CSE62 COMPUTER NETWORKS LAB MANUAL

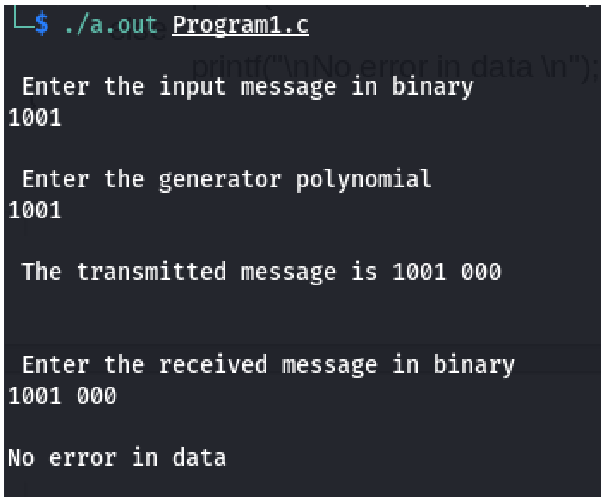
Program	Program statement
1	Write a program for error detecting code-using CRC-CCITT (16-bits).
2	Write a program to implement Go-Back N and Selective repeat sliding window protocol.
3	Write a program for implementation of stop and wait.
4	Using TCP/IP Sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.
5	Design, develop, and execute a program in C under UNIX / LINUX environment to implement a simple echo server and demonstrate its working. Both the server and client are to be connectionless and use UDP. The system works as follows: Client reads a line from the standard input and writes the line to the server; the server reads a line from its network input and echoes the line back to the client; the client reads the echoed line and prints it on its standard output.
6	Write a program for Congestion control using the leaky bucket algorithm
7	Write a program for Distance Vector Algorithm to find suitable path for transmission.
8	Write a program for Link State Algorithm to find suitable path for transmission.
9	Write a program for encryption and decryption using RSA algorithm.
10	Write a program to implement Diffie Hellman Key exchange.
11	a)Simulate Capturing and analysing Ethernet frames. b)Simulate capturing a bulk TCP transfer from your computer to a remote server.
12	Simulate i) Analysis of ICMP and PING messages ii) Analysis of ICMP and Traceroute

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

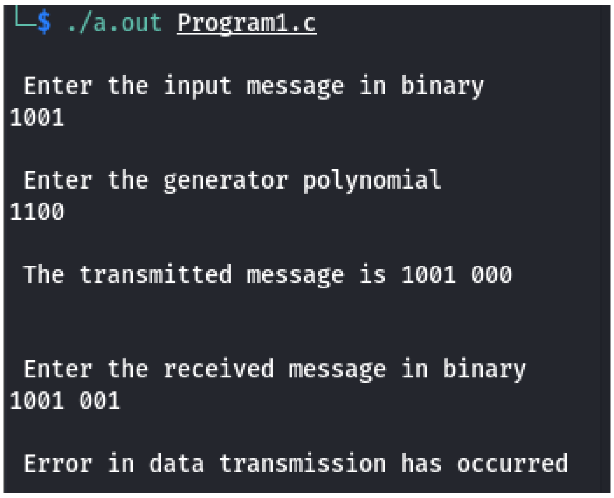
```
#include<stdio.h>
#include<string.h>

int crc(char *input, char *output, char *gp)
{
        int i, j;
        for(i=0; i<strlen(input); i++)
                if(output[i] == '1')

        for(j=0; j<strlen(gp); j++)
        {
                if (output[i+j]==gp[j])
                output[i+j]='0';

                else
                output[i+j]='1';
        }

        for(i=0; i<strlen(output); i++)
                if(output[i] == '1')
                        return 1;
                return 0;
}
int main()
{
        char input[50],output[50], recv[50], gp[50];
        int i;

        printf("\n Enter the input message in binary\n");
        scanf("%s",input);
        printf("\n Enter the generator polynomial\n");
        scanf("%s",gp);

        strcpy(output, input);

        for(i=1; i<strlen(gp); i++)
                strcat(output,"0");

        crc(input,output,gp);

        printf("\n The transmitted message is %s %s\n",input, output+strlen
(input));
        printf("\n\n Enter the received message in binary \n");
        scanf("%s",recv);
        if(crc(input,recv,gp))
                printf("\n Error in data transmission has occurred \n");

        else
```

```
                printf("\nNo error in data \n");
}
```

