	Bansilal Ramnath Agarwal Charitable Trust's Vishwakarma Institute of Information Technology **Department of Artificial Intelligence and Data Science**

Name: Siddhesh Dilip Khairnar		
Class: SY	Division: B	Roll No: 272028
Semester: IV	Academic Year: 2022-2023	
Subject Name & Code: Fundamentals of Computer Networks: ADUA22203		
Title of Assignment: Write a program for error detection and correction for 7/8 bits ASCII codes using Hamming Codes		

ASSIGNMENT NO. 5

FCN Assignment NO 5

Name: siddhesh Dilip Khairnar
Rollno: 272028
PRN no: 22110398
Division: B

* Problem Defination: - Write a program for error detection and correction for 7/8 bits ASCII codes or CRC.

* Prerequisite:
1. Data link layer: Roles, Protocols (Ethernet)
2. c/c++ programming syntax
3. Wireshark Tool.

* Theory: →

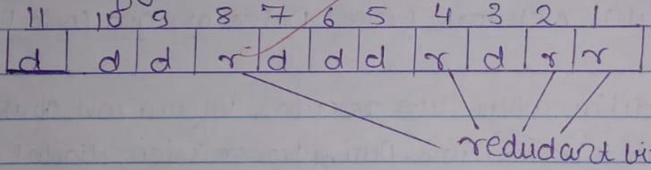
Computer Network Error Detection and Correction

Error: A condition when ~~the~~ receiver's information does not matches with the sender's information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from sender to receiver. That means a 0 bit may change to 1 or a 1 bit may change to 0.

Some popular techniques for error detection are:
1. Simple parity check
2. Two-dimensional Parity check
3. Checksum
4. Cyclic redundancy check

1. Hamming code :- It is a set of error-correction codes that can be used to detect and correct bit error that can occur when computer data is moved or stored.

Calculating the Hamming code :-
Determining the Position of redundacy bits we know that to detect error in a 7 bit code.
4 redundant bit are required.
Now, the next task is to determines the position at which the data unit is present.
• These redundancy bit are placed at the position which correspond to the power of 2.
• for eg. in case of 7 bit data, 4 dedundancy bit are required. so making total no. of bit as 11. The redundancy bit are placed in position 1, 2, 4 and 8 as shown in figure.



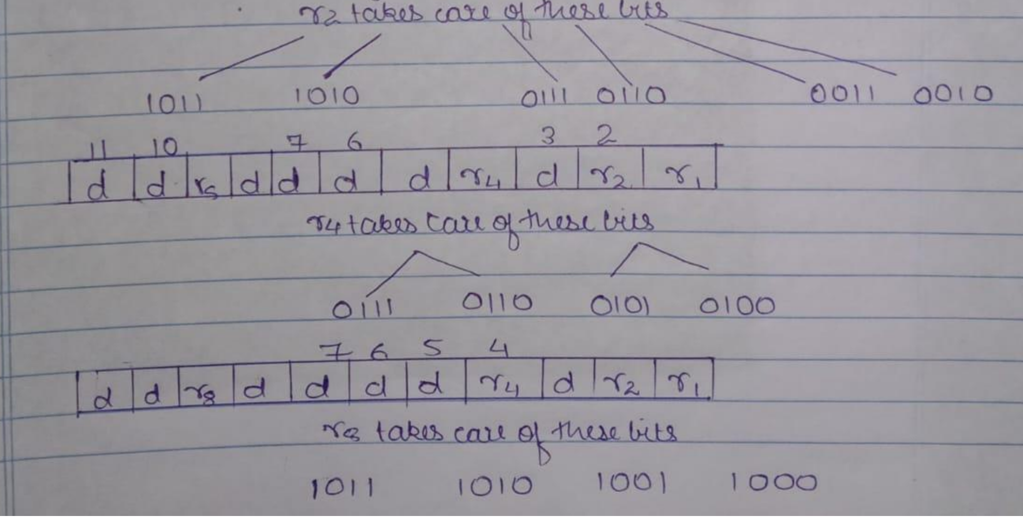
Position of redundancy bits in Hamming codes

- In Hamming code, each r bit is the VRC for one combination of data bits. r1 is the VRC bit for one combination of data bit. r2 is the VRC for another combination of data bit, and so on.
- Each data bit may be included in more than one VRC calculation.
- r1 bit is calculated using all bits position whose binary representation includes a 1 in the rightmost position.
- bit calculated using all the bit position with a 1 in the second position and so on.
- Therefore the various r bit are parity bit for different combination of bits.

