## Remaining Lab topics:

LRC = 2D parity (Program No:11)

Checksum = Addition Method (Program No:11)

CRC = Division Method (Program No:12)

Hamming Code : (Program No:13)

Leaky Bucket : (Program No:14)

Token Bucket : (Program No:15)

## Program 11 LRC :

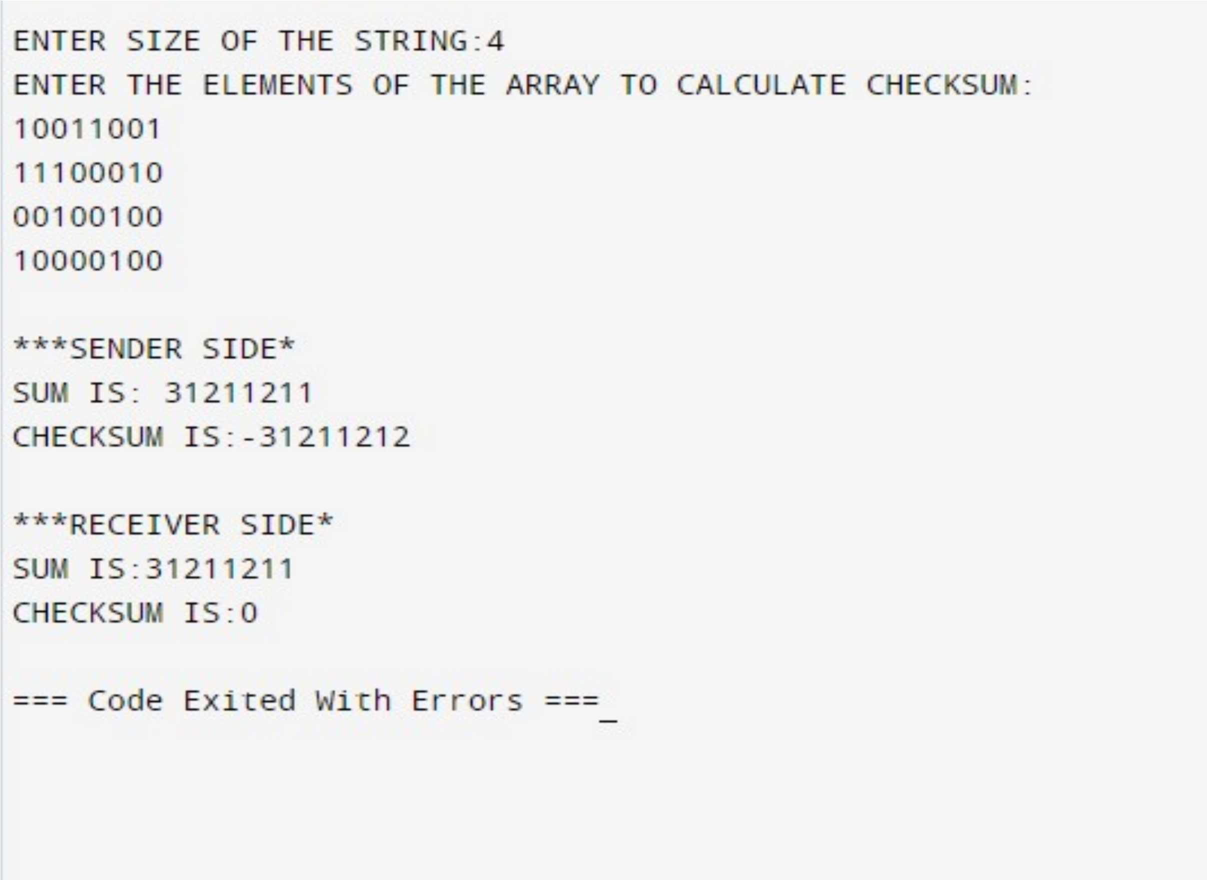
```
#include <stdio.h>

// Function to calculate LRC
unsigned char calculateLRC(unsigned char *data, int length) {
   unsigned char lrc = 0;
   for (int i = 0; i < length; i++) {
      lrc += data[i];
   }
   // Take the one's complement of the sum
   lrc = (~lrc) + 1;
   return lrc;
}

// Function to print a byte in binary format
void printBinary(unsigned char byte) {
   for (int i = 7; i >= 0; i--) {
      printf("%d", (byte >> i) & 1);
   }
}

int main() {
   // Example data to be sent (replace this with your actual data)
   unsigned char dataToSend[] = {0x41, 0x42, 0x43, 0x44}; // "ABCD" in ASCII
```

```
    int dataLength = sizeof(dataToSend) / sizeof(dataToSend[0]);

    // Calculate LRC for the data
    unsigned char lrc = calculateLRC(dataToSend, dataLength);

    // Append LRC to the data
    dataToSend[dataLength] = lrc;

    // Display the data with appended LRC in binary format
    printf("Data with appended LRC (in binary):\n");
    for (int i = 0; i < dataLength + 1; i++) {
        printBinary(dataToSend[i]);
        printf(" ");
    }
    printf("\n");

    return 0;
}
```