Output:



```
└─$ ./a.out Program1.c

 Enter the input message in binary
1001

 Enter the generator polynomial
1001

 The transmitted message is 1001 000


 Enter the received message in binary
1001 000

No error in data
```



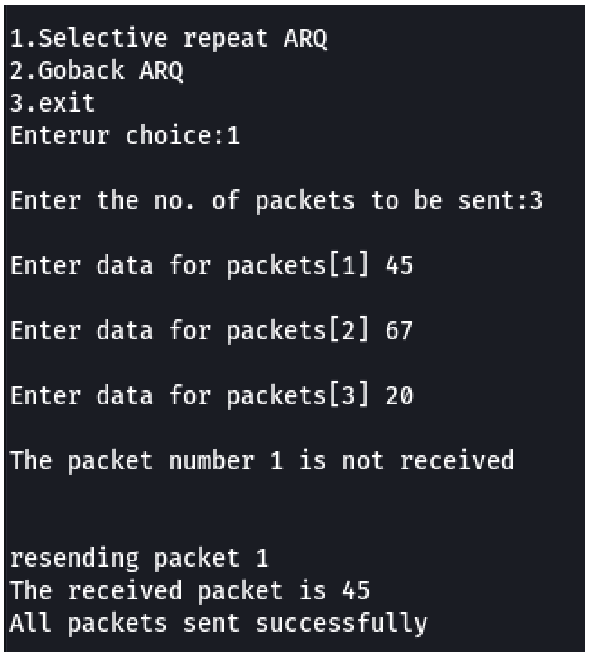
```
└─$ ./a.out Program1.c

 Enter the input message in binary
1001

 Enter the generator polynomial
1100

 The transmitted message is 1001 000


 Enter the received message in binary
1001 001

 Error in data transmission has occurred
```

2. Write a program to implement Go-Back N and Selective repeat sliding window protocol.

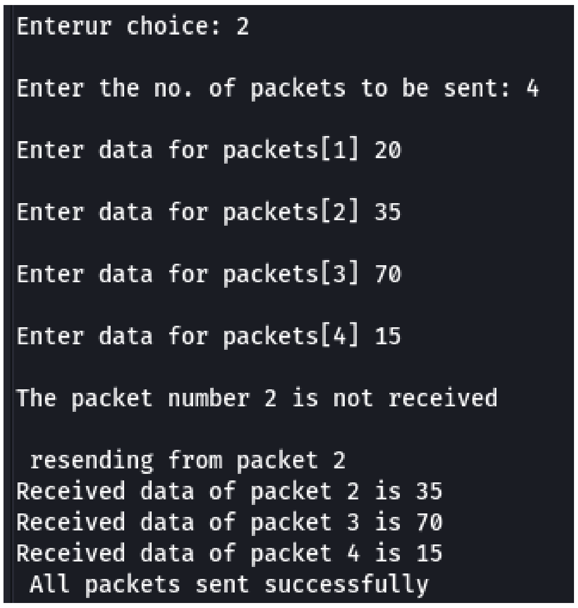
```
#include<stdio.h>

#include<stdlib.h>

#include<math.h>


int n,r;

struct frame

{

        char ack;

        int data;

} frm[10];


int sender(void);

void recvfrm(void);

void resend(void);

void resend1(void);

void goback(void);
void selective(void);

int main()
{
int c;
do
{
        printf("\n\n1.Selective repeat ARQ\n2.Goback ARQ\n3.exit");
        printf("\nEnter choice:");
        scanf("%d",&c);

        switch(c)
        {
                case 1:selective();
                break;
                case 2:goback();
                break;
                case 3:exit(0);
                break;
        }
} while(c>=4);
}

void goback()
{
        sender();
        recvfrm();
        resend1();
        printf("\n All packets sent successfully\n");
```

```
}

void selective()
{
        sender();
        recvfrm();
        resend();
        printf("\nAll packets sent successfully");
}

int sender()
{
        int i;
        printf("\nEnter the no. of packets to be sent:");
        scanf("%d",&n);

        for(i=1;i<=n;i++)
        {
                printf("\nEnter data for packets[%d]",i);
                scanf("%d",&frm[i].data);
                frm[i].ack='y';
        }
return 0;
}

void recvfrm()
{
        int i;
        rand();
        r=rand()%n;
        frm[r].ack='n';

        for(i=1;i<=n;i++)
        {
                if(frm[i].ack=='n')
                printf("\nThe packet number %d is not received\n",r);
        }
}

void resend()
{
        printf("\nResending packet %d",r);
        sleep(2);
        frm[r].ack='y';
        printf("\nThe received packet is %d",frm[r].data);
}

void resend1()
{
        int i;
        printf("\n Resending from packet %d",r);

        for(i=r;i<=n;i++)
        {
                sleep(2);
                frm[i].ack='y';
                printf("\nReceived data of packet %d is %d",i,frm[i].data);
        }
}
```

