COMPUTER NETWORKS

EXPERIMENT 4

NAME: VARUN VISWANATH

SAPID: 60004210105

BRANCH: B1 COMPS

AIM:

To implement CRC and Hamming Code as error detection and correction codes.

THEORY:

Hamming code is a type of error-correcting code used in digital communication and data storage systems to detect and correct errors that may occur during data transmission. It was developed by Richard Hamming in 1950 and is named after him.

The code is designed to add redundancy to data bits by adding extra bits known as parity bits. These bits allow the receiver to detect and correct single-bit errors in the transmitted data. Hamming code is capable of correcting up to one-bit errors and detecting up to two-bit errors.

The code works by dividing the data bits into groups and adding parity bits to each group. The position of the parity bits is determined by their binary representation, with each bit representing a power of 2. The parity bits are calculated by checking the corresponding data bits and adding 1 for every bit that is set to 1.

Hamming code has been widely used in computer memory systems, such as RAM and flash memory, as well as in digital communication systems like satellite communication, wireless communication, and Ethernet. It has been proven to be an effective way to detect and correct errors in data transmission, making it an important tool in modern communication and computing systems.

CODE:

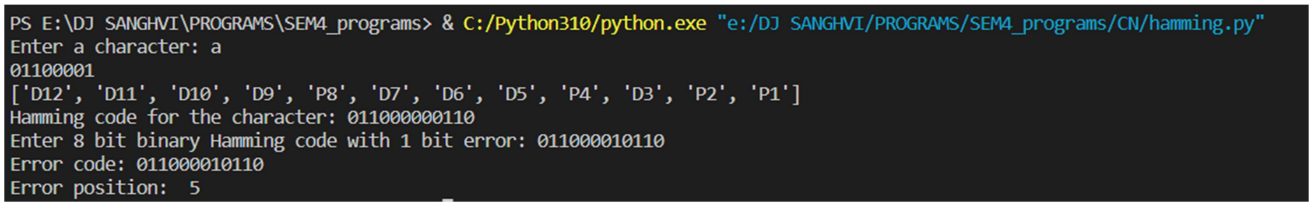
```
sentence = input("Enter a character: ")
binary1 = ""
for ch in sentence:
    binary1 += '0'+str(bin(ord(ch)))[2:]

#binary1 = '01100001'
binary = binary1[::-1]
print(binary1)

r=0
while 2**r < len(binary)+r+1:
    r = r + 1
b=0
hamming_list = []
value_list = []
m = 0
for i in range(12):
    if i+1 == 2**b:
        hamming_list.append('P'+str(i+1))
        value_list.append('X')
        b +=1
    else:
        hamming_list.append('D'+str(i+1))
        value_list.append(binary[m])
```

```
        m+=1


hamming_list.reverse()
print(hamming_list)


P1 = [3,5,7,9,11]
P2 = [3,6,7,10,11]
P4 = [4,5,6,7,12]
P8 = [8,9,10,11,12]
c1 = 0
c2 = 0
c4 = 0
c8 = 0
pairs = [(P1,c1),(P2,c2),(P4,c4),(P8,c8)]


def checkParity(pair):
    P,c = pair
    for index in P:
        if value_list[index-1] == '1':
            c+=1
    return P,c


for pair in pairs:
    P,c = checkParity(pair)
    set_parity = '1'
```

```
    if c%2==0: set_parity='0'


    if P==P1:
        value_list[0] = set_parity
    elif P==P2:
        value_list[1] = set_parity
    elif P==P4:
        value_list[3] = set_parity
    elif P==P8:
        value_list[7] = set_parity

val_disp = value_list.copy()
val_disp.reverse()
print("Hamming code for the character: "+"".join(val_disp))

bin_error = input("Enter 8 bit binary Hamming code with 1 bit error: ")
#bin_error = '011000010110'
print("Error code: "+bin_error)
bin_error = bin_error[::-1]
cb1 = [1,3,5,7,9,11]
cb2 = [2,3,6,7,10,11]
cb3 = [4,5,6,7,12]
cb4 = [8,9,10,11,12]

check_bits = []
```

```
cb_all = [cb1,cb2,cb3,cb4]

sum = 0

for cb in cb_all:
    for index in cb:
        sum += int(bin_error[index-1])
    if sum%2==0: check_bits.append('0')
    else: check_bits.append('1')
    sum=0


check_disp = check_bits.copy()

check_disp.reverse()

error_pos = int("".join(check_disp),2)

print("Error position: ",error_pos)
```

OUTPUT:



THEORY:

Cyclic Redundancy Check (CRC) is a type of error-detecting code commonly used in digital communication and data storage systems. It was first introduced in the early 1960s and has since become an important tool in modern communication and computing systems.

The code works by adding a checksum value to the data being transmitted or stored. The checksum value is calculated based on the binary data using a mathematical algorithm known as polynomial division. The resulting checksum is appended to the original data, forming the transmitted or stored message.

At the receiver end, the same polynomial division algorithm is used to calculate a new checksum value based on the received message. If the calculated checksum value matches the transmitted checksum value, the receiver can be confident that the data has not been corrupted during transmission. If the checksum values do not match, an error is detected, and the data can be retransmitted or discarded.

CRC is widely used in computer networks, such as Ethernet and Wi-Fi, as well as in storage systems like hard drives and flash memory. It is capable of detecting a wide range of errors, including both single-bit and multi-bit errors, making it an effective method of error detection. The strength of the CRC algorithm can be adjusted by choosing different polynomial values, allowing it to be customized for different applications.

CODE:

```
dividend = list(int(ch) for ch in list(input("Enter dividend\n")))

divisor = list(int(ch) for ch in list(input("Enter divisor\n")))

for i in range(len(divisor) - 1):

    dividend.append(0)

if len(divisor) > len(dividend):

    print("Invalid input!")

remainder, lst = [], dividend[0:len(divisor)]

for i in range(len(dividend) - len(divisor) + 1):

    if (lst[0] == 1):

        comp = divisor
```

```
    else:
        comp = list(0 for j in range(len(divisor)))
    for j in range(len(divisor)):
        if (lst[j] == comp[j]):
            remainder.append(0)
        else:
            remainder.append(1)
    lst = remainder[1:]
    if (i != ((len(dividend) - len(divisor)))):
        lst.append(dividend[i + len(divisor)])
    remainder.clear()
print("Remainder is = ",end="")
print("".join(map(str,lst)))
```

OUTPUT: