1. .tcl

```tcl
# Letter S is capital
set ns [new Simulator]
##### open a nam AND trace file in write mode
# nf nam filename
set nf [open /home/mca/sandhya/1.nam w]
$ns namtrace-all $nf

# tf trace filename
set tf [open /home/mca/sandhya/1.tr w]
$ns trace-all $tf
proc finish { } {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam 1.nam &
exit 0
}
# # creates 3 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
# establishing links
# you need to modify the bandwidth
# to observe the variation in packet.
$ns duplex-link $n0 $n1 200Mb 10ms DropTail
$ns duplex-link $n1 $n2 100Kb 1000ms DropTail
#set queue size
$ns queue-limit $n0 $n1 10
$ns queue-limit $n1 $n2 10

# attaching transport layer protocols------START
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp

# attaching application layer protocols----MID
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packet_size_ 500

# creating sink(destination) node----------END
set null [new Agent/Null]
$ns attach-agent $n2 $null
$ns connect $udp $null

$ns at 0.1 "$cbr start"
$ns at 1.0 "finish"
$ns run

```

Output :

```
ns 1.tcl
```

1.awk

```
#awk file run in newterminal
BEGIN{
c=0;

}

{

if($1=="d")

{
c++;
printf("%s\t%s\n",$5,$11);
}
} END{
printf("The number of packets dropped =%d\n",c);

}

```

output:-open new terminal

```
awk -f 1.awk 1.tr

The number of packets dropped =28
```

8.tcl

```
set ns [new Simulator]
set nf [open 8b.nam w]
$ns namtrace-all $nf
set tf [open 8b.tr w]
$ns trace-all $tf

set n0 [$ns node]
$n0 color "red"
$n0 label "src1"

set n1 [$ns node]
set n2 [$ns node]
$n2 color "red"
$n2 label "src2"

set n3 [$ns node]
$n3 color "blue"
$n3 label "dest2"

set n4 [$ns node]
set n5 [$ns node]
$n5 color "blue"
$n5 label "dest1"
$ns make-lan "$n0 $n1 $n2 $n3 $n4" 1Mb 100ms LL Queue/DropTail Mac/802_3

$ns duplex-link $n4 $n5 1Kb 1ms DropTail
$ns queue-limit $n4 $n5 1

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.0001

set sink5 [new Agent/TCPSink]
$ns attach-agent $n5 $sink5
$ns connect $tcp0 $sink5

set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set packetSize_ 600
$ftp2 set interval_ 0.001
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3 $ns connect $tcp2 $sink3

proc finish { } {
global ns nf tf
$ns flush-trace
close $tf
close $nf
exec nam 8b.nam &
exit 0
}

$ns at 0.1 "$ftp0 start"
$ns at 5 "$ftp0 stop"
$ns at 7 "$ftp0 start"
$ns at 0.2 "$ftp2 start"
$ns at 8 "$ftp2 stop"
$ns at 14 "$ftp0 stop"
$ns at 10 "$ftp2 start"
$ns at 15 "$ftp2 stop"
$ns at 16 "finish"
$ns run
```

8b.awk

```
#awk file run in new terminal
BEGIN{
c=0;
}
{
if($1=="d")
{
c++;
}
}
END{
printf("The number of %s packets dropped =%d\n",$5,c);   }
```

2. output:-

```
the number of ack packets dropped=3
```

7

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
set nf [open 7.nam w]
$ns namtrace-all $nf
set tf [open 7.tr w]
$ns trace-all $tf

#Create five nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n4 1Mb 10ms DropTail
$ns duplex-link $n1 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n3 1Mb 10ms DropTail
$ns duplex-link $n4 $n2 1Mb 10ms DropTail

#Setup a TCP connection
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp

#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n2 $null
$ns connect $udp $null

#Setup a CBR over UDP
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packet_size_ 500

#Schedule events for the CBR and FTP agents $ns at 0.0 "$cbr start"
$ns at 0.0 "$ftp start"
$ns at 20.0 "$ftp stop"
$ns at 20.0 "$cbr stop"

#Define a 'finish' procedure
proc finish {} {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam 7.nam &
exit 0
}
$ns at 10.0 "finish"

