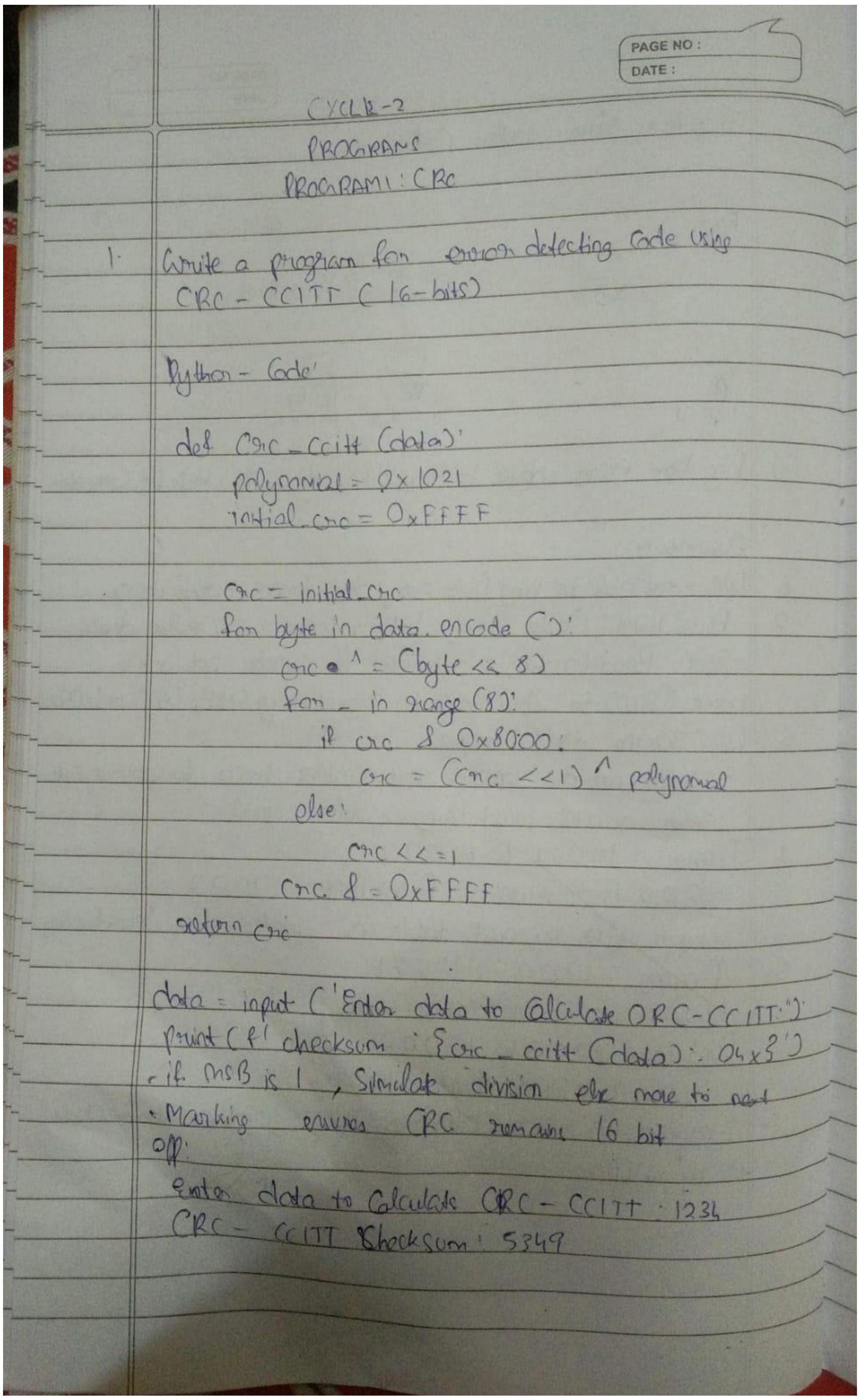
LABORATORY PROGRAM – 1

Write a program for error detecting code using CRC-CCITT (8-bits).



