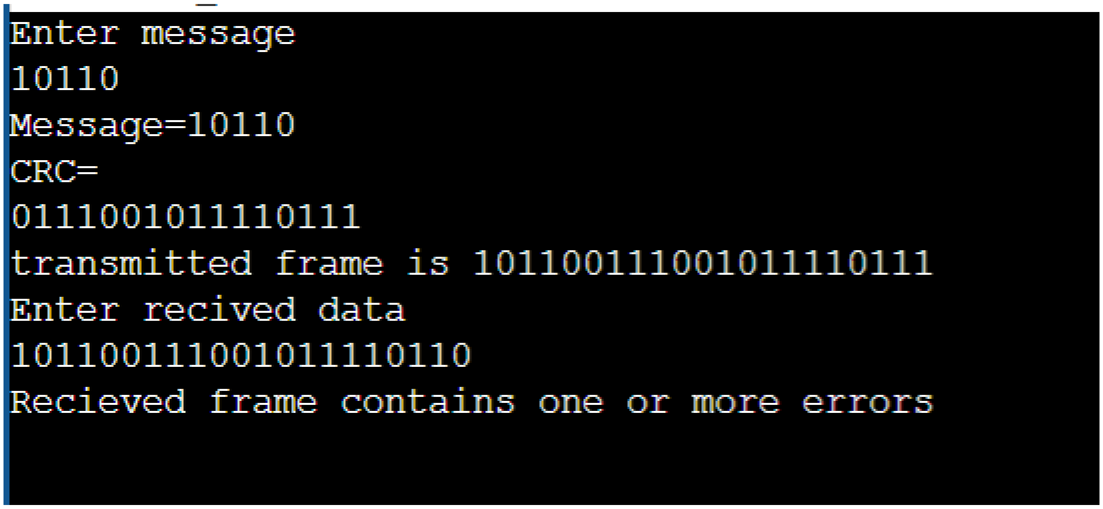
Prog 1 :
CRC-16

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

```
import java.util.*;

public class crc

{
        public static int n;

        public static void main(String[] args)

{

                Scanner in=new Scanner(System.in);

                crc ob=new crc();

                String code, copy, rec,zero="0000000000000000";

                System.out.println("Enter message");

                code=in.nextLine();

                n=code.length();

                copy=code;

                code+=zero;

                code=ob.divide(code);

                System.out.println("Message="+copy);

                copy=copy.substring(0,n)+code.substring(n);

                System.out.println("CRC=");

                System.out.println(code.substring(n));

                System.out.println("transmitted frame is "+copy);
        System.out.println("Enter recived data");

        rec=in.nextLine();

        if(zero.equals(ob.divide(rec).substring(n)))

            System.out.println("Correct bits recieved");

        else

            System.out.println("Recieved frame contains one or more errors");

        in.close();

    }

    public String divide(String s)

{

            int i,j;

            char x;

            String div="10001000000100001";

            for(i=0;i<n;i++)

            {

                x=s.charAt(i);

                for(j=0;j<17;j++)

                {

                    if(x=='1')

                    {if(s.charAt(i+j)!=div.charAt(j))

                        s=s.substring(0,i+j)+"1"+s.substring(i+j+1);

                    else
```

```
                        s=s.substring(0,i+j)+"0"+s.substring(i+j+1);

                    }


                }

            }

            return s;
    }

}
```

OUTPUT :



```
Enter message
10110
Message=10110
CRC=
01110010111110111
transmitted frame is 1011001110010111110111
Enter recived data
1011001110010111110110
Recieved frame contains one or more errors
```

