COMPUTER NETWORKS
LAB 4

NAME - KAPAROTU VENKATA SURYA THARANI
USN - 22BTRAD018
BRANCH - AIDE

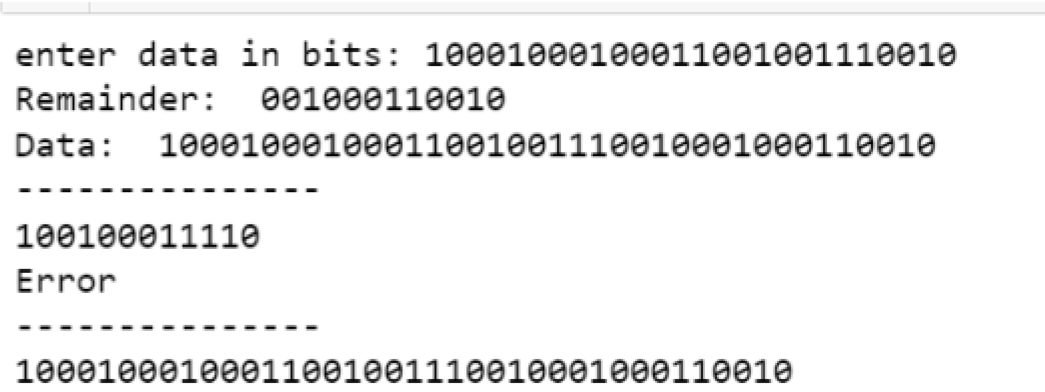
CRC Code Computation

1. Write a program to compute the CRC code for the polynomial CRC-12.

CODE:

```
class CRC:
    def __init__(self):
        self.cdw = ''
    def xor(self,a,b):
        result = []
        for i in range(1,len(b)):
            if a[i] == b[i]:
                result.append('0')
            else:
                result.append('1')
        return ''.join(result)
    def crc(self,message, key):
        pick = len(key)
        tmp = message[:pick]
        while pick < len(message):
            if tmp[0] == '1':
                tmp = self.xor(key,tmp)+message[pick]
            else:
                tmp = self.xor('0'*pick,tmp) + message[pick]
            pick+=1
        if tmp[0] == "1":
            tmp = self.xor(key,tmp)
        else:
            tmp = self.xor('0'*pick,tmp)
        checkword = tmp
        return checkword
    def encodedData(self,data,key):
        l_key = len(key)
        append_data = data + '0'*(l_key-1)
        remainder = self.crc(append_data,key)
        codeword = data+remainder
        self.cdw += codeword
        print("Remainder: " ,remainder)
        print("Data: " ,codeword)

    def reciverSide(self,key,data):
        r = self.crc(data,key)
        size = len(key)
        print(r)
        if r==size*0:
            print("No Error")
        else:
            print("Error")

data = input('enter data in bits: ')
#crc-12 polynomial
key = '1100000001111'
c = CRC()
c.encodedData(data,key)
print('---------------')
c.reciverSide(c.cdw,key)
print('---------------')
print(c.cdw)
```

OUTPUT:



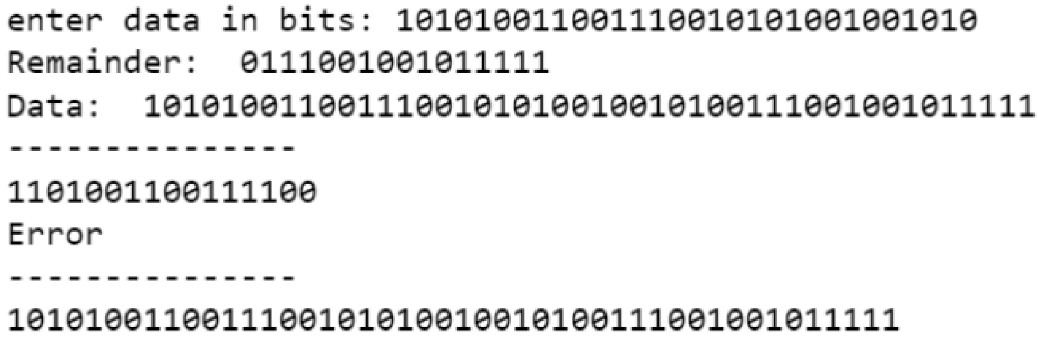
```
enter data in bits: 10001000100011001001110010
Remainder:  001000110010
Data:  10001000100011001001110010001000110010
---------------
100100011110
Error
---------------
10001000100011001001110010001000110010
```

