8/10/24

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 & correct the errors that can occur when the data is transmitted from the sender to the receiver. It is a technize developed by R.W Hamming for error correction.

Create sender program with 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 add 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 for 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 avoid and display the out put.

PROGRAM 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)
    i = 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 hamming_code

def calculate_parity_value(hamming_code, r):
    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 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 detected at position: {error_position}")
        hamming_code[error_position-1] ^= 1
        print(f"Corrected Hamming Code: {hamming_code}")
    else:
        print("No error detected")
```

```
return hamming_code.

def atract_data_from_hamming (hamming_code,r):

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

def_main()
    input_string = "Mysdf Vijay"
    binary_data = strong_to_binary (input_string)
    Print (f"Binary representation of '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 ("Hamming code with parity_value(hamming_code)
    Print ("To introduce a single-bit error for
            demonstration....:").
    error_bit= Int(input ("Enter the bit position (1-ilen
            (hamming_code)}) to introduce an error:")
    Print ("Hamming code with error: {hamming_code}")
    hamming_Code = detect_and_correct_error(hamming_code)
    Corrected_binary_data = extract_data_from_hamming
                                (hamming_code)
    Correcte_string = binary_to_string (corrected_
                                binary_data)
    Print ("Final output after correcting Hamming
            Code: {corrected_string}")
```

```
if __name__ == "__main__":
    return ().
```

OUTPUT:

```
Enter the string: Hi
Binary    : 01001000 01101001
Hamming cod: 01001001 10000 1101001
Flip a bit for error ...
Flip bit (1-21): 2
Redundant but close another position
Flip a bit for error ...
Flip bit (1-21): 3
Error
Hamming code with error: 01101001 10000 1100100
Error at 3
Binary pos: 00011
Corrected code: 01001001 10000 1100100
Final output : Hi
```

RESULT:

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