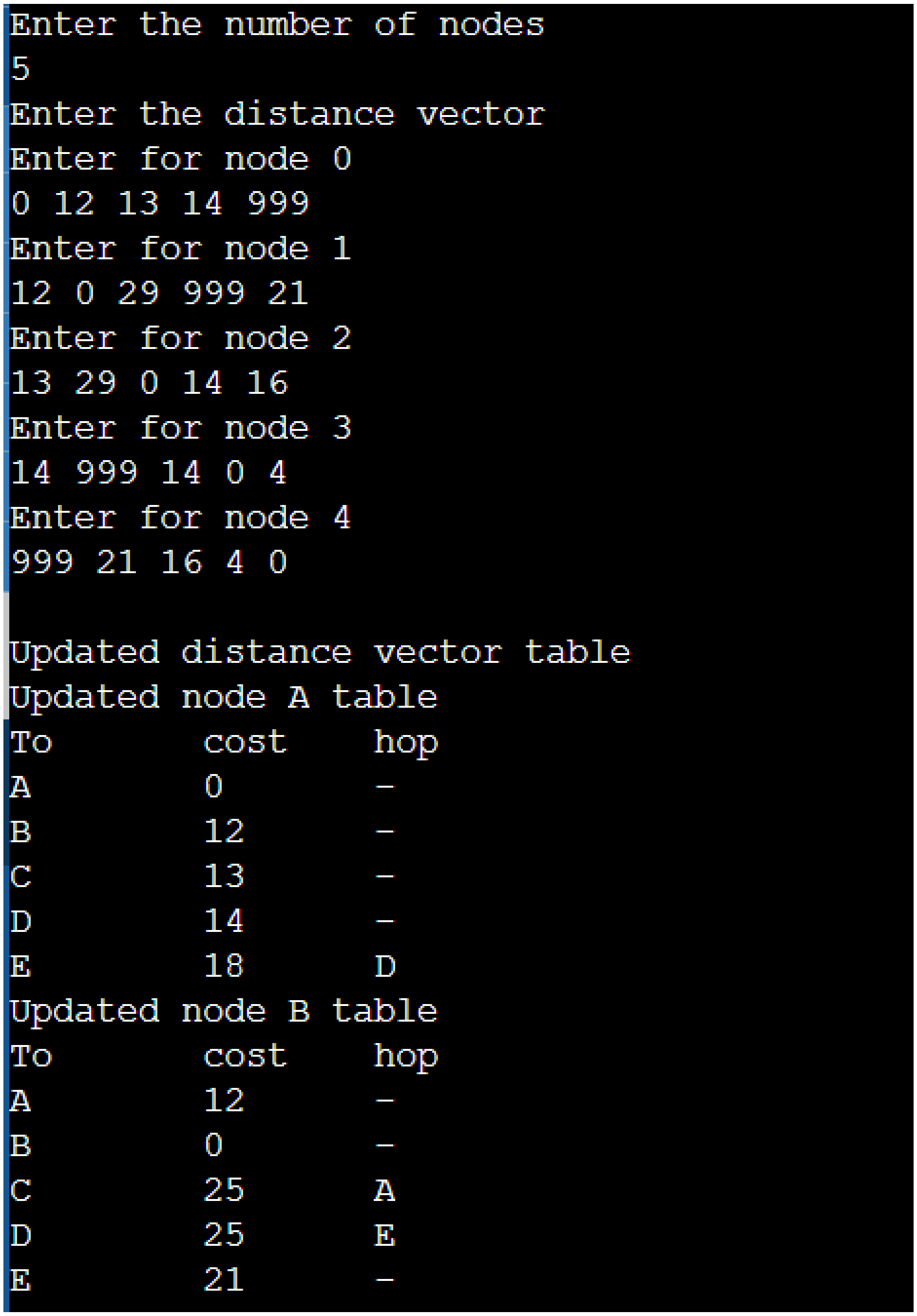
Prog 2 :
Write a program for distance vector algorithm to find suitable path for transmission.