PAGE NO :
DATE :

CYCLE-2
PROGRAMS
PROGRAM1: CRC

1. Write a program for error detecting code using CRC - CCITT (16-bits)

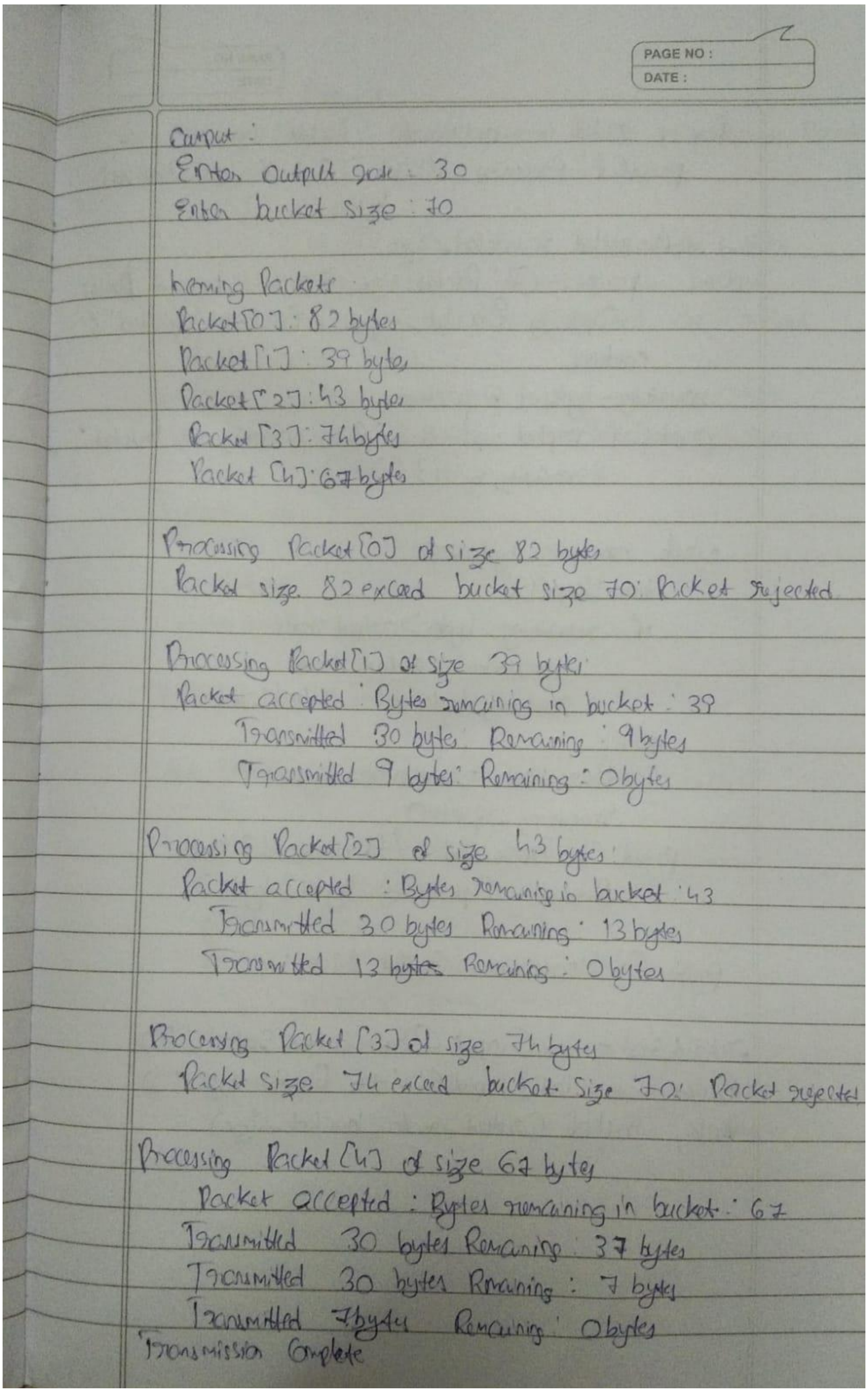
Python - Code'

```
def crc_ccitt (data)'
        polynomial = 0x1021
        initial_crc = 0xFFFF

        crc = initial_crc
        for byte in data.encode ():
            crc ^= (byte << 8)
            for _ in range (8):
                if crc & 0x8000:
                    crc = (crc <<1) ^ polynomial
                else:
                    crc <<=1
            crc &= 0xFFFF
    return crc

data = input ('Enter data to Calculate CRC-CCITT:')
print (f' checksum : {crc_ccitt (data): 04x}')
```

