

```

=====
Name      :Lalita Sharma
RNo       :06
=====

```

Experiment :03

Aim:Write a program to implement CRC algorithm. Ask the user to input variable length data and generator polynomial. Find the CRC code on the basis of input data. Invert any bit of CRC frame and detect the error at receiver side.

code:

```

def xor(a, b):
    xor = []
    # Traverse all bits, if bits are same, then XOR is 0, else 1
    # starting from 1 beacuse we are here in xor only if first bit is 1
    for i in range(1, len(b)):
        if a[i] == b[i]:
            xor.append('0')
        else:
            xor.append('1')
    result=''.join(xor)
    return result
# *****
def mod2div(divident, divisor):
    # Number of bits to be XORed at a time.
    leng = len(divisor)
    # Slicing the divident to appropriate length for particular step
    tmp = divident[0 : leng]

    while leng < len(divident):

        if tmp[0] == '1':

            # replace the divident by the result of XOR and pull 1 bit down
            tmp = xor(divisor, tmp) + divident[leng]

        else:
            # If leftmost bit is '0' If the leftmost bit of the divident
            #(or the part used in each step) is 0, the step cannot
            # use the regular divisor; we need to use an all-0s divisor.
            tmp = xor('0'*leng, tmp) + divident[leng]

        # increment length to move further
        leng += 1

    # For the last n bits, we have to carry it out normally as increased value
    #of pick will cause Index Out of Bounds.
    if tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0'*leng, tmp)
    checkword = tmp
    return checkword
# *****
def RoughCRCcode(msg,generatorpolynomial):
    toaddbits='0' *(len(generatorpolynomial)-1)
    return msg+toaddbits
# *****

```

```

def CRCcode(msg,generatorpolynomial,roughCRCcode):
    data=msg # copy original msg i.e without crc wala in data
    remainder = mod2div(roughCRCcode, generatorpolynomial)

    # Append remainder in the original data
    finaldata = data + remainder
    print("Remainder : ", remainder)
    print("Encoded Data (Data + Remainder) : ",
          finaldata)
    return finaldata
# *****

def CRCreceiver(dataR,generatorpolynomial):
    remainder = mod2div(dataR, generatorpolynomial)
    print("Remainder at receiver=",remainder)
    if remainder=='0'*(len(generatorpolynomial)-1):
        print("No error")
    else:
        print("Error!!!!")
# *****

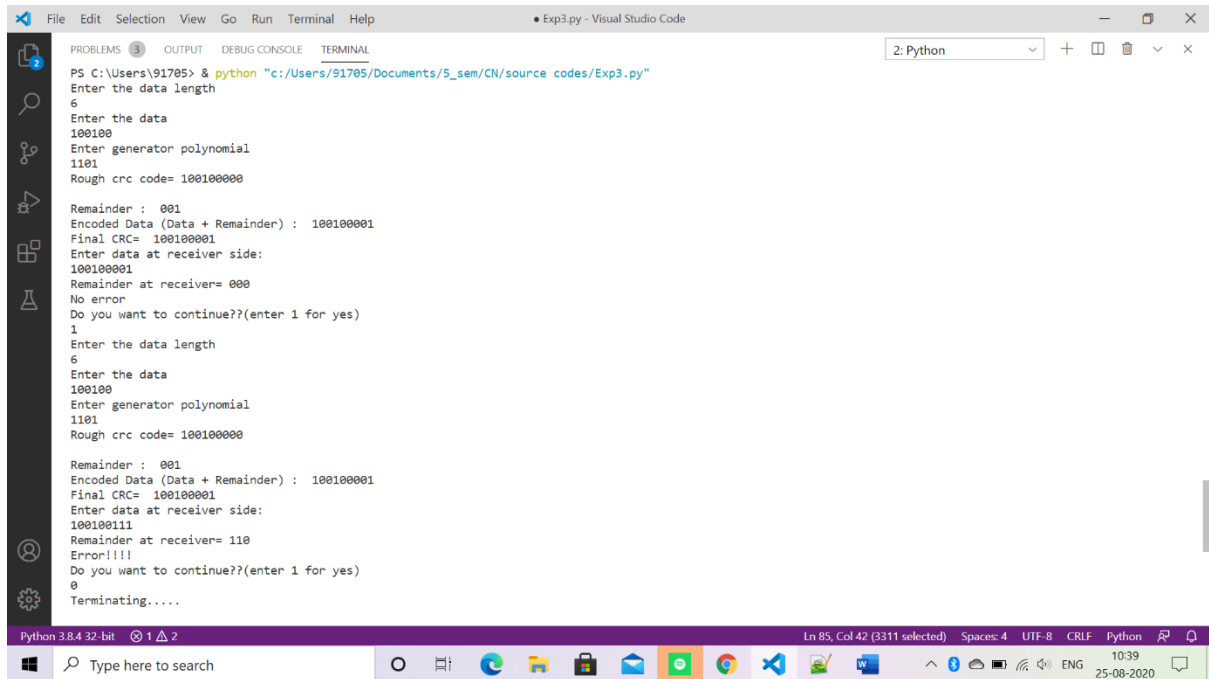
if __name__ == "__main__":
    i=1
    while(i==1):
        msglength=int(input("Enter the data length\n"))
        msg=input("Enter the data\n")
        generatorpolynomial=input("Enter generator polynomial\n")
        roughCRCcode=RoughCRCcode(msg,generatorpolynomial)
        print("Rough crc code= {}".format(roughCRCcode))
        finalcrc=CRCcode(msg,generatorpolynomial,roughCRCcode)
        print("Final CRC= ",finalcrc)
        dataR=input('Enter data at receiver side:\n')
        CRCreceiver(dataR,generatorpolynomial)

        i=int(input("Do you want to continue??(enter 1 for yes)\n"))
        if i!=1:
            print("Terminating.....\n\n")

```

OUTPUT

+++++



```
PS C:\Users\91705> & python "c:/Users/91705/Documents/5_sem/CN/source codes/Exp3.py"
Enter the data length
6
Enter the data
100100
Enter generator polynomial
1101
Rough crc code= 100100000

Remainder : 001
Encoded Data (Data + Remainder) : 100100001
Final CRC= 100100001
Enter data at receiver side:
100100001
Remainder at receiver= 000
No error
Do you want to continue??(enter 1 for yes)
1
Enter the data length
6
Enter the data
100100
Enter generator polynomial
1101
Rough crc code= 100100000

Remainder : 001
Encoded Data (Data + Remainder) : 100100001
Final CRC= 100100001
Enter data at receiver side:
100100111
Remainder at receiver= 110
Error!!!!
Do you want to continue??(enter 1 for yes)
0
Terminating.....
```

***** END *****