Prog :

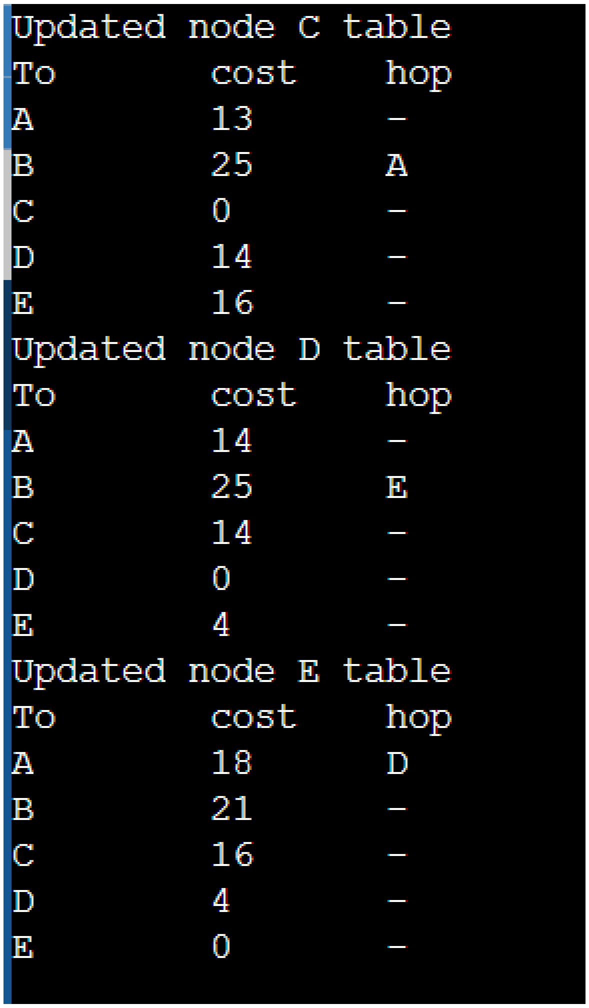
```
#include<stdio.h>
#define inf 999
struct routing{
    int dist[10];
    int hop[10];
};
struct routing nodes[10];

void init(int n){
    int i, j;
    for(i=0; i<n; i++){
        for(j=0;j<n;j++){
            if(i!=j){
                nodes[i].dist[j] = inf;
                nodes[i].hop[j] = -20;
            }
            else{
                nodes[i].dist[j] = 0;
                nodes[i].hop[j] = -20;
            }

        }
    }
}

void update(int i,int j,int k){
    nodes[i].hop[j] = k;
    nodes[i].dist[j] = nodes[i].dist[k] + nodes[k].dist[j];
}

void dvr(int n){
    int i,j,k;
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
        for(k=0;k<n;k++)
        if(nodes[i].dist[j]>(nodes[i].dist[k] + nodes[k].dist[j]))
        update(i,j,k);
}
```

```
int main(){
    int i, j, n;
    printf("Enter the number of nodes\n");
    scanf("%d",&n);
    init(n);
    printf("Enter the distance vector\n");
    for(i=0;i<n;i++){
            printf("Enter for node %d\n",i);
        for(j=0;j<n;j++){
            scanf("%d",&nodes[i].dist[j]);
        }
    }
    dvr(n);
    printf("\nUpdated distance vector table\n");
    for(i=0;i<n;i++){
        printf("Updated node %c table\n",65+i);
        printf("To\t cost\t hop\n");
        for(j=0;j<n;j++){
            printf("%c\t %d\t %c\n",65+j,nodes[i].dist[j], 65+nodes[i].hop[j]);
        }
    }

    return 0;
}
```

OUTPUT:



```
Enter the number of nodes
5
Enter the distance vector
Enter for node 0
0 12 13 14 999
Enter for node 1
12 0 29 999 21
Enter for node 2
13 29 0 14 16
Enter for node 3
14 999 14 0 4
Enter for node 4
999 21 16 4 0

Updated distance vector table
Updated node A table
To       cost    hop
A        0       -
B        12      -
C        13      -
D        14      -
E        18      D
Updated node B table
To       cost    hop
A        12      -
B        0       -
C        25      A
D        25      E
E        21      -
```

