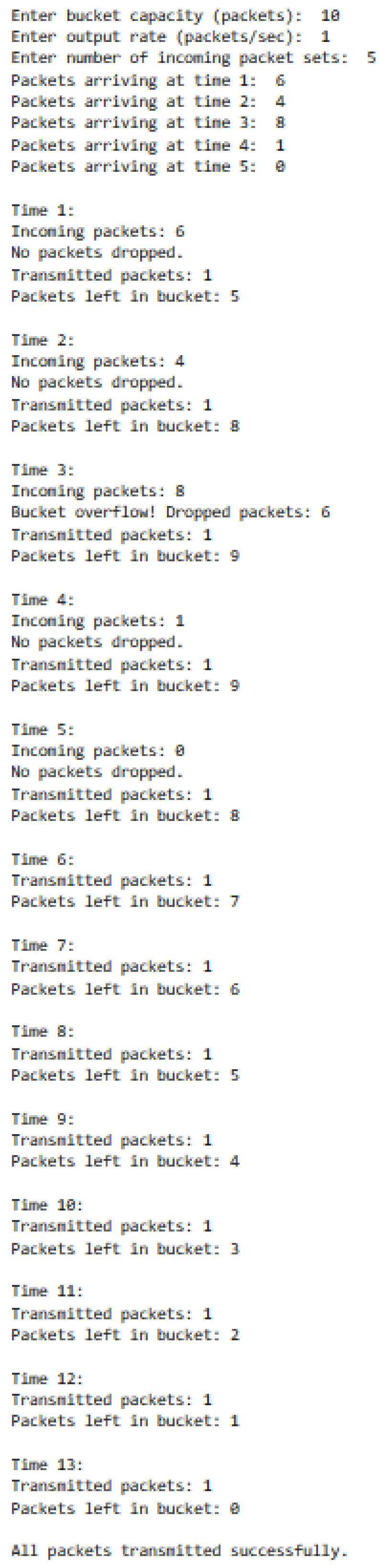
Leaky Bucket

```
def leaky_bucket(bucket_capacity, output_rate, incoming_packets):
    stored = 0  # current number of packets in the bucket

    for time, packets in enumerate(incoming_packets, start=1):
        print(f"\nTime {time}:")
        print(f"Incoming packets: {packets}")

        # Check for overflow
        if packets + stored > bucket_capacity:
            dropped = (packets + stored) - bucket_capacity
            stored = bucket_capacity
            print(f"Bucket overflow! Dropped packets: {dropped}")
        else:
            stored += packets
            print("No packets dropped.")

        # Transmit packets at output rate
        transmitted = min(stored, output_rate)
        stored -= transmitted
        print(f"Transmitted packets: {transmitted}")
        print(f"Packets left in bucket: {stored}")

    # Empty remaining packets in the bucket
    time = len(incoming_packets)
    while stored > 0:
        time += 1
        transmitted = min(stored, output_rate)
        stored -= transmitted
        print(f"\nTime {time}:")
        print(f"Transmitted packets: {transmitted}")
        print(f"Packets left in bucket: {stored}")

    print("\nAll packets transmitted successfully.")


# ---- Main Program ----
if __name__ == "__main__":
    bucket_capacity = int(input("Enter bucket capacity (packets): "))
    output_rate = int(input("Enter output rate (packets/sec): "))

    n = int(input("Enter number of incoming packet sets: "))
    incoming_packets = []
    for i in range(n):
        packets = int(input(f"Packets arriving at time {i + 1}: "))
        incoming_packets.append(packets)

    leaky_bucket(bucket_capacity, output_rate, incoming_packets)
```

Output:

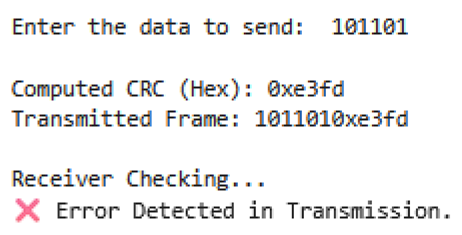


```
Enter bucket capacity (packets):  10
Enter output rate (packets/sec):  1
Enter number of incoming packet sets:  5
Packets arriving at time 1:  6
Packets arriving at time 2:  4
Packets arriving at time 3:  8
Packets arriving at time 4:  1
Packets arriving at time 5:  0

Time 1:
Incoming packets: 6
No packets dropped.
Transmitted packets: 1
Packets left in bucket: 5

Time 2:
Incoming packets: 4
No packets dropped.
Transmitted packets: 1
Packets left in bucket: 8

Time 3:
Incoming packets: 8
Bucket overflow! Dropped packets: 6
Transmitted packets: 1
Packets left in bucket: 9

Time 4:
Incoming packets: 1
No packets dropped.
Transmitted packets: 1
Packets left in bucket: 9

Time 5:
Incoming packets: 0
No packets dropped.
Transmitted packets: 1
Packets left in bucket: 8

Time 6:
Transmitted packets: 1
Packets left in bucket: 7

Time 7:
Transmitted packets: 1
Packets left in bucket: 6

Time 8:
Transmitted packets: 1
Packets left in bucket: 5

Time 9:
Transmitted packets: 1
Packets left in bucket: 4

Time 10:
Transmitted packets: 1
Packets left in bucket: 3

Time 11:
Transmitted packets: 1
Packets left in bucket: 2

Time 12:
Transmitted packets: 1
Packets left in bucket: 1

Time 13:
Transmitted packets: 1
Packets left in bucket: 0

All packets transmitted successfully.
```

## CRC-CCITT

```
def crc_ccitt(data, poly=0x1021, init_crc=0xFFFF):
    """
    Calculate 16-bit CRC using CRC-CCITT algorithm.

    data: input string of bits (e.g. '1101011011')
    poly: generator polynomial (0x1021 by default)
    init_crc: initial value (0xFFFF)
    """
    crc = init_crc
    for byte in data.encode('utf-8'):  # process byte-by-byte
        crc ^= (byte << 8)
        for _ in range(8):
            if (crc & 0x8000):
                crc = (crc << 1) ^ poly
            else:
                crc <<= 1
            crc &= 0xFFFF  # keep crc 16-bit
    return crc

# --- Main Program ---

data = input("Enter the data to send: ")

# Compute CRC
crc_value = crc_ccitt(data)
print(f"\nComputed CRC (Hex): {hex(crc_value)}")

# Append CRC to data (simulate transmission)
transmitted_data = data + hex(crc_value)
print(f"Transmitted Frame: {transmitted_data}")

# Receiver side: Recompute CRC to check for errors
received_data = data[:-1]+'x'
received_crc = crc_ccitt(received_data)

print("\nReceiver Checking...")
if received_crc == crc_value:
    print("✅ No Error Detected.")
else:
    print("❌ Error Detected in Transmission.")
```

Output:



```
Enter the data to send:  101101

Computed CRC (Hex): 0xe3fd
Transmitted Frame: 1011010xe3fd

Receiver Checking...
❌ Error Detected in Transmission.
```