16/8/24 Exercise: 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.

Error Correction at Data Link Layer:-

Hamming code is set of error correction codes that can be used to detect and correct the errors that can 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 below features:

1) Input to sender file should be a text any length. Program should convert the text to binary.

2) Apply hamming code concept on the binary data and add redundant bits to it.

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 error.

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

Student Observation :-

Program :-

```
def calc_r(l):
    for i in range(l):
        if(2**i >= l+i+1):
            return i

def par_red_bits(b,r):
    j,k = 0,1
    bits = ""

for i in range(1, len(b)+r+1):
    if(i == 2**j):
        bits += "0"
        j += 1
    else:
        bits += b[-1*k]
        k += 1
    return bits[::-1]

def calc_parity(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**i)] + str(val) + arr[n-(2**i)+1:]
    return arr.
```

```
def flip(data, pos):
    if pos<1 or pos>len(data):
        print("Invalid position!")
        return data
    data_list = list(data)
    data_list[pos-1] = '1' if data_list[pos-1] == '0' else '0'
    return ''.join(data_list)

def bin_to_dec(b):
    return int(b, 2)
s = input("Enter a string to encode: ")

bin_val = ''.join([bin(ord(c))[2:].zfill(8) for c in s])
print(f"Binary representation of '{s}': {bin_val}")

l = len(bin_val)
r = calc_r(l)
print(f"Number of redundant bit: {r}")

pos = pos_red_bits(bin_val, r)
enc_data = calc_parity(pos, r)
print(f"Data with redundant bits: {enc_data}")

while True:
    err_pos = int(input("Enter the position of the bit to
        flip (1-based index, 1 to {len(enc_data)}): "))
```

```
if err_pos in [2**i for i in range(r)]:
    print("Cannot flip a redundant bit position. Please
          enter a valid position:")
    continue
else:
    enc_data_err = flip(enc_data, err_pos)
    print(f"Data with error introduced: {enc_data_err}")
    break

err_detected = detect_err(enc_data_err, r)
if err_detected == 0:
    print("No error detected in the received data.")
else:
    err_pos_detected = err_detected
    err_pos_left = len(enc_data_err) - err_pos_detected + 1
    bin_err_pos = bin(err_pos_left)[2:].zfill(4)
    dec_err_pos = bin_to_dec(bin_err_pos)
    print(f"Error detected at position: {err_pos_left}")
    print(f"Binary error position: {bin_err_pos},
          Decimal position: {dec_err_pos}")
    correct = input("Do you want to correct the error? (yes/no):")
        .strip().lower()
    if correct == 'yes':
        corrected_data = flip(enc_data_err, err_pos_left)
        print(f"Corrected data: {corrected_data}")
    else
        print("Error was not corrected.")
```

Output:-

Enter a string to encode: Hi

Binary representation of 'Hi': 0100100001101001

Number of redundant bits: 5

Data with redundant bits: 0100 100000 1100100100

Enter the position of the bit to flip (1 to 21): 4

Cannot flip a redundant bit position. Please enter a valid position.

Enter the position of the bit to flip (1 to 21): 6

Data with error introduced: 0100110000 1100100100

Error detected at position: 6

Binary error position: 0110, Decimal position: 6

Do you want to correct the error? (yes/no): yes

Corrected data: 0100 100000 1100100100

=== Code Execution Successful ===

Result:-

Thus, the error detection and correction using Hamming code concept was successfully implement and Executed.

16/8/24