Output:



```
1.Selective repeat ARQ
2.Goback ARQ
3.exit
Enterur choice:1

Enter the no. of packets to be sent:3

Enter data for packets[1] 45

Enter data for packets[2] 67

Enter data for packets[3] 20

The packet number 1 is not received



resending packet 1
The received packet is 45
All packets sent successfully
```

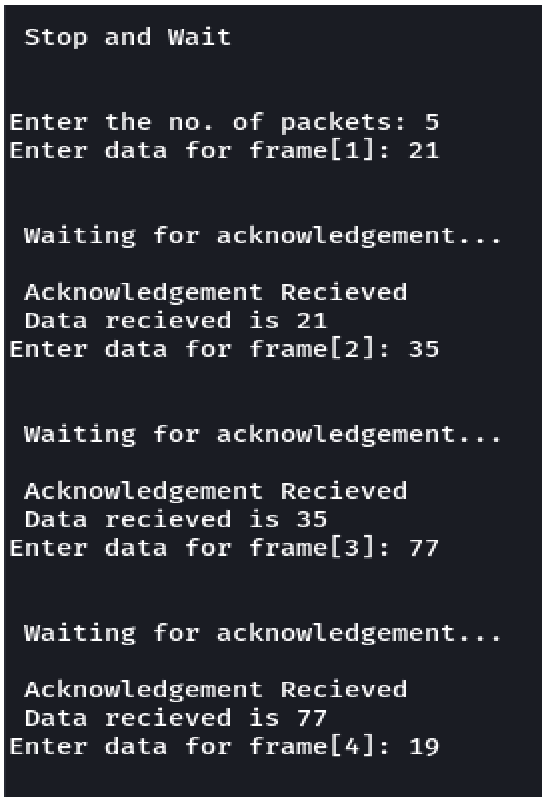


```
Enterur choice: 2

Enter the no. of packets to be sent: 4

Enter data for packets[1] 20

Enter data for packets[2] 35

Enter data for packets[3] 70

Enter data for packets[4] 15

The packet number 2 is not received

 resending from packet 2
Received data of packet 2 is 35
Received data of packet 3 is 70
Received data of packet 4 is 15
 All packets sent successfully
```

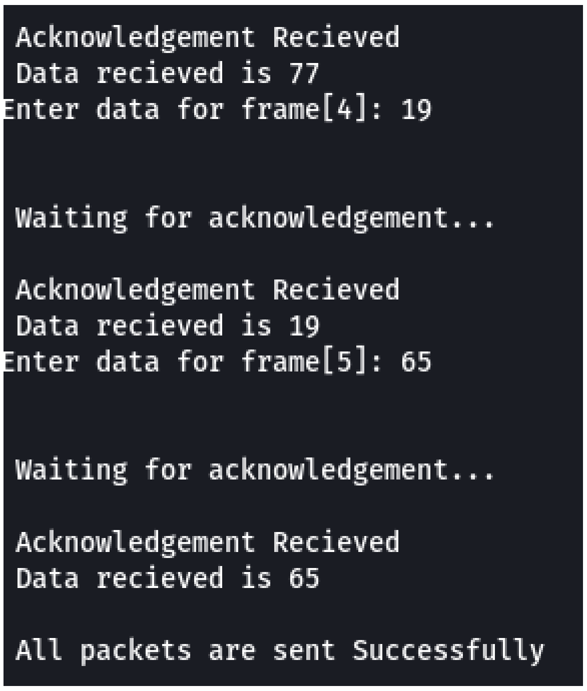
3. Write a program for implementation of stop and wait.