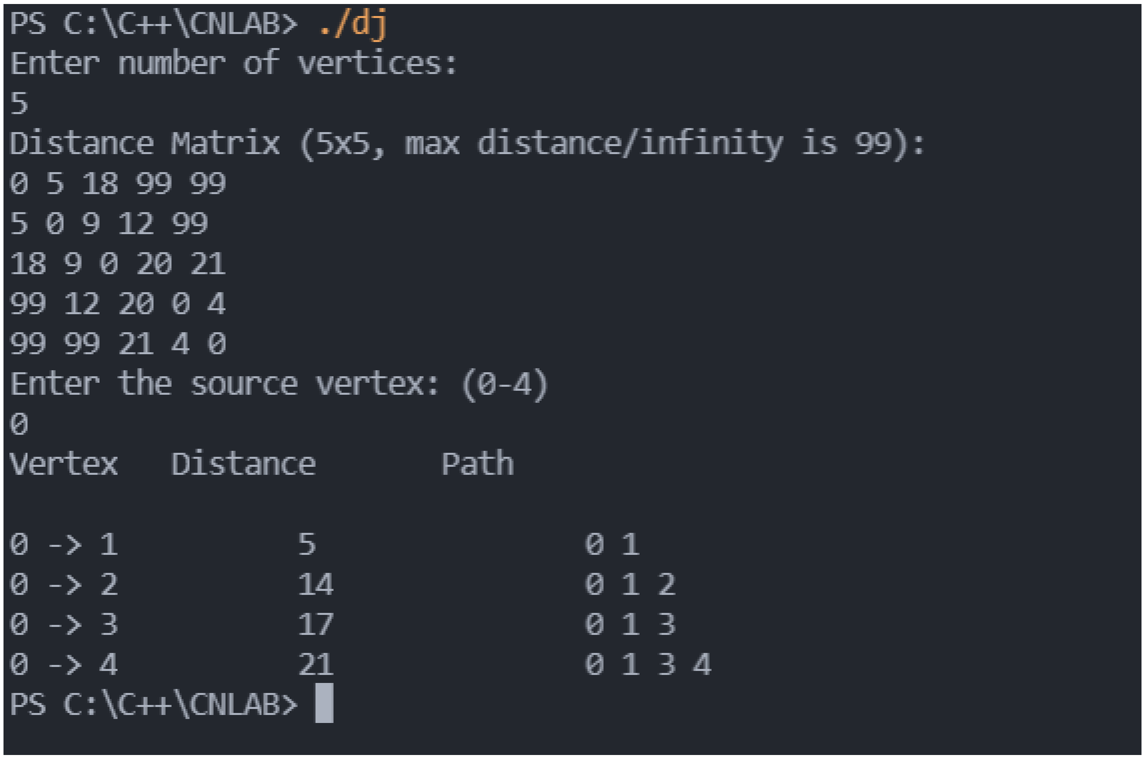


```
Updated node C table
To        cost    hop
A         13      -
B         25      A
C         0       -
D         14      -
E         16      -
Updated node D table
To        cost    hop
A         14      -
B         25      E
C         14      -
D         0       -
E         4       -
Updated node E table
To        cost    hop
A         18      D
B         21      -
C         16      -
D         4       -
E         0       -
```