The various combination are:→
r1: bits 1,3,5,7,9,11
r2: bits 2,3,6,7,10,11
r4: bits 4,5,6,7
r8: bits 8,9,10,11

<u>Example of Hamming code Generation:-</u>

Redundancy bits calculation



suppose a binary data 1001101 is to be transmitted.
To implement hamming code for this following step as used:

1) Calculating the no. of redundancy bits required. Since number of data bit
is 7, the value of r is calculated as.

$2^r \geq m+r+1$

$2^4 \geq 7+4+1$

no. of redundancy bit = 4.

2) Determining the Position of various data bit The various r bit are placed at the position that correspond to the power of 2 that 1, 2, 4, 8.

Conclusion: Hence, we have studied and were able to understand the concept of error detection and correction using Hamming codes & CRC.

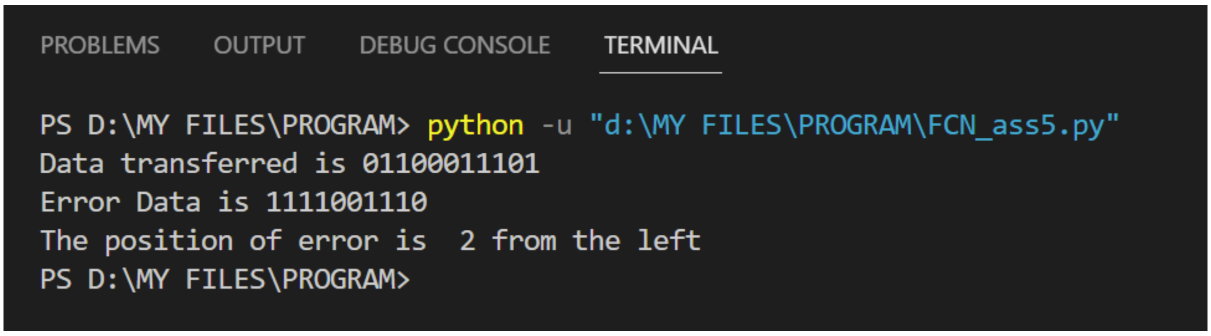
Ⓒ

[illegible]
2/4/23

Program and Output:

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

def postRedundantBits(data, r):
    j = 0
    k = 0
    m = len(data)
    res = ''

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

def calcParityBits(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 detectError(arr, nr):
    n = len(arr)
    res = 0

    for i in range(nr):
        val=0
        for j in range(1,n + 1):
            if(j & (2**i) == (2**i)):
                val = val ^ int(arr[-1 * j])
        res = res + val*(10**i)

    return int(str(res), 2)

data = '1011001'
```

```
m = len(data)
r = calcRedundantBits(m)
arr = postRedundantBits(data, r )
arr = calcParityBits(arr, r )
print("Data transferred is " + arr)
arr = '1111001110'
print("Error Data is " + arr)
correction = detectError(arr, r)
if(correction==0):
    print("There is no error on the recieved message.")
else:
    print("The position of error is ",len(arr)-correction+1,"from the
left")
```



```
PS D:\MY FILES\PROGRAM> python -u "d:\MY FILES\PROGRAM\FCN_ass5.py"
Data transferred is 01100011101
Error Data is 1111001110
The position of error is  2 from the left
PS D:\MY FILES\PROGRAM>
```

Conclusion: Hence, we have studied and were able to understand the concept of error detection and correction using hamming codes