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

int nor;

struct frame
{
  char ack;
  int data;
}   frm [10];

void revack(int j);
void resend(int j);

int main()
{
  int i, n;

  printf("\n \n Stop and Wait");
  printf("\n \n \nEnter the no. of packets: ");
  scanf("%d", &n);

  for(i=1; i<=n; i++)
  {
            printf("Enter data for frame[%d]: ", i);
            scanf("%d", &frm[i].data);

            printf("\n \n Waiting for acknowledgement...");

            frm[i].ack ='y';
            revack(i);
  }

printf("\n All packets are sent Successfully");
//getch();
return 0;
}

void revack(int j)
{
  int i, r, n;
  rand();
  r = rand()%n;

  if(j==r)
  {
            frm[r].ack='n';
            printf("\n Packet not recieved");
            resend(j);
  }

  else
  {
            printf("\n \n Acknowledgement Recieved");
            printf("\n Data recieved is %d \n", frm[j].data);
  }
}

void resend(int j)
{
  printf("\n \n Resending the Frame...");
  sleep(2);

  frm[j].ack='y';
  printf("\n Data recieved is %d \n", frm[j].data);
}
```

Output:



```
 Stop and Wait


Enter the no. of packets: 5
Enter data for frame[1]: 21


 Waiting for acknowledgement...

 Acknowledgement Recieved
 Data recieved is 21
Enter data for frame[2]: 35


 Waiting for acknowledgement...

 Acknowledgement Recieved
 Data recieved is 35
Enter data for frame[3]: 77


 Waiting for acknowledgement...

 Acknowledgement Recieved
 Data recieved is 77
Enter data for frame[4]: 19
```



```
 Acknowledgement Recieved
 Data recieved is 77
Enter data for frame[4]: 19


 Waiting for acknowledgement...

 Acknowledgement Recieved
 Data recieved is 19
Enter data for frame[5]: 65


 Waiting for acknowledgement...

 Acknowledgement Recieved
 Data recieved is 65

 All packets are sent Successfully
```

4. 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.

Header Filename: P4headers.h

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <fcntl.h>
#include <string.h>
#include <stdlib.h>
#define SERV_TCP_PORT 6880
#define SERV_HOST_ADDR "127.0.0.1"
```

Client.c

```
#include "P4headers.h"

int main()
{
        int sockfd;
        struct sockaddr_in serv_addr, cli_addr;
        char filename[100], buf[1000];
        int n;

        serv_addr.sin_family=AF_INET;
        serv_addr.sin_addr.s_addr=inet_addr(SERV_HOST_ADDR);
        serv_addr.sin_port=htons(SERV_TCP_PORT);

        if((sockfd=(socket(AF_INET,SOCK_STREAM,0)))<0)
        {
                printf("Client: can’t open stream socket\n");
                exit(0);
        }

        else
                printf("Client: stream socket opened successfully\n");


        if(connect (sockfd,(struct sockaddr *) &serv_addr, sizeof(serv_addr))<0)
        {
                printf("Client: cant connect to server\n");
                exit(0);
        }

        else
                printf("Client: connected to server successfully\n");


        printf("\n Enter the file name to be displayed :");
        scanf("%s",filename);

        write(sockfd, filename, strlen(filename));
        printf("\n filename transferred to server\n");

        n=read(sockfd,buf,1000);
        buf[n]='\0';

        printf("\n Client : Displaying file content of %s\n", filename);

        puts(buf);
        close(sockfd);

        exit(0);
}
```

<u>Server.c</u>

```
#include"P4headers.h"
```

```
int main()
{
	int sockfd,newsockfd,clilen;
	struct sockaddr_in cli_addr, serv_addr;

	char filename[25],buf[1000];
	int n,m,fd,size;

	serv_addr.sin_family=AF_INET;
	serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
	serv_addr.sin_port=htons(SERV_TCP_PORT);

	if((sockfd=(socket(AF_INET,SOCK_STREAM,0)))<0)
	{
		printf("Server: can’t open stream socket\n");
		exit(0);
	}

	else
		printf("Server: stream socket opened successfully\n");

	if((bind(sockfd,(struct sockaddr *) &serv_addr, sizeof(serv_addr)))<0)
	{
		printf("Server:cant bind local address\n");
		exit(0);
	}

	else
		printf("Server: bind to local address\n");

	listen(sockfd,5);
	printf("\n SERVER : Waiting for client...\n");

	clilen=sizeof(cli_addr);
	newsockfd=accept(sockfd,(struct sockaddr*)&cli_addr,&clilen);


	if(newsockfd<0)
	{
		printf("Server:accept error\n");
		exit(0);
	}

	else
		printf("Server: accepted\n");

	n=read(newsockfd,filename,25);
	filename[n]='\0';
```