## Program 11 Checksum:

```
#include<stdio.h>
#include<math.h>

 int sender(int arr[10],int n)
{
int checksum,sum=0,i;
printf("\n***SENDER SIDE*\n");
for(i=0;i<n;i++)
sum+=arr[i];
printf("SUM IS: %d",sum);
checksum=~sum; //1's complement of sum
printf("\nCHECKSUM IS:%d",checksum);
return checksum;
}
void receiver(int arr[10],int n,int sch)
{
int checksum,sum=0,i;
 printf("\n\n***RECEIVER SIDE*\n");
 for(i=0;i<n;i++)
sum+=arr[i];
printf("SUM IS:%d",sum);
sum=sum+sch;
checksum=~sum; //1's complement of sum
printf("\nCHECKSUM IS:%d",checksum);
}

void main()
{
 int n,sch,rch;
 printf("\nENTER SIZE OF THE STRING:");
scanf("%d",&n);
 int arr[n];
 printf("ENTER THE ELEMENTS OF THE ARRAY TO CALCULATE CHECKSUM:\n");
for(int i=0;i<n;i++)
{
 scanf("%d",&arr[i]);
}
 sch=sender(arr,n);
 receiver(arr,n,sch);
 }
```

**Output:**



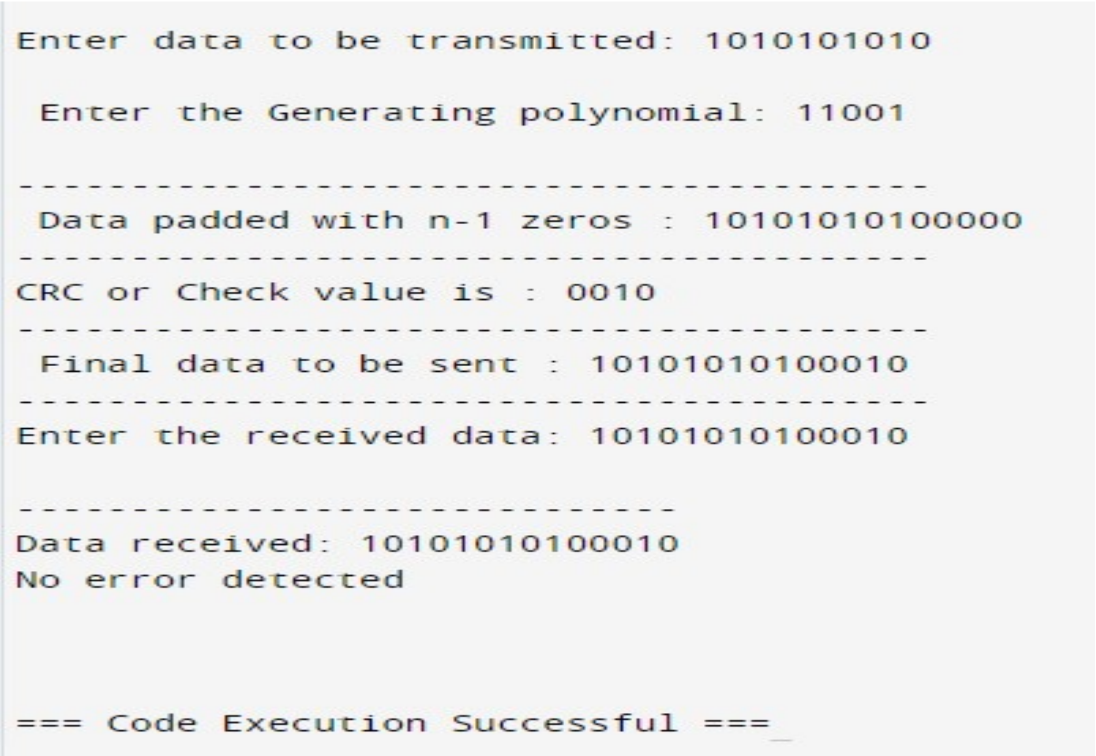
```
ENTER SIZE OF THE STRING:4
ENTER THE ELEMENTS OF THE ARRAY TO CALCULATE CHECKSUM:
10011001
11100010
00100100
10000100

***SENDER SIDE*
SUM IS: 31211211
CHECKSUM IS:-31211212

***RECEIVER SIDE*
SUM IS:31211211
CHECKSUM IS:0

=== Code Exited With Errors ===_
```

