1*k]
            k += 1
    return res[::-1]

def calcParityBits(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[-1*j])
    arr = arr[:n - 2(2**i)] + str(val) +
                arr[n - (2**i) + 1:]
    return arr
def detectError(arr, nr):
    n = len(arr)
    res = 0
    for i in range(nr):
        val = 0
        for j in range(1, n+1):
            if j & (2**i) == (2**i):
                val = val ^ int(arr[-1*j])
        res = res + val * (10**i)
    return int(str(res), 2)
def correctError(received_data, error_position):
    if error_position == 0:
        return received_data
    data_list = list(received_data)
    error_index = len(received_data) - error_position
    corrected_bit = '0' if data_list[error_index] == '1'
                    else '1'
    return ''.join(data_list)
```

```
def main():
    word = input("Enter a word to be transmitted
    binary_data = text_to_binary(word)
    print(f"Binary representation of '{word}'
                is {binary_data}").
    m = len(binary_data)
    r = calcRedundantBits(m).
    arr = posRedundantBits(binary=data, r)
    arr = calcParityBits(arr, r)
    print("Data transferred is "+arr)
    received_data = input("Enter the received
                data (with one errors):")
if not all(bit in '01' for bit in data):
    print("Invalid input"):
    return.
correction = detectError(received_data, r)
    if correction == 0:
        print("There is no error in the
received data"):
else:
    corrected_data = correctError(received
                data, correction)
    print.("The position of error is,los
```

```
(received_data - correction+1, "from the left").
print("Corrected data is " + corrected_data)
if __name__ == '__main__':
    main()
```

Output:

Enter a word to be transmitted : hello

Binary representation of hello:

01101000011001010110110001101100011011

Data transmitted:

01101000011001010110110001101100011011

Enter recieved data (with possible errors):

01101000011001010100110001101100011011

The position of error is 20 from left.

Corrected data:

01101000011001010110110001101100011011

Result: Hamming code for detecting the position of error in recieved data is successfully executed and output is verified.

18/9/24