7/8/24

PRACTICAL - 6 : HAMMING CODE

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 receiver. It is a technique developed by R.W Hamming for error correction.

CODE:

```
def string-to-binary (input-string):
    return ''.join (format (ord (c), '08b') for c 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 ** j:
            hamming_code.append('0')
            j += 1
        else:
            hamming_code.append(int(data[k]))
            k += 1
    return ''.join(map(str, hamming_code))

def calculate_parity_values(hamming_code, r):
    hamming_code = list(map(int, hamming_code))
    n = len(hamming_code)
    for i in range(r):
        parity_pos = 2 ** i
        parity_val = 0
        for j in range(1, n+1):
            if j & parity_pos and j != parity_pos:
                parity_val ^= hamming_code[j-1]
        hamming_code[parity_pos-1] = parity_val
    return ''.join(map(str, hamming_code))

def detect_and_correct_error(hamming_code, r):
    hamming_code = list(map(int, hamming_code))
    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("Error at: {error_position}")
    print("Binary pos: {bin(error_position)[2:].zfill(r)}")
    hamming_code[error_position-1] ^= 1
    print("corrected code: {''.join(map(str, hamming_code))}").
else:
    print("No error")
return ''.join(map(str, hamming_code))

def extract_data_from_hamming(hamming_code):
    j=0
    data = []
    for i in range(1, len(hamming_code)+1):
        if i != 2**j:
            data.append(hamming_code[i-1])
        else:
            j+=1
    return ''.join(map(str, data))

def main():
    input_string = input("Enter the String:")
    binary_data = string_to_binary(input_string)
    print("Binary: {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("Hamming code: {hamming_code}")
    redundant bits: {2**i for i in range(r)}
```

```
while True:
    print("\n Flip a bit for error ....")
    error_bit = int(input(f" Flip Bit (1-{len(hamming_code)}):"))
    if error_bit in redundant_bits:
        print("Redundant bit. choose another position").
    else :
        hamming_code = hamming_code[error_bit-1] + ('1' if
                hamming_code[error_bit-1] == '0' else '0') +
                hamming_code[error_bit]

        print(f"Hamming code with error: {hamming_code}")
        break
    hamming_code = detect_and_correct_error(hamming_code, r)
    corrected_data = extract_data_from_hamming(hamming_code, r)
    corrected_string = binary_to_string(corrected_data)
    print(f" Final output: '{corrected_string}'")
if __name__ == "__main__":
    main().
```

OUTPUT:

```
Enter the string: Hi
Binary : 01001000 01101001
Hamming code : 0100 1001 1000 0110 1001
Flip a bit for error...
Flip bit (1-21) : 2
Redundant bit ! choose another option
Flip bit for error !...
Flip bit (1-21) : 3
Error !
Hamming code with error : 0110 1001 1000 0110 1001
corrected code : 0100 1001 1000 0110 1001
```

Error at : 3
Binary pos : 0011

RESULT: Thus program for Hamming code for implementing error detection and correction is successfully executed and output is verified.

9/10/24