- if MSB is 1 , Simulate division else move to next
- Marking ensures CRC remains 16 bit

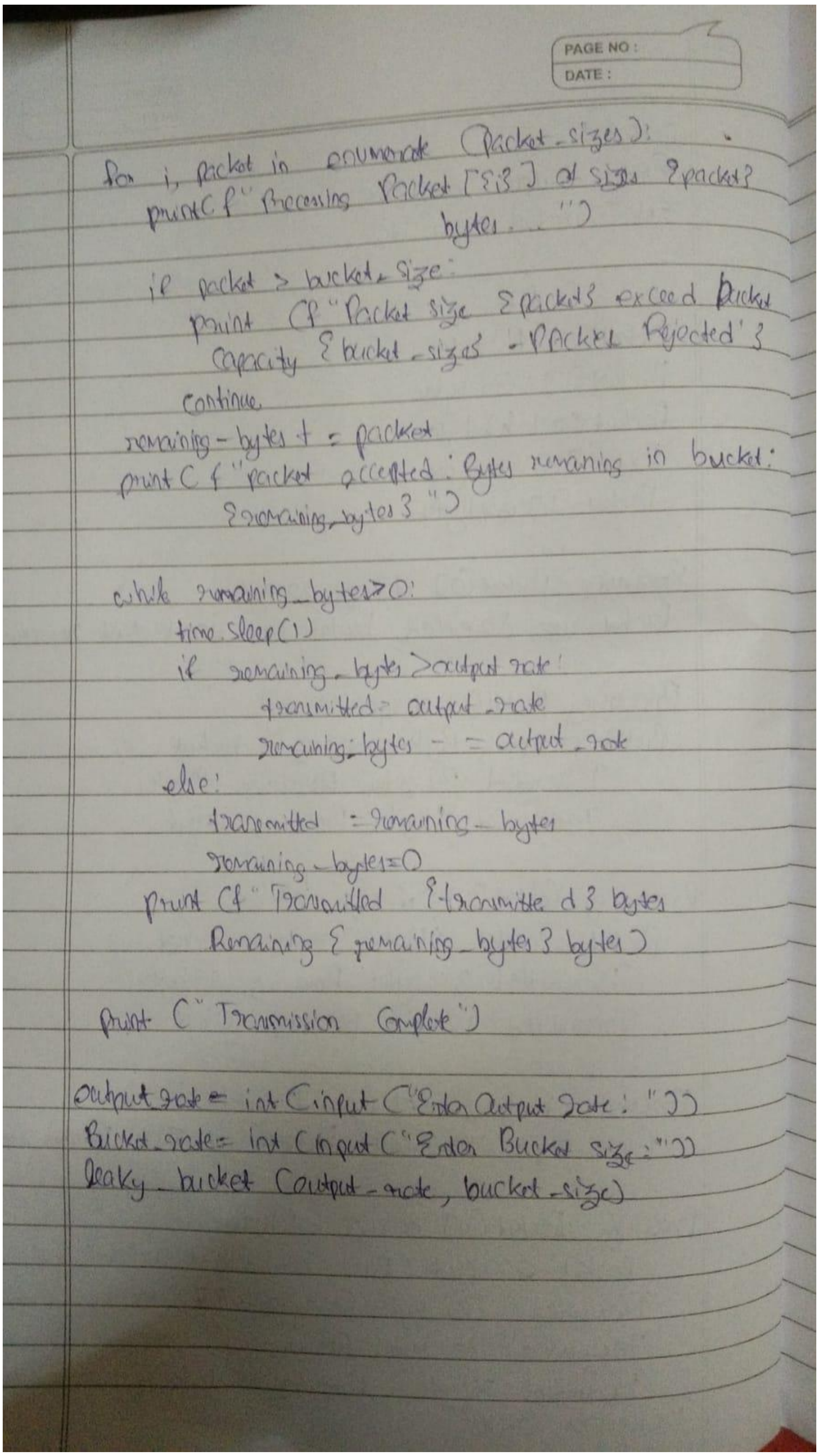
o/p:

```
Enter data to Calculate CRC - CCITT : 1234
CRC - CCITT Checksum: 5349
```


