EXP-3

CN - C32 - 2103164

AIM - Implementation of CRC(Cyclic Redundany Check)

- Theory :- CRC is method of detcting error in communication channel. Given K bit frame or message, the transmitter generates a n bit sequence known, as frame check sequence (FCS), so that resulting frame, consisting of (K+n) bits.

- Bits sequences can be written as polynomials with coefficient 0 and 1.

- frame with K bits is considered as polynomial of degree K-1.

- The most significant bit is coefficient of x^{K-1} The next bit is coefficient of x^{K-2}. Example: The bit sequence 10011010 corresponds to this polynomial:

- Sending and recieving message can be imagined as exchange of polynomials

$M(x) = 1 * x^7 + 0 * x^6 + 0 * x^5 + 1 * x^4 + 1 * x^3 + 0 * x^2 + 1 * x^1 + 0 * x^0$

$= x^7 + x^4 + x^3 + x^1$

- The Data link Layer Protocal specifies a generator polynomial C(x). Generator polynomial is available for both sender and reciever side.

CN-C32-2103164

- C(x) is a polynomial of degree K
- if eg

$C(x) = x^3 + x^2 + x^0$

$= 1101$, then $K = 3$

- Therefore, generator polynomial is degree 3.

- The degree of generator polynomial is equal to , of bits minus one.

- If for a frame, the CRC need to be calculated, are appended to the frame.

- n corresponds to degree of generator polynomial

generator polynomial	100110

- The generator polynomial has 6 digits
  - Therefore, five 0 bits are appended.

Frame (payload)	10101
Frame with appended 0 bit	1010100000

CN - C32 - 2103164

Sender Side (Generation of Encoded Data from Data and Generator Polynomial (or Key)):

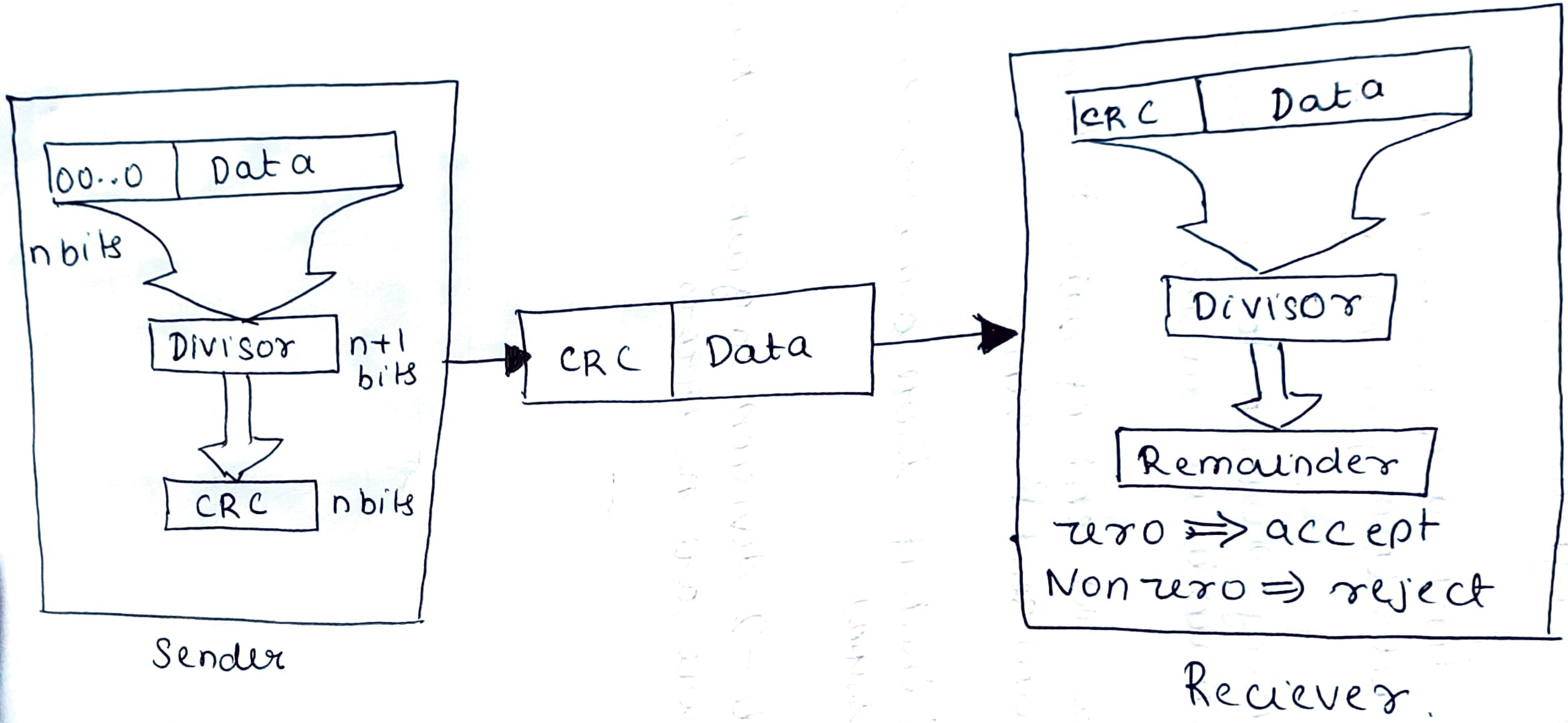
- binary data is first argumented by adding n zeros in end of the data (n degree of generator polynomial)