Prog 3 :
Implement Dijkstra’s algorithm to compute the shortest path for a given topology.

```
#include <bits/stdc++.h>

using namespace std;

int V;


int minDistance(int dist[], bool sptSet[]) {
    int min = 9999, min_index;

    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}

void printPath(int parent[], int j) {
    if (parent[j] == -1)
        return;

    printPath(parent, parent[j]);

    cout << j << " ";
}

void printSolution(int dist[], int n, int parent[]) {
    int src = 0;
    cout << "Vertex\t Distance\tPath" << endl;
    for (int i = 1; i < V; i++) {
        cout << "\n"
             << src << " -> " << i << " \t \t" << dist[i] << "\t\t" << src << " ";
        printPath(parent, i);
    }
}

void dijkstra(int graph[10][10], int src) {
    int dist[V];

    bool sptSet[V];
    int parent[V];

    for (int i = 0; i < V; i++) {
        parent[i] = -1;
        dist[i] = 9999;
        sptSet[i] = false;
    }

    dist[src] = 0;

    for (int count = 0; count < V - 1; count++) {
        int u = minDistance(dist, sptSet);

        sptSet[u] = true;

        for (int v = 0; v < V; v++)

            if (!sptSet[v] && graph[u][v] &&
                dist[u] + graph[u][v] < dist[v]) {
                parent[v] = u;
                dist[v] = dist[u] + graph[u][v];
            }
    }

    printSolution(dist, V, parent);
}

int main() {
    cout<<"Enter number of vertices:"<<endl;
    cin>>V;
    int graph[10][10];
    cout << "Distance Matrix (" << V << "x" << V << ", max distance/infinity is 99): " << endl;
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++)
            cin >> graph[i][j];
    }
    cout << "Enter the source vertex: (0-" << V - 1 << ")" << endl;
    int src;
    cin >> src;

    dijkstra(graph, src);
    cout << endl;
    return 0;
}
```

OUTPUT :



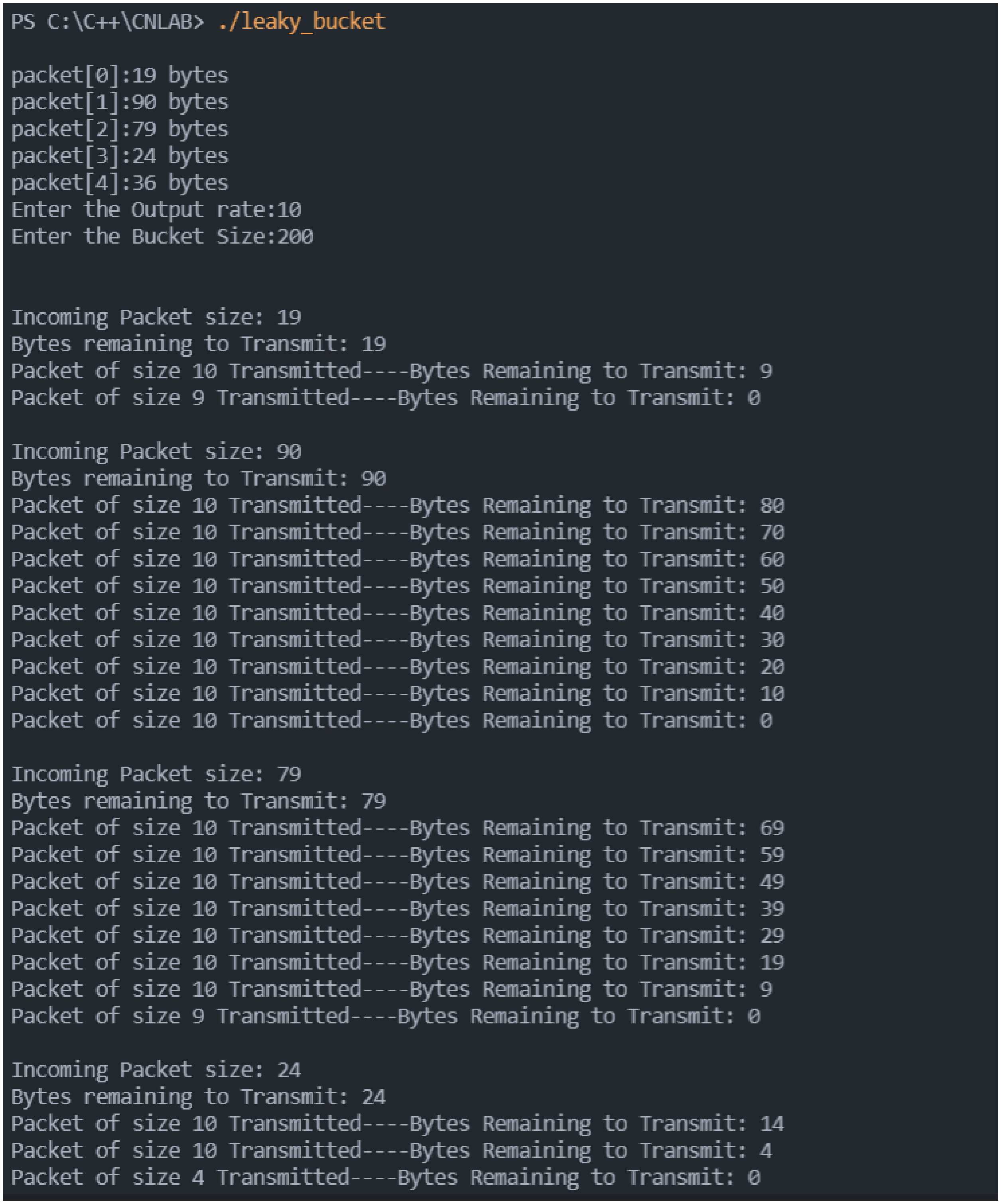
```
PS C:\C++\CNLAB> ./dj
Enter number of vertices:
5
Distance Matrix (5x5, max distance/infinity is 99):
0 5 18 99 99
5 0 9 12 99
18 9 0 20 21
99 12 20 0 4
99 99 21 4 0
Enter the source vertex: (0-4)
0
Vertex   Distance       Path

0 -> 1          5               0 1
0 -> 2          14              0 1 2
0 -> 3          17              0 1 3
0 -> 4          21              0 1 3 4
PS C:\C++\CNLAB> 
```