## Program 12 CRC:

```
#include<stdio.h>
#include<string.h>
// length of the generator polynomial
#define N strlen(gen_poly)
// data to be transmitted and received
char data[28];
// CRC value
char check_value[28];
// generator polynomial
char gen_poly[10];
// variables
int data_length,i,j;
// function that performs XOR operation
void XOR(){
   // if both bits are the same, the output is 0
   // if the bits are different the output is 1
   for(j = 1;j < N; j++)
   check_value[j] = (( check_value[j] == gen_poly[j])?'0':'1');

}
// Function to check for errors on the receiver side
void receiver(){
// get the received data
   printf("Enter the received data: ");
   scanf("%s", data);
   printf("\n-----------------------------\n");
   printf("Data received: %s", data);
// Cyclic Redundancy Check
   crc();
// Check if the remainder is zero to find the error
   for(i=0;(i<N-1) && (check_value[i]!='1');i++);
      if(i<N-1)
         printf("\nError detected\n\n");
      else
         printf("\nNo error detected\n\n");
}

void crc(){
   // initializing check_value
   for(i=0;i<N;i++)
      check_value[i]=data[i];
   do{
   // check if the first bit is 1 and calls XOR function
      if(check_value[0]=='1')
         XOR();
// Move the bits by 1 position for the next computation
      for(j=0;j<N-1;j++)
         check_value[j]=check_value[j+1];
      // appending a bit from data
      check_value[j]=data[i++];
   }while(i<=data_length+N-1);
// loop until the data ends
}

int main()
{
   // get the data to be transmitted
   printf("\nEnter data to be transmitted: ");
   scanf("%s",data);
```

```
    printf("\n Enter the Generating polynomial: ");
    // get the generator polynomial
    scanf("%s",gen_poly);
    // find the length of data
    data_length=strlen(data);
    // appending n-1 zeros to the data
    for(i=data_length;i<data_length+N-1;i++)
        data[i]='0';
    printf("\n----------------------------------------");
// print the data with padded zeros
    printf("\n Data padded with n-1 zeros : %s",data);
    printf("\n----------------------------------------");
// Cyclic Redundancy Check
    crc();
// print the computed check value
    printf("\nCRC or Check value is : %s",check_value);
// Append data with check_value(CRC)
    for(i=data_length;i<data_length+N-1;i++)
        data[i]=check_value[i-data_length];
    printf("\n----------------------------------------");
// printing the final data to be sent
    printf("\n Final data to be sent : %s",data);
    printf("\n----------------------------------------\n");
// Calling the receiver function to check errors
    receiver();
        return 0;
}
```

**Output:**



```
Enter data to be transmitted: 1010101010

 Enter the Generating polynomial: 11001

----------------------------------------
 Data padded with n-1 zeros : 10101010100000
----------------------------------------
CRC or Check value is : 0010
----------------------------------------
 Final data to be sent : 10101010100010
----------------------------------------
Enter the received data: 10101010100010

-----------------------------
Data received: 10101010100010
No error detected



=== Code Execution Successful ===
```