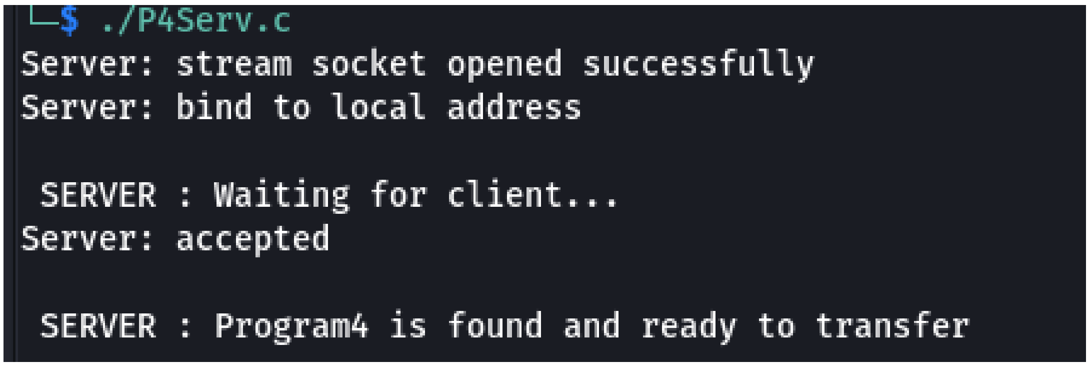
```
        printf("\n SERVER : %s is found and ready to transfer \n",filename);
        fd=open(filename,O_RDONLY);

        if(fd==-1)
        {
                write(newsockfd,"File doesn't exist",25);
                exit(0);
        }

        size=lseek(fd,0,2);
        lseek(fd,0,0);

        n=read(fd,buf,size);
        buf[n]='\0';

        write(newsockfd,buf,n);
        printf("\n transfer success\n");

        puts(buf);
        close(newsockfd);

        exit(0);
}
```

Output:

Server

gcc P4Server.c -o P4Serv.c

./P4Serv.c

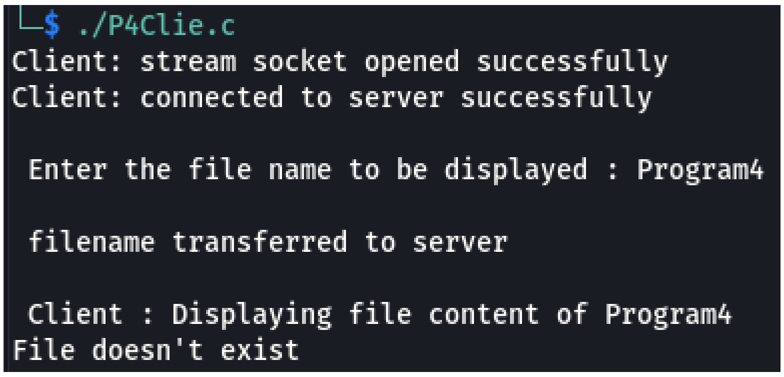


```
└─$ ./P4Serv.c
Server: stream socket opened successfully
Server: bind to local address

 SERVER : Waiting for client...
Server: accepted

 SERVER : Program4 is found and ready to transfer
```

Client

gcc P4Client.c -o P4Clie.c

./P4Clie.c



```
└─$ ./P4Clie.c
Client: stream socket opened successfully
Client: connected to server successfully

 Enter the file name to be displayed : Program4

 filename transferred to server

 Client : Displaying file content of Program4
File doesn't exist
```

5. Design, develop, and execute a program in C under UNIX / LINUX environment to implement a simple echo server and demonstrate its working. Both the server and client are to be connectionless and use UDP. The system works as follows: Client reads a line from the standard input and writes the line to the server; the server reads a line from its network input and echoes the line back to the client; the client reads the echoed line and prints it on its standard output.