- Use modulo-2 binary division to divide binary data by Key and store remainder of division.

- Append remainder at end of data to form the encoded data and send the same.

Reciever Side (Check if there are error introduced in transmission)

- Perform modulo-2 division again if the remainder is 0, then there are no errors



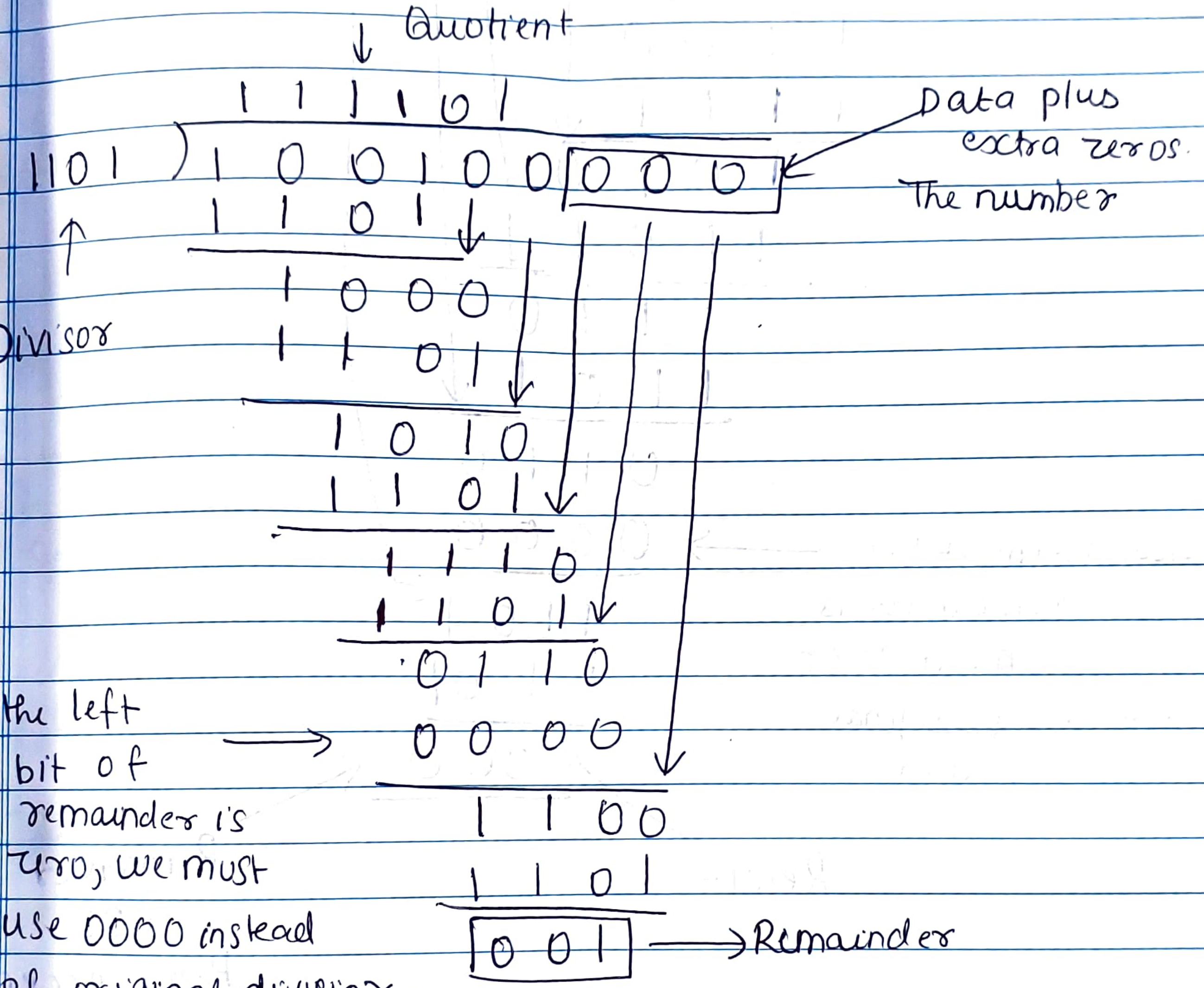
CRC Cyclic Reducany

CN-C32-2103114

CN-C32-2103114

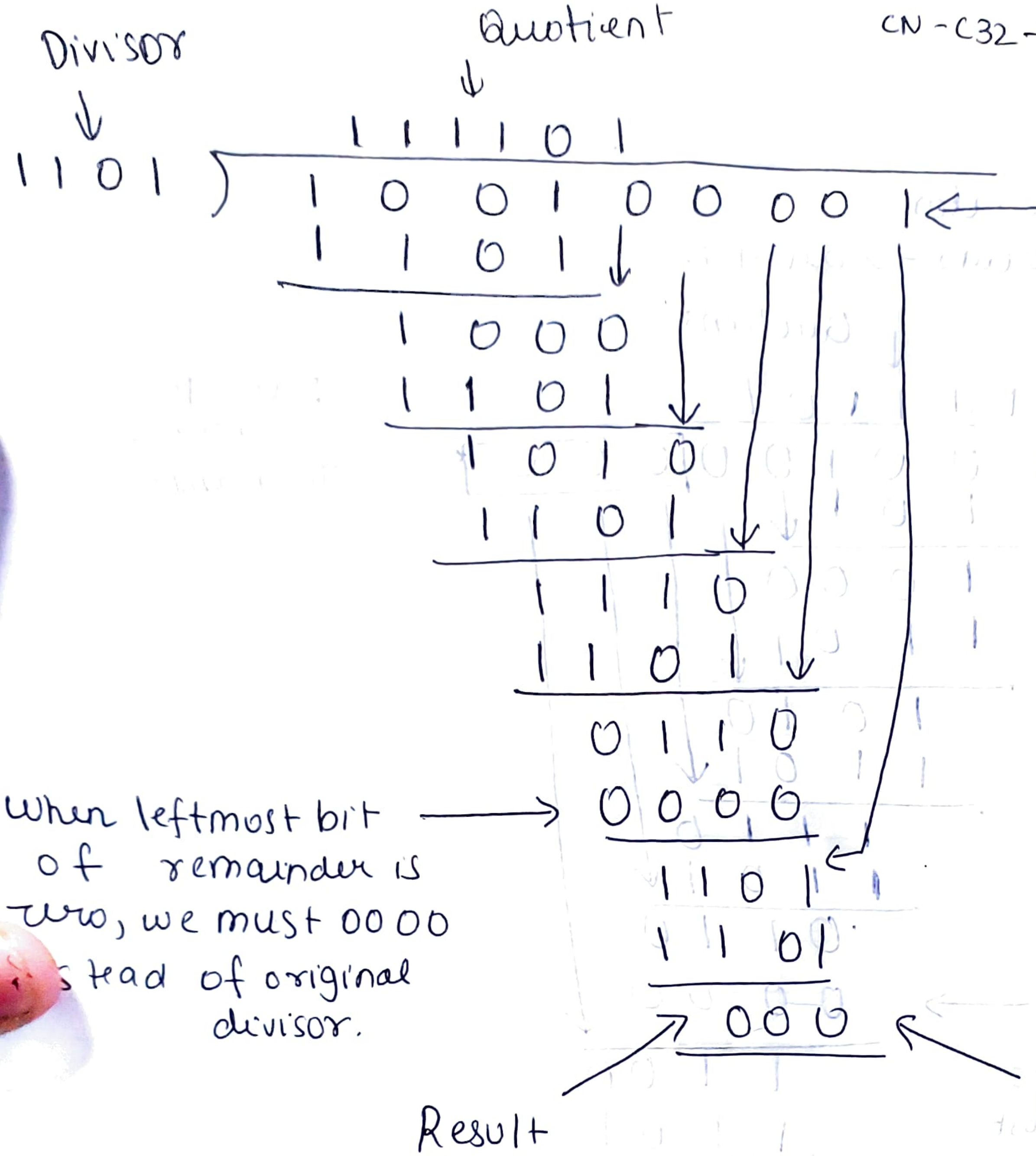
Eg:-

⇒ CRC generator uses modular-2 division



The left bit of remainder is zero, we must use 0000 instead of original divisior.

CN-C32-2103164



When leftmost bit of remainder is zero, we must 0000 instead of original divisor.