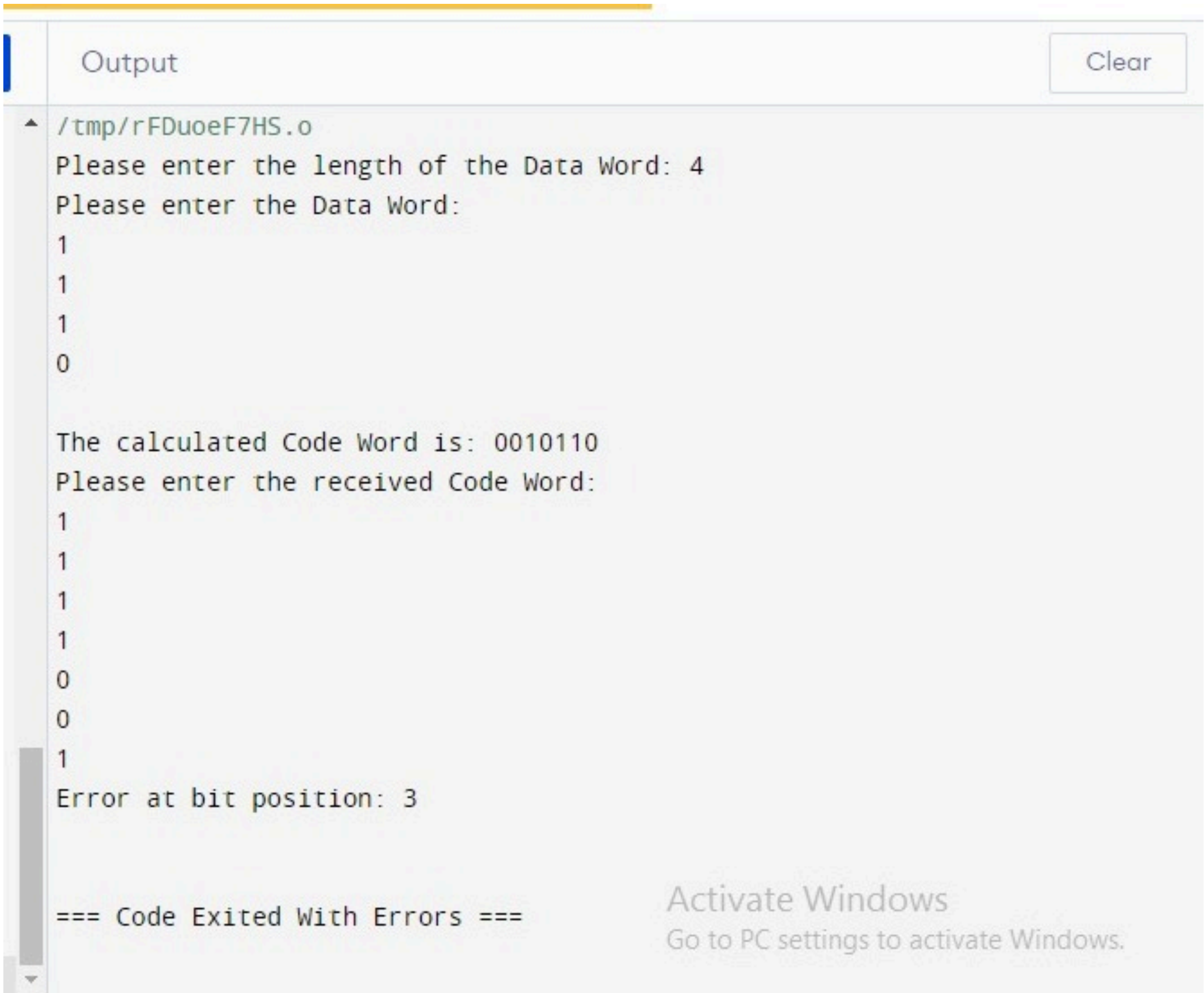
## Practical No. 13 Hamming Code :

```
#include <stdio.h>
#include <math.h>
int input[32];
int code[32];
int ham_calc(int,int);
void main()
{
        int n,i,p_n = 0,c_l,j,k;
        printf("Please enter the length of the Data Word: ");
        scanf("%d",&n);
        printf("Please enter the Data Word:\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&input[i]);
        }

        i=0;
```

```
while(n>(int)pow(2,i)-(i+1))
{
    p_n++;
    i++;
}

c_l = p_n + n;

j=k=0;
for(i=0;i<c_l;i++)
{

    if(i==((int)pow(2,k)-1))
    {
        code[i]=0;
        k++;
    }
    else
    {
        code[i]=input[j];
        j++;
    }
}
for(i=0;i<p_n;i++)
{
    int position = (int)pow(2,i);
    int value = ham_calc(position,c_l);
    code[position-1]=value;
}
printf("\nThe calculated Code Word is: ");
for(i=0;i<c_l;i++)
    printf("%d",code[i]);
printf("\n");
printf("Please enter the received Code Word:\n");
for(i=0;i<c_l;i++)
    scanf("%d",&code[i]);

int error_pos = 0;
for(i=0;i<p_n;i++)
{
    int position = (int)pow(2,i);
    int value = ham_calc(position,c_l);
    if(value != 0)
```

```
            error_pos+=position;
    }
    if(error_pos == 1)
        printf("The received Code Word is correct.\n");
    else
        printf("Error at bit position: %d\n",error_pos);
}
int ham_calc(int position,int c_l)
{
    int count=0,i,j;
    i=position-1;
    while(i<c_l)
    {
        for(j=i;j<i+position;j++)
        {
            if(code[j] == 1)
                count++;
        }
        i=i+2*position;
    }
    if(count%2 == 0)
        return 0;
    else
        return 1;
}
```