PROG 4 :
Write a program for congestion control using Leaky bucket algorithm

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<time.h>
#include<iostream>
using namespace std;
#define NOF_PACKETS 5



int main()
{
   srand(time(0));
   int packet_sz[NOF_PACKETS], i, clk, b_size, o_rate, p_sz_rm=0, p_sz, p_time, op;
   for(i = 0; i<NOF_PACKETS; ++i)
      packet_sz[i] = rand() % 100;
   for(i = 0; i<NOF_PACKETS; ++i)
      printf("\npacket[%d]:%d bytes\t", i, packet_sz[i]);
   printf("\nEnter the Output rate:");
   cin>>o_rate;
   printf("Enter the Bucket Size:");
   cin>>b_size;
   for(i = 0; i<NOF_PACKETS; ++i)
   {
      if( (packet_sz[i] + p_sz_rm) > b_size)
         if(packet_sz[i] > b_size)/*compare the packet siz with bucket size*/
            printf("\n\nIncoming packet size (%dbytes) is Greater than bucket capacity
(%dbytes)-PACKET REJECTED", packet_sz[i], b_size);
         else
            printf("\n\nBucket capacity exceeded-PACKETS REJECTED!!");
      else
      {
         p_sz_rm += packet_sz[i];
         printf("\n\nIncoming Packet size: %d", packet_sz[i]);
         printf("\nBytes remaining to Transmit: %d", p_sz_rm);
         while(p_sz_rm>0)
         {
            sleep(1);
            if(p_sz_rm)
            {
               if(p_sz_rm <= o_rate)/*packet size remaining comparing with output rate*/
```

```
                        op = p_sz_rm, p_sz_rm = 0;
                    else
                        op = o_rate, p_sz_rm -= o_rate;
                    printf("\nPacket of size %d Transmitted", op);
                    printf("----Bytes Remaining to Transmit: %d", p_sz_rm);
                }
                else
                {
                     printf("\nNo packets to transmit!!");
                }
            }
        }
    }
}
```

OUTPUT :