2. Write a program to compute the CRC code for the polynomial CRC-16.

CODE:

```
class CRC:
    def __init__(self):
        self.cdw = ''
    def xor(self,a,b):
        result = []
        for i in range(1,len(b)):
            if a[i] == b[i]:
                result.append('0')
            else:
                result.append('1')
        return ''.join(result)
    def crc(self,message, key):
        pick = len(key)
        tmp = message[:pick]
        while pick < len(message):
            if tmp[0] == '1':
                tmp = self.xor(key,tmp)+message[pick]
            else:
                tmp = self.xor('0'*pick,tmp) + message[pick]
            pick+=1
        if tmp[0] == "1":
            tmp = self.xor(key,tmp)
        else:
            tmp = self.xor('0'*pick,tmp)
        checkword = tmp
        return checkword
    def encodedData(self,data,key):
        l_key = len(key)
        append_data = data + '0'*(l_key-1)
        remainder = self.crc(append_data,key)
        codeword = data+remainder
        self.cdw += codeword
        print("Remainder: " ,remainder)
        print("Data: " ,codeword)

    def reciverSide(self,key,data):
        r = self.crc(data,key)
        size = len(key)
        print(r)
        if r==size*0:
            print("No Error")
        else:
            print("Error")

data = input('enter data in bits: ')
#crc-12 polynomial
key = '1100000001111'
c = CRC()
c.encodedData(data,key)
print('---------------')
c.reciverSide(c.cdw,key)
print('---------------')
print(c.cdw)
```

OUTPUT:



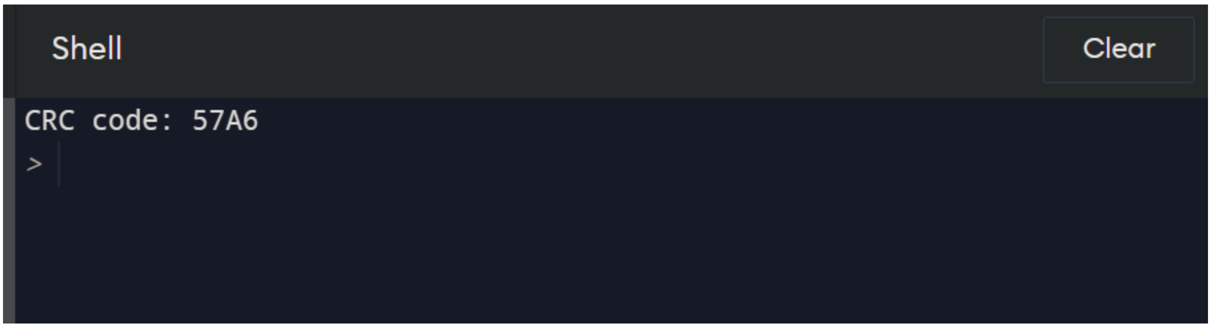
```
enter data in bits: 101010011001110010101001001010
Remainder:  0111001001011111
Data:  1010100110011100101010010010100111001001011111
---------------
1101001100111100
Error
---------------
1010100110011100101010010010100111001001011111
```

3. Write a program to compute the CRC code for the polynomial CRC CCIP.

CODE:

```
def crc_ccitt(data):
    polynomial = 0b11000000000000101
    crc = 0xFFFF

    #convert data to bytes
    data_bytes = data.encode()


    # Perform division
    for byte in data_bytes:
        # XOR the CRC register with the next data byte
        crc ^= (byte << 8)

        # Perform a bitwise XOR operation with the polynomial for each bit
        for _ in range(8):
            if crc & 0x8000:
                crc = (crc << 1) ^ polynomial
            else:
                crc = crc << 1
            crc &= 0xFFFF
    crc_str = hex(crc)[2:].zfill(4).upper()
    return crc_str

# Example usage
data = "Hello, world!"  # Data for CRC calculation
crc = crc_ccitt(data)
print("CRC code:", crc)
```

OUTPUT:



```
CRC code: 57A6
>
```