**OUTPUT :**



```
/tmp/rFDuoeF7HS.o
Please enter the length of the Data Word: 4
Please enter the Data Word:
1
1
1
0

The calculated Code Word is: 0010110
Please enter the received Code Word:
1
1
1
1
0
0
1
Error at bit position: 3


=== Code Exited With Errors ===
```

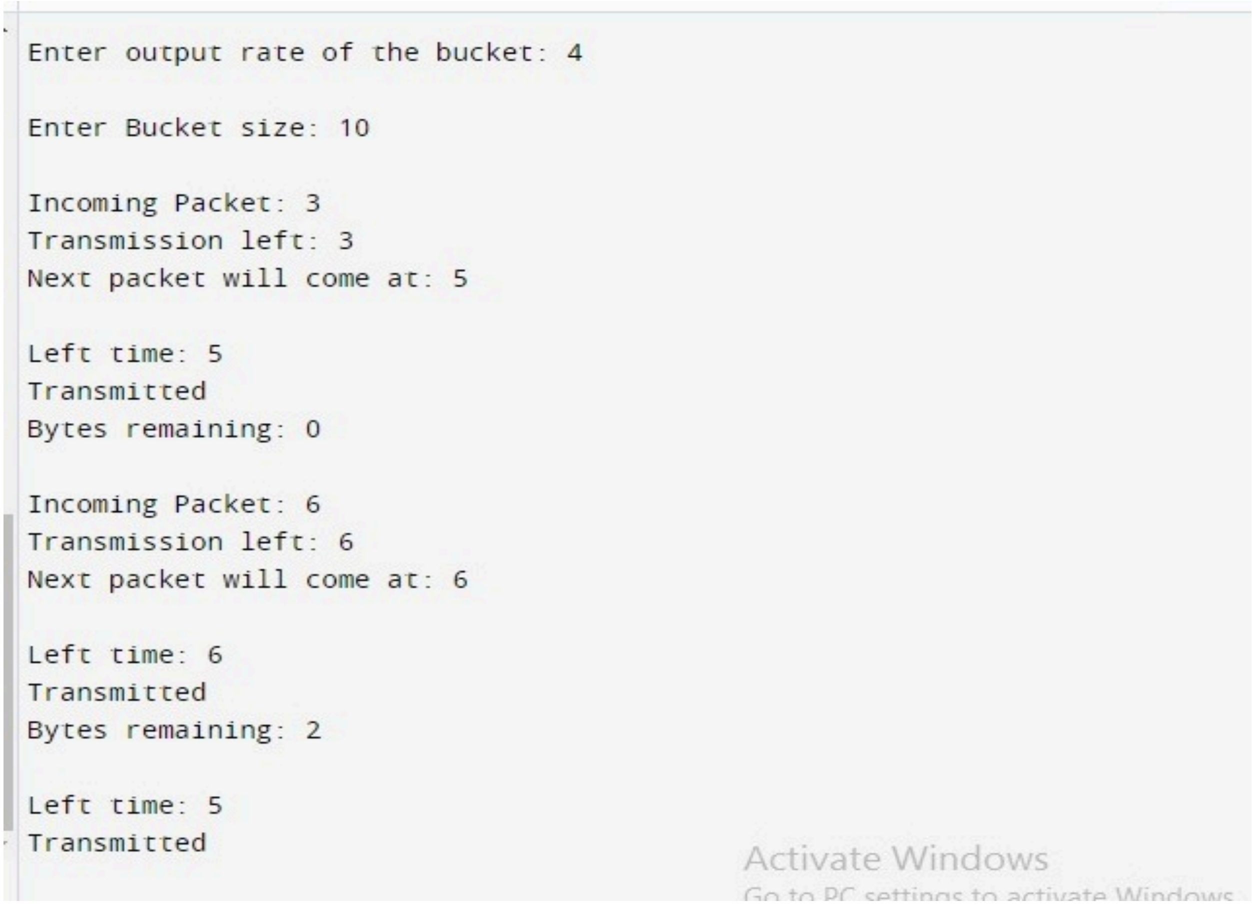
**Program 14  Leaky Bucket :**

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> // For sleep function

int main() {
   int i, packets[10], content = 0, newcontent, time, clk, bucket_size, output_rate;

