EXP NO: 6

DATE:

Pratical 6

Hamming Code

AIM:

Program to implement error detection & correction using hamming code concept. Make a test run to input data stream & verify error correction feature.

Create sender prg with below features.

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 send to receiver.

Create receiver prg with below features.

- receiver prg should read the I/P from channel file.
- Apply hamming code on binary data to check for errors.
- If there is an errors, display the position of the error.
- else remove the redundant bits & convert the binary & convert binary data to ascii & display the output.

STUDENT

Code :

def tex

ret

def bin

ch

re

def get

r

co

re

def ha

r

s

n

a

J

fo

for

STUDENT OBSERVATION:

Code:

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

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

def get_redudant_bits_length(m):
    r=1
    while (2**r)<(m+r+1):
        r+=1
    return r

def hamming_encode(data):
    m = len(data)
    r = get_redudant_bits_length(m)
    n = m+r
    arr = ['0'] * (n+1)
    j=0
    for i in range(1,n+1):
        if (i & (i-1))==0:
            arr[i]='0'
        else:
            arr[i]= data[j]
            j+=1
    for i in range(r):
        parity_pos = 2**i
```

```
        parity_pos = 2**i
        parity = 0
        for k in range(1, n+1):
            if k & parity_pos:
                parity ^= int(arr[k])
        arr[parity_pos] = str(parity)
    return ''.join(arr[1:])

def hamming_decode(data):
    n = len(data)
    r = 0
    while 2**r < n+1:
        r += 1
    arr = ['0'] + list(data)
    error_pos = 0
    for i in range(r):
        parity_pos = 2**i
        parity = 0
        for k in range(1, n+1):
            if k & parity_pos:
                parity ^= int(arr[k])
        if parity != 0:
            error_pos += parity_pos

    if error_pos != 0:
        print(f"Error detected at position: {error_pos}")
        arr[error_pos] = '1' if arr[error_pos] == '0'
                         else '0'
```

```
else:
    print
decoded_b
for i in
    if

return

def sender(
    text =
    binary
    encod
    block
    with

    prin

def recei
```

```
        else:
            print("No error detected.")

        decoded_bits = []
        for i in range(1, n+1):
            if (i & (i-1)) != 0:
                decoded_bits.append(arr[i])
        return ''.join(decoded_bits)

def sender():
    text = input("enter txt to send:")
    binary_data = text_to_binary(text)
    encoded_data = ''
    blocks = split_chunks(binary_data, 4)
    with open("channel.txt", 'w') as f:
        f.write(encoded_data)
    print("Data encoded & saved to channel.txt")

def receiver():
    with open("channel.txt", "r") as f:
        received_data = f.read().strip()
    decoded_data = ''
    blocks = split_chunks(received_data, 7)
    for block in blocks:
        corrected = hamming_decode(block)
        decoded_data += corrected
    decoded_data = decoded_data[:len(decoded_data) - (len(decoded_data) % 8)]
    text = binary_to_text(decoded_data)
```

print ("Received text after error correction:")
print (text)

Sender()
Receiver()

Input:
enter data bits : 1011
generated hamming code: 0110011

Input (No error):
enter received code to check : 1110011

output:
No error detected in received code.

Input: (with error)
enter received code to check : 0110011

output:
Error detected at bit positions: 4
Corrected code : 0110011

Result

Hence the pgm on Hamming code is executed & implemented successfully.

Exp No: 07

AIM: