CS23532-COMPUTER NETWORKS-LAB MANUAL
Practical -6

Name: Tharunraj
RegNo:230701362

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 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 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 ascii and display the output.

Program:
Sender.py

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

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

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

```
def calc_parity_bits(arr, r):
    n = len(arr)
    arr = list(arr)
    for i in range(r):
        val = 0
        for j in range(1, n + 1):
            if j & (2**i) and arr[-j] == '1':
                val ^= 1
        arr[-(2**i)] = str(val)
    return ''.join(arr)

data = input("Enter text: ")
bin_data = text_to_bin(data)
r = calc_redundant_bits(bin_data)
arr = pos_redundant_bits(bin_data, r)
arr = calc_parity_bits(arr, r)

with open("channel.txt", "w") as f:
    f.write(arr)
print("Data sent to channel.txt:", arr)
```

receiver.py

```
def detect_error(arr, r):
    n = len(arr)
    res = 0
    for i in range(r):
        val = 0
        for j in range(1, n + 1):
            if j & (2**i) and arr[-j] == '1':
                val ^= 1
        res += val * (10**i)
    return int(str(res), 2)


def bin_to_text(bin_data):
    chars = [bin_data[i:i+8] for i in range(0, len(bin_data), 8)]
    return ''.join(chr(int(b, 2)) for b in chars if len(b) == 8)


with open("channel.txt") as f:
    arr = list(f.read().strip())
```

```
r = 0
while (2**r < len(arr) + 1):
    r += 1
error_pos = detect_error(arr, r)
if error_pos == 0:
    print("No error detected.")
else:
    print("Error detected at bit position:", error_pos)
    arr[-error_pos] = '1' if arr[-error_pos] == '0' else '0'
# Remove redundant bits
j, corrected = 0, ''
for i in range(1, len(arr)+1):
    if i != 2**j: corrected += arr[-i]
    else: j += 1
print("Received Text:", bin_to_text(corrected[::-1]))
```

Input:-

Enter text: Hi

Output:

No error detected.
Received Text: Hi

RESULT:

Thus the program to implement error detection and correction using HAMMING code has been executed successfully.