   // Generate random packet sizes
```

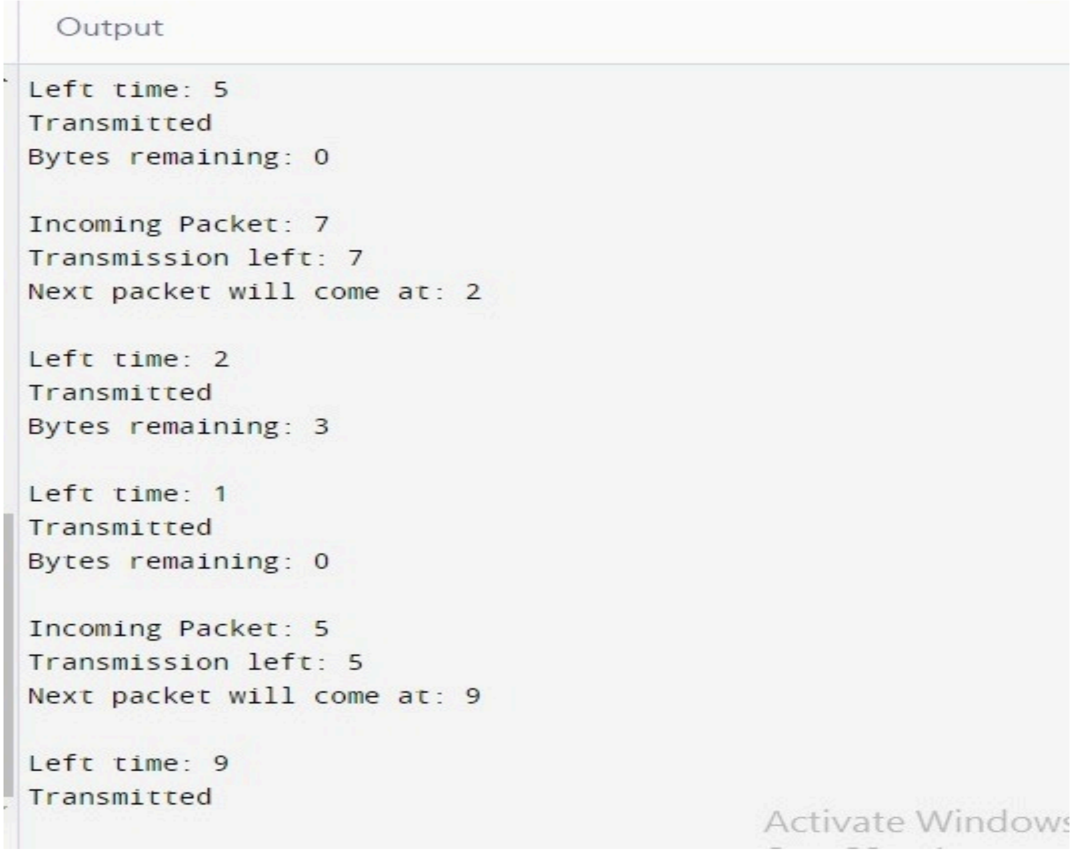
```
    for (i = 0; i < 5; i++) {
        packets[i] = rand() % 10;
        if (packets[i] == 0)
            i--; // Regenerate if packet size is 0
    }

    printf("\nEnter output rate of the bucket: ");
    scanf("%d", &output_rate);

    printf("\nEnter Bucket size: ");
    scanf("%d", &bucket_size);

    for (i = 0; i < 5; ++i) {
        if ((packets[i] + content) > bucket_size) {
            if (packets[i] > bucket_size)
                printf("\nIncoming packet size %d greater than the size of the bucket\n",
packets[i]);
            else
                printf("\nBucket size exceeded\n");
        } else {
            newcontent = packets[i];
            content += newcontent;
            printf("\nIncoming Packet: %d\n", newcontent);
            printf("Transmission left: %d\n", content);
            time = rand() % 10;
            printf("Next packet will come at: %d\n", time);

            for (clk = 0; clk < time && content > 0; ++clk) {
                printf("\nLeft time: %d", (time - clk));
                sleep(1);

                if (content > 0) {
                    printf("\nTransmitted\n");
                    if (content < output_rate)
                        content = 0;
                    else
```