PAGE NO :
DATE :

Output :

```
Enter output rate : 30
Enter bucket size : 70

Incoming Packets
Packet[0] : 82 bytes
Packet[1] : 39 bytes
Packet[2] : 43 bytes
Packet[3] : 74 bytes
Packet[4] : 67 bytes

Processing Packet[0] of size 82 bytes
Packet size 82 exceed bucket size 70 : Packet rejected

Processing Packet[1] of size 39 bytes
Packet accepted : Bytes remaining in bucket : 39
        Transmitted 30 bytes Remaining : 9 bytes
        Transmitted 9 bytes : Remaining : 0 bytes

Processing Packet[2] of size 43 bytes
Packet accepted : Bytes remaining in bucket : 43
    Transmitted 30 bytes Remaining : 13 bytes
    Transmitted 13 bytes Remaining : 0 bytes

Processing Packet[3] of size 74 bytes
Packet size 74 exceed bucket size 70 : Packet rejected

Processing Packet[4] of size 67 bytes
    Packet accepted : Bytes remaining in bucket : 67
    Transmitted 30 bytes Remaining : 37 bytes
    Transmitted 30 bytes Remaining : 7 bytes
    Transmitted 7 bytes Remaining : 0 bytes
Transmission Complete
```



PAGE NO :
DATE :

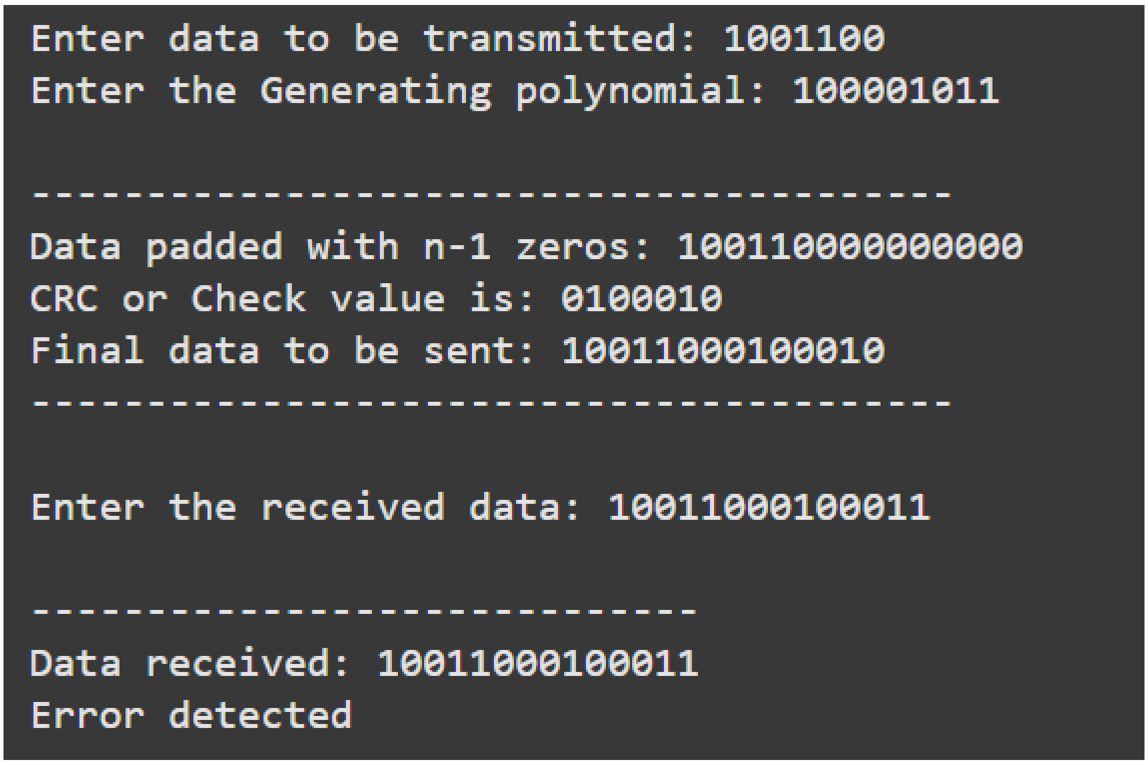
```
for i, packet in enumerate (Packet_sizes):
    print(f" Processing Packet [{i}] of size {packet}
                        bytes ... ")

    if packet > bucket_size:
        print (f" Packet size {packet} exceed bucket
          capacity {bucket_size} - Packet Rejected'}
        continue
    remaining - bytes + = packet
    print (f "packet accepted: Bytes remaining in bucket:
            {remaining_bytes} ")

    while remaining_bytes>0:
        time.sleep(1)
        if remaining_bytes >output rate:
            transmitted = output_rate
            remaining_bytes - = output_rate
      else:
          transmitted = remaining_bytes
          remaining_bytes=0
      print (f" Transmitted {transmitted} bytes
          Remaining {remaining_bytes} bytes)

print (" Transmission Complete")

output_rate = int (input ("Enter Output rate: "))
Bucket_rate = int (input ("Enter Bucket size: "))
leaky_bucket (output_rate, bucket_size)
```

