Sayan Chatterjee
2K19/CO/348

# Experiment- 4

## Aim:

Implement CYCLIC REDUNDANCY CHECK technique for error detection in data sent in form of bits.

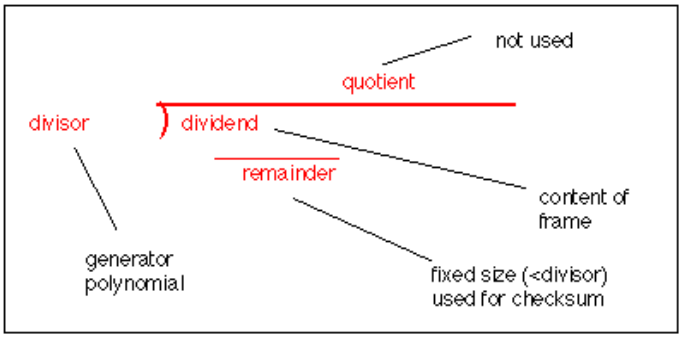
## Language Used:

Python

## Theory:

CRC stands for Cyclic Redundancy Check. It is an error-detecting code used to determine if a block of data has been corrupted. CRCs are ubiquitous. They are present in many of the link layers that TCP/IP is used over. For instance, Ethernet and Wi-Fi packets both contain CRCs.

In the cyclic redundancy check, a fixed number of check bits, often called a checksum, are appended to the message that needs to be transmitted. The data receivers receive the data, and inspect the check bits for any errors.




## Code:

```
def xor(a, b):
        result = []
        for i in range(1, len(a)):
                if a[i] == b[i]:
                        result.append('0')
                else:
                        result.append('1')
        return ''.join(result)

def mod2div(dvd, dvs):
        covered = len(dvs)
        picked = dvd[0 : covered]
        while covered < len(dvd):
                if picked[0] == '1': picked = xor(picked,dvs) + dvd[covered]
                else: picked = xor(picked,'0'*covered) + dvd[covered]
                covered += 1
        # For the last bit, done manually so that index does not go out of
bounds
        if picked[0] == '1': picked = xor(picked,dvs)
```

```
            else: picked = xor(picked,'0'*covered)
            #print("Remainder: ", picked)
            return picked

    def crc(data, gen):
            appended = data + '0'*(len(gen)-1)
            remainder = mod2div(appended, gen)
            encoded = data + remainder
            print("Encoded Data: ",encoded)

    data = input("Enter the Data")
    gen = input("Enter the Generator")
    crc(data, gen)

    receive = input("Enter the Received message : ")

    if int(mod2div(receive, gen)) == 0:
       print("Correct Received, Original Message is: ", receive[:-(len(gen)-1)])
    else:
       print("Wrong Data")
```

## Output:

```
Enter the Data 101010111
Enter the Generator 1101
Encoded Data:  101010111011
Enter the Received message :  101010111011
Correct Received, Original Message is:  101010111
```

```
Enter the Data 10011101
Enter the Generator 1001
Encoded Data:  10011101100
Enter the Received message :  10011101100
Correct Received, Original Message is:  10011101
```