```
                        content -= output_rate;
                    printf("Bytes remaining: %d\n", content);
                } else {
                    printf("\nNo packets to send\n");
                }
            }
        }
    }

    return 0;
}
```

**Output :**

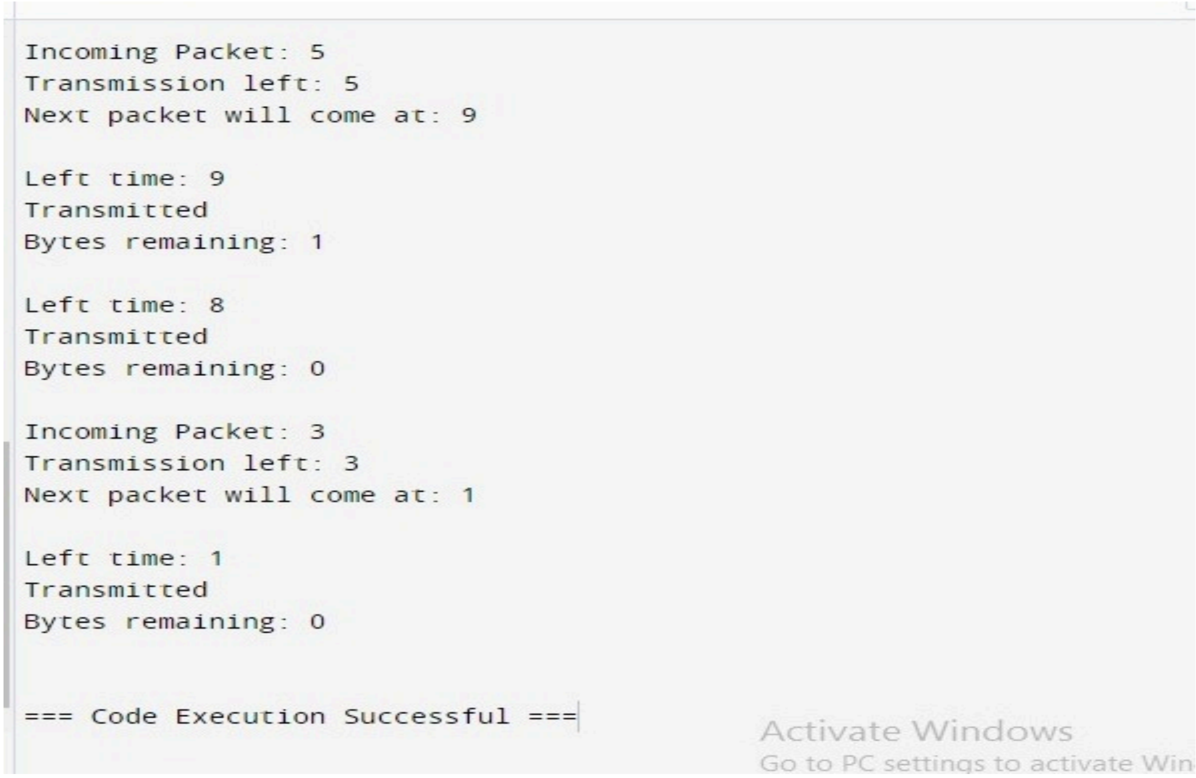


```
Enter output rate of the bucket: 4

Enter Bucket size: 10

Incoming Packet: 3
Transmission left: 3
Next packet will come at: 5

Left time: 5
Transmitted
Bytes remaining: 0

Incoming Packet: 6
Transmission left: 6
Next packet will come at: 6

Left time: 6
Transmitted
Bytes remaining: 2

Left time: 5
Transmitted
```



Output

```
Left time: 5
Transmitted
Bytes remaining: 0

Incoming Packet: 7
Transmission left: 7
Next packet will come at: 2

Left time: 2
Transmitted
Bytes remaining: 3

Left time: 1
Transmitted
Bytes remaining: 0

Incoming Packet: 5
Transmission left: 5
Next packet will come at: 9

Left time: 9
Transmitted
```

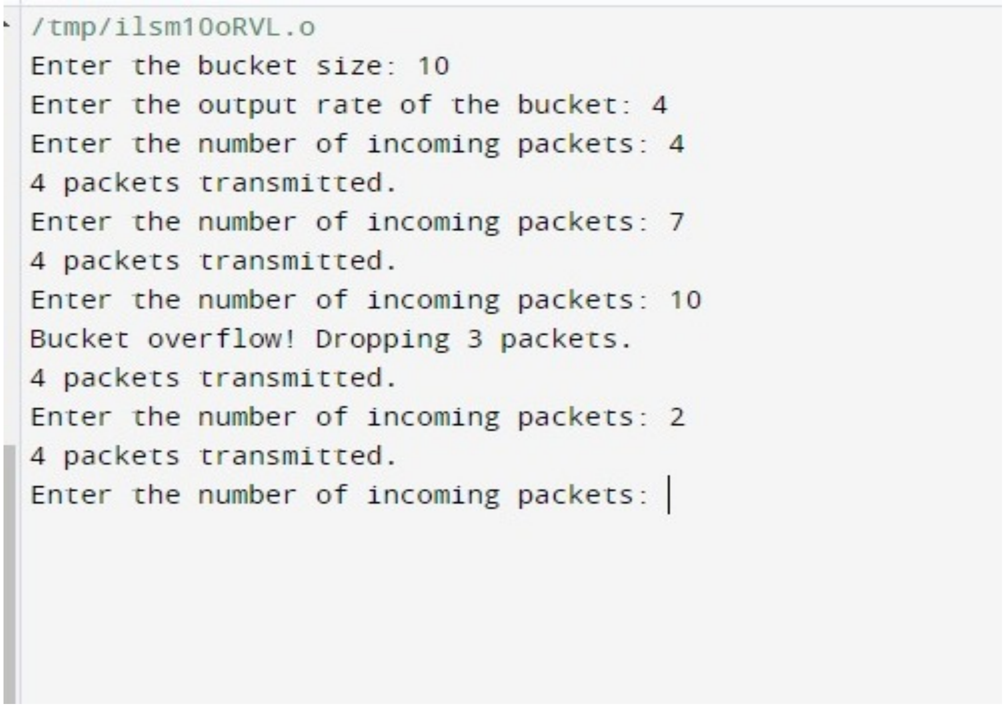


```
Incoming Packet: 5
Transmission left: 5
Next packet will come at: 9

Left time: 9
Transmitted
Bytes remaining: 1

Left time: 8
Transmitted
Bytes remaining: 0

Incoming Packet: 3
Transmission left: 3
Next packet will come at: 1

Left time: 1
Transmitted
Bytes remaining: 0


=== Code Execution Successful ===
```

**Program 15  Token Bucket :**

```
#include <stdio.h>
#include <stdbool.h>
#include <unistd.h> // for usleep function

int main() {
    int bucket_size, output_rate;

    // User input for bucket size and output rate