```
PS C:\C++\CNLAB> ./leaky_bucket

packet[0]:19 bytes
packet[1]:90 bytes
packet[2]:79 bytes
packet[3]:24 bytes
packet[4]:36 bytes
Enter the Output rate:10
Enter the Bucket Size:200


Incoming Packet size: 19
Bytes remaining to Transmit: 19
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 9
Packet of size 9 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 90
Bytes remaining to Transmit: 90
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 80
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 70
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 60
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 50
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 40
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 30
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 20
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 10
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 79
Bytes remaining to Transmit: 79
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 69
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 59
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 49
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 39
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 29
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 19
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 9
Packet of size 9 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 24
Bytes remaining to Transmit: 24
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 14
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 4
Packet of size 4 Transmitted----Bytes Remaining to Transmit: 0
```

PROG 5 :
Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

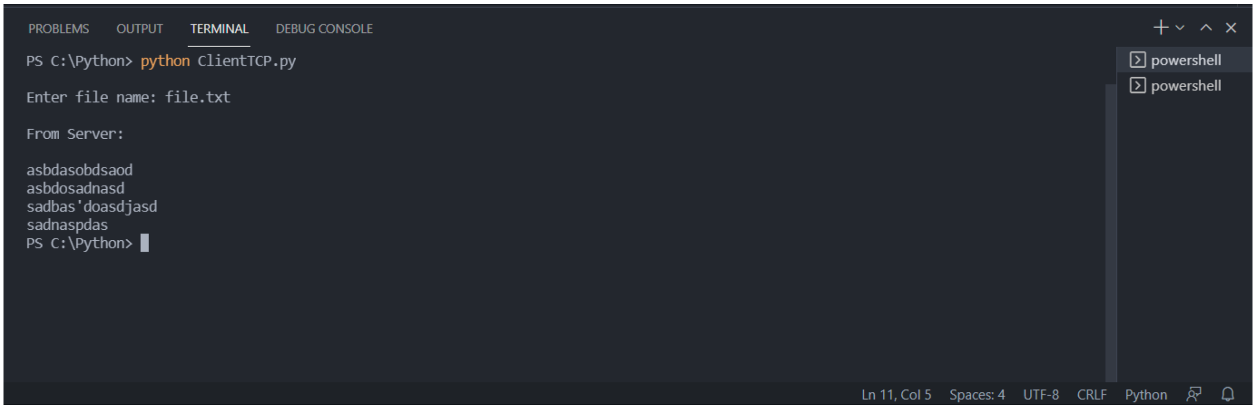
ClientTCP.py :

```
from socket import *
sentence = input("\nEnter file name: ")
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print ('\nFrom Server:\n')
print(filecontents)
clientSocket.close()
```

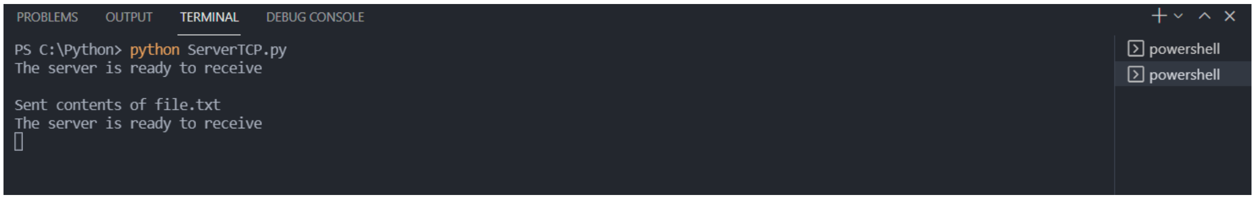
ServerTCP.py :

```
from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
    print ("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()

    file=open(sentence,"r")
    l=file.read(1024)
    connectionSocket.send(l.encode())
    print ('\nSent contents of ' + sentence)
    file.close()
    connectionSocket.close()
```

OUTPUT :