Remainder 000

Hence acc

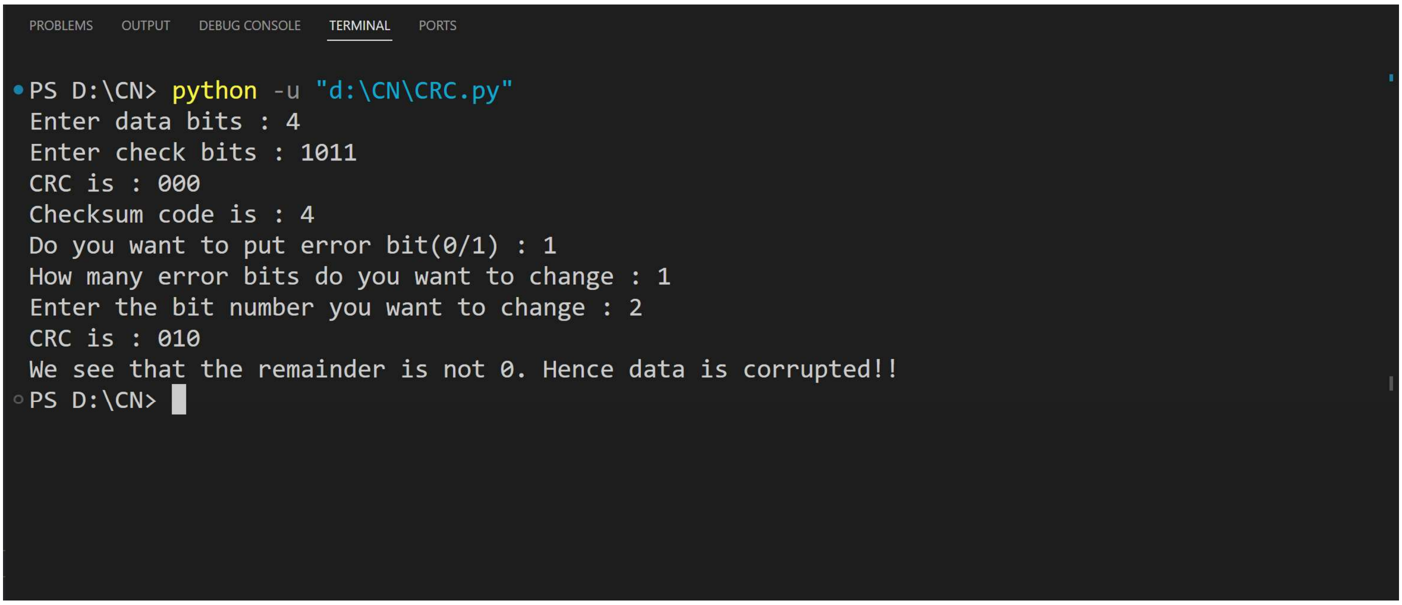
29/08/23

A

```
def XOR(x, y):

    if x == y:

        return 0

    return 1


def flip(x):

    if x == 0:

        return 1

    return 0


def moduloDivision(data, dividend, divisor):

    for i in range(len(data)):

        if dividend[i] == 1:

            for j in range(len(divisor)):

                dividend[i + j] = XOR(dividend[i + j], divisor[j])

    return dividend


def displayCRC(data, dividend):

    print("CRC is :", end=" ")

    for i in range(len(data), len(dividend)):

        print(dividend[i], end="")

    print()


def displayCheckSum(data, dividend):

    print("Checksum code is :", end=" ")

    for i in range(len(data)):

        dividend[i] = int(data[i])

        print(dividend[i], end="")

    print()


def main():

    print("Enter data bits : ", end="")
```

```
data = input()
print("Enter check bits : ", end="")
check = input()
dividend = [0] * (len(data) + len(check) - 1)
divisor = [0] * len(check)

for i in range(len(data)):
    dividend[i] = int(data[i])

for i in range(len(check)):
    divisor[i] = int(check[i])

# Calculating remainder (CRC)
dividend = moduloDivision(data, dividend, divisor)

# Display remainder
displayCRC(data, dividend)

# Display checksum
displayCheckSum(data, dividend)

# Asking for a change in checksum
print("Do you want to put error bit(0/1) : ", end="")
choice = int(input())

if choice == 1:
    print("How many error bits do you want to change : ", end="")
    select = int(input())
    print("Enter the bit number you want to change : ", end="")
    change = input()
    for i in range(select):
        dividend[int(change[i])] = flip(dividend[int(change[i])])
        dividend = moduloDivision(data, dividend, divisor)

        displayCRC(data, dividend)

        print("We see that the remainder is not 0. Hence data is corrupted!!")

    else:

        print("CRC obtained at the receiver side is zero")

        print("Data sent without corruption")


if __name__ == "__main__":

    main()
```

OUTPUT



```
• PS D:\CN> python -u "d:\CN\CRC.py"
  Enter data bits : 4
  Enter check bits : 1011
  CRC is : 000
  Checksum code is : 4
  Do you want to put error bit(0/1) : 1
  How many error bits do you want to change : 1
  Enter the bit number you want to change : 2
  CRC is : 010
  We see that the remainder is not 0. Hence data is corrupted!!
○ PS D:\CN> 
```