    printf("Enter the bucket size: ");
    scanf("%d", &bucket_size);
    printf("Enter the output rate of the bucket: ");
    scanf("%d", &output_rate);

    int bucket = 0;  // Current size of the bucket

    while (true) {
        // Generate some data, e.g., incoming packets
        int incoming_packets;
        printf("Enter the number of incoming packets: ");
        scanf("%d", &incoming_packets);

        // Add incoming packets to the bucket
        if (bucket + incoming_packets <= bucket_size) {
            bucket += incoming_packets;
        } else {
            printf("Bucket overflow! Dropping %d packets.\n", incoming_packets +
bucket - bucket_size);
            bucket = bucket_size;  // Bucket is full
        }

        // Transmit data from the bucket
        if (bucket >= output_rate) {
            printf("%d packets transmitted.\n", output_rate);
            bucket -= output_rate;
        } else {
            printf("Bucket empty.\n");
        }

        // Wait for a second before the next iteration
        usleep(1000000);  // Sleep for 1 second (1 million microseconds)
    }

    return 0;
}
```

**Output :**



```
/tmp/ilsm1OoRVL.o
Enter the bucket size: 10
Enter the output rate of the bucket: 4
Enter the number of incoming packets: 4
4 packets transmitted.
Enter the number of incoming packets: 7
4 packets transmitted.
Enter the number of incoming packets: 10
Bucket overflow! Dropping 3 packets.
4 packets transmitted.
Enter the number of incoming packets: 2
4 packets transmitted.
Enter the number of incoming packets: 
```