```
PS C:\Python> python ClientTCP.py

Enter file name: file.txt

From Server:

asbdasobdsaod
asbdosadnasd
sadbas'doasdjasd
sadnaspdas
PS C:\Python> 
```



```
PS C:\Python> python ServerTCP.py
The server is ready to receive

Sent contents of file.txt
The server is ready to receive
```

PROG 6 :
Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

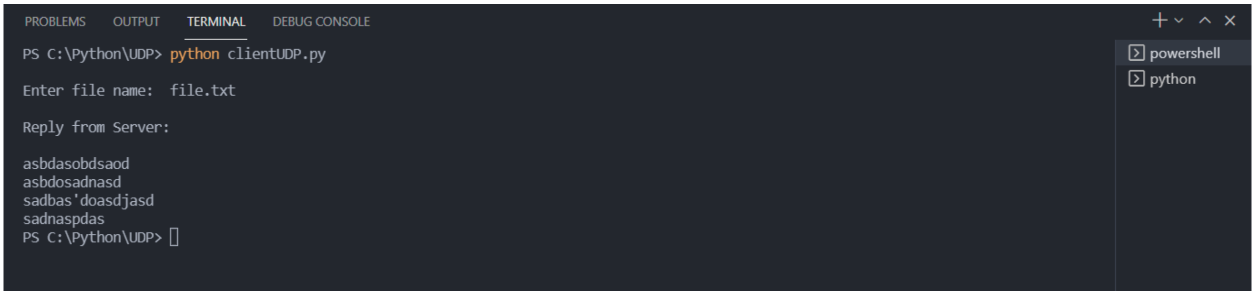
ClientUDP.py

```
from socket import *
sentence = input("\nEnter file name:  ")
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048)
print ('\nReply from Server:\n')
print (filecontents.decode("utf-8"))
clientSocket.close()
```

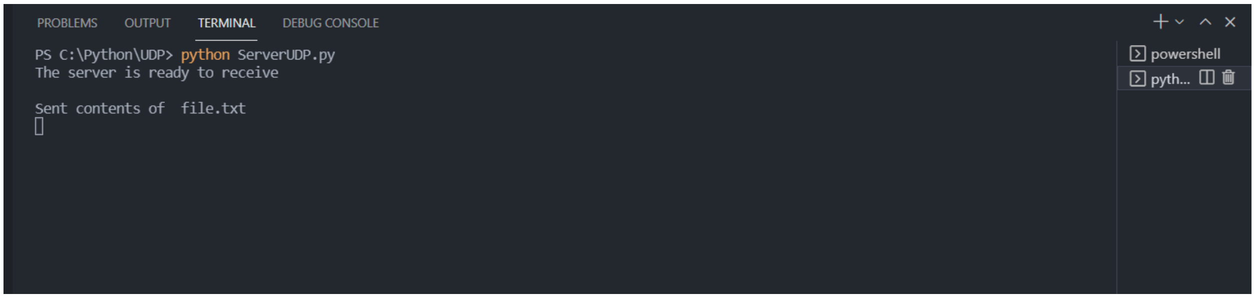
ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('DESKTOP-4S4CA6T', serverPort))
print ("The server is ready to receive")
while 1:
   sentence,clientAddress = serverSocket.recvfrom(2048)
file=open(sentence,"r")
l=file.read(2048)
serverSocket.sendto(bytes(l,"utf-8"),clientAddress)
print ('\nSent contents of ', end = '')
print (sentence)
file.close()
```

OUTPUT :



```
PS C:\Python\UDP> python clientUDP.py

Enter file name:  file.txt

Reply from Server:

asbdasobdsaod
asbdosadnasd
sadbas'doasdjasd
sadnaspdas
PS C:\Python\UDP> 
```



```
PS C:\Python\UDP> python ServerUDP.py
The server is ready to receive

Sent contents of  file.txt
```