Server.c

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <stdlib.h>

int main()
{
        int udpSocket, nBytes;
        char buffer[1024];

        struct sockaddr_in serverAddr, clientAddr;
        struct sockaddr_storage serverStorage;
        socklen_t addr_size, client_addr_size;
        int i;

        udpSocket = socket(PF_INET, SOCK_DGRAM, 0);

        serverAddr.sin_family = AF_INET;
        serverAddr.sin_port = htons(7891);
        serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

        memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);
        bind(udpSocket, (struct sockaddr *) &serverAddr, sizeof(serverAddr));

        addr_size = sizeof serverStorage;

while(1)
{
nBytes = recvfrom(udpSocket,buffer,1024,0,(struct sockaddr *) &serverStorage,
&addr_size);

for(i=0;i<nBytes-1;i++)
        buffer[i] = toupper(buffer[i]);

sendto(udpSocket,buffer,nBytes,0,(struct sockaddr *)
&serverStorage,addr_size);
}

return 0;
}
```

Client.c

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>

int main()
{
        int clientSocket, portNum, nBytes;
        char buffer[1024];

        struct sockaddr_in serverAddr;
        socklen_t addr_size;

        clientSocket = socket(PF_INET, SOCK_DGRAM, 0);

        serverAddr.sin_family = AF_INET;
        serverAddr.sin_port = htons(7891);
        serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

        memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);


        addr_size = sizeof serverAddr;

while(1)
{
        printf("Type a sentence to send to server:\n");
        fgets(buffer,1024,stdin);
        printf("You typed: %s",buffer);

        nBytes = strlen(buffer) + 1;
```

```
            sendto(clientSocket,buffer,nBytes,0,(struct sockaddr *)
&serverAddr,addr_size);
            nBytes = recvfrom(clientSocket,buffer,1024,0,NULL, NULL);

            printf("Received from server: %s\n",buffer);
}
return 0;

}
```

Output:

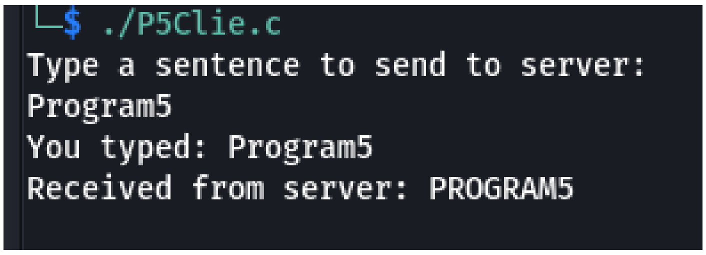
Server

```
gcc P5Server.c -o P5Serv.c
./P5Serv.c
```

(No Output Shown on Server Side)

