Exp no: 6
Date: 2/8/24

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.

Program:

```
def text_to_binary(text):
    binary_data = ''
    for char in text:
        binary_data += ''.join(str((ord(char) >> i) & 1) for i in range(7, -1, -1))
    return binary_data

def encode_data(binary_data):
    encoded_data = ''
    for bit in binary_data:
        encoded_data += bit * 3
    return encoded_data

def sender_program(text):
    binary_data = text_to_binary(text)
    file = open('channel.txt', 'w')
    file.write(encoded_data)
    print("Encoded data saved to 'channel.txt': " + encoded_data)

def change_bit_in_file(file_path):
    file = open(file_path, 'r')
    binary_data = file.read().strip()
    print("current data: " + binary_data)

while True:
    try:
        bit_position = int(input("Enter the position of the bit to change (1- " +
str(len(binary_data)) + "): "))
```

```
        if 1 <= bit_position <= len(binary_data):
            break
        else:
            print("Invalid position. Please try again.")
    except ValueError:
        print("Invalid input. Please enter a number.")

binary_list = list(binary_data)
if binary_list[bit_position - 1] == '0':
    binary_list[bit_position - 1] = '1'
else:
    binary_list[bit_position - 1] = '0'
modified_data = ''.join(binary_list)
file = open(file_path, 'w')
file.write(modified_data)
print("Bit at position " + str(bit_position) + " changed successfully")

def remove_redundant_bits(data, redundancy):
    return ''.join([data[i] for i in range(0, len(data), redundancy)])

def binary_to_text(binary_data):
    binary_values = []
    for i in range(0, len(binary_data), 8):
        binary_values.append(binary_data[i:i+8])
    text = ''.join([chr(int(val, 2)) for val in binary_values])
    return text

def detect_error(data, redundancy):
    for i in range(0, len(data), redundancy):
        bit = data[i]
        if not all(b == bit for b in data[i:i+redundancy]):
            print("Error detected")
            return
    print("No error detected")
```

```
def receiver_program():
    file = open('channel.txt','r')
    received_data = file.read().strip()
    detect_error(received_data,3)
    corrected_data = remove_redundant_bits(received_data,3)
    text = binary_to_text(corrected_data)
    print("Decoded txt : "+text)
text = input("Enter a text : ")
sender_program(text)
print()
change_bit_in_file('channel.txt')
print()
receiver_program()
```

O/P:

Enter text : data.
Data has been encoded and saved to 'channel' file.

Receiver.py

Error detected at position : 9

Error is corrected.

Result:

Thus the program was successfully executed and the o/p is verified.

14/9/24