Code

```
def xor(dividend, divisor):
    """Perform XOR operation between dividend and divisor."""
    result = ''
    for i in range(1, len(divisor)):
        result += '0' if dividend[i] == divisor[i] else '1'
    return result

def crc(data, gen_poly):
    """Compute the CRC check value using CRC-CCITT (8-bit)."""
    data_length = len(data)
    gen_length = len(gen_poly)

    # Append n-1 zeros to the data
    padded_data = data + '0' * (gen_length - 1)
    check_value = padded_data[:gen_length]

    for i in range(data_length):
        if check_value[0] == '1':
            # XOR operation if the first bit is 1
            check_value = xor(check_value, gen_poly)
        else:
            # Retain original check value if first bit is 0
            check_value = check_value[1:]

        # Shift left and add the next data bit
        if i + gen_length < len(padded_data):
            check_value += padded_data[i + gen_length]

    return check_value[1:]  # Remove the leading bit

def receiver(data, gen_poly):
    """Simulate the receiver side to check for errors."""
    print("\n----------------------------")
    print("Data received:", data)

    # Perform CRC computation on received data
    remainder = crc(data, gen_poly)

    # Check if the remainder is all zeros
    if '1' in remainder:
        print("Error detected")
    else:
        print("No error detected")

if __name__ == "__main__":
    # Input data and generator polynomial
    data = input("Enter data to be transmitted: ")
    gen_poly = input("Enter the Generating polynomial: ")
```

```
# Compute CRC check value
check_value = crc(data, gen_poly)
print("\n----------------------------------------")
print("Data padded with n-1 zeros:", data + '0' * (len(gen_poly) - 1))
print("CRC or Check value is:", check_value)

# Append check value to data for transmission
transmitted_data = data + check_value
print("Final data to be sent:", transmitted_data)
print("----------------------------------------\n")

# Simulate the receiver side
received_data = input("Enter the received data: ")
receiver(received_data, gen_poly)
```

Output



```
Enter data to be transmitted: 1001100
Enter the Generating polynomial: 100001011

----------------------------------------
Data padded with n-1 zeros: 1001100000000000
CRC or Check value is: 0100010
Final data to be sent: 10011000100010
----------------------------------------

Enter the received data: 10011000100011

------------------------------
Data received: 10011000100011
Error detected
```