Client

```
gcc P5Client.c -o P5Clie.c
./P5Clie.c
```

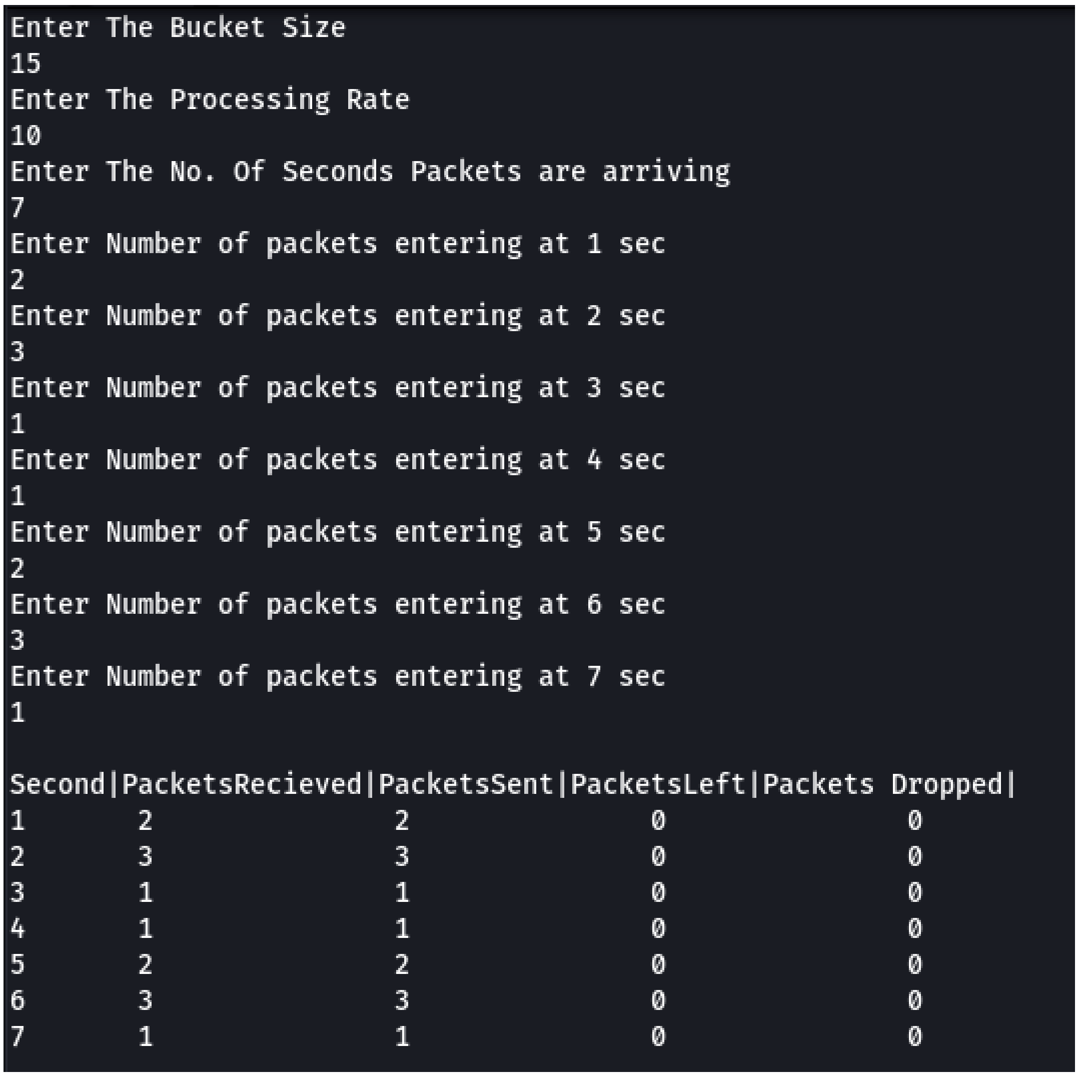


```
└─$ ./P5Clie.c
Type a sentence to send to server:
Program5
You typed: Program5
Received from server: PROGRAM5
```

6.Write a program for Congestion control using the leaky bucket algorithm

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

int min(int x, int y)
{
        if(x<y)
        return x;

        else
        return y;
}

int main()
{
        int drop=0,mini,nsec,cap,count=0,i,inp[25],process;
        system("clear");

        printf("Enter The Bucket Size: ");
        scanf("%d",&cap);
        printf("\n Enter The Processing Rate: ");
        scanf("%d",&process);

        printf("\n Enter The No. Of Seconds Packets are arriving: ");
        scanf("%d",&nsec);

        for(i=1;i<=nsec;i++)
        {
                printf("\n Enter Number of packets entering at %d sec: ",i);
                scanf("%d",&inp[i]);
        }

        printf("\nSecond|PacketsRecieved|PacketsSent|PacketsLeft|Packets
Dropped|\n");

        for(i=1;i<=nsec;i++)
        {
                count=count+inp[i];
                if(count>cap)
                {
                        drop=count-cap;
                        count=cap;
                }
                printf("%d",i);
                printf("\t%d",inp[i]);

                mini=min(count,process);
                printf("\t\t%d",mini);

                count=count-mini;

                printf("\t\t%d",count);
                printf("\t\t%d\n",drop);

                drop=0;
        }

        for(;count!=0;i++)
        {
                if(count>cap)
                {
                        drop=count-cap;
                        count=cap;
                }

                printf("%d",i);
                printf("\t0");
                mini=min(count,process);
                printf("\t\t%d",mini);

                count=count-mini;

                printf("\t\t%d",count);
                printf("\t\t%d\n",drop);
        }
}
```

Output:



```
Enter The Bucket Size
15
Enter The Processing Rate
10
Enter The No. Of Seconds Packets are arriving
7
Enter Number of packets entering at 1 sec
2
Enter Number of packets entering at 2 sec
3
Enter Number of packets entering at 3 sec
1
Enter Number of packets entering at 4 sec
1
Enter Number of packets entering at 5 sec
2
Enter Number of packets entering at 6 sec
3
Enter Number of packets entering at 7 sec
1

Second|PacketsRecieved|PacketsSent|PacketsLeft|Packets Dropped|
1       2               2               0               0
2       3               3               0               0
3       1               1               0               0
4       1               1               0               0
5       2               2               0               0
6       3               3               0               0
7       1               1               0               0
```