#Run the simulation
$ns run
```

3. 7.awk:-

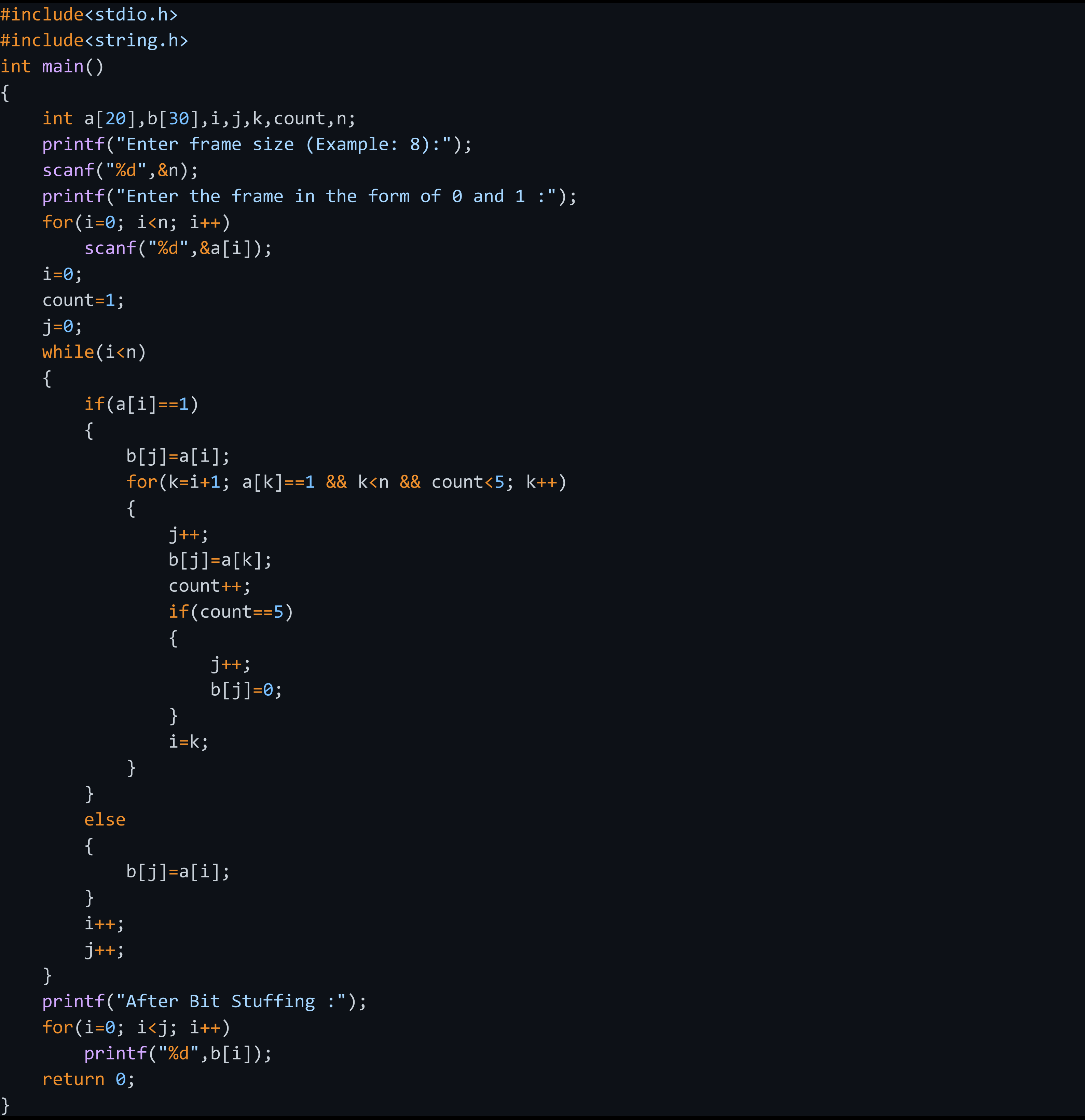
```
BEGIN{
udp=0;
tcp=0;
}
{
if($1 == "r" && $5 == "cbr")
{
udp++;
}
else if($1 == "r" && $5 == "tcp")
{
tcp++;
}
}
END{
printf("Number of packets sent by TCP = %d\n", tcp);
printf("Number of packets sent by UDP=%d\n",udp);
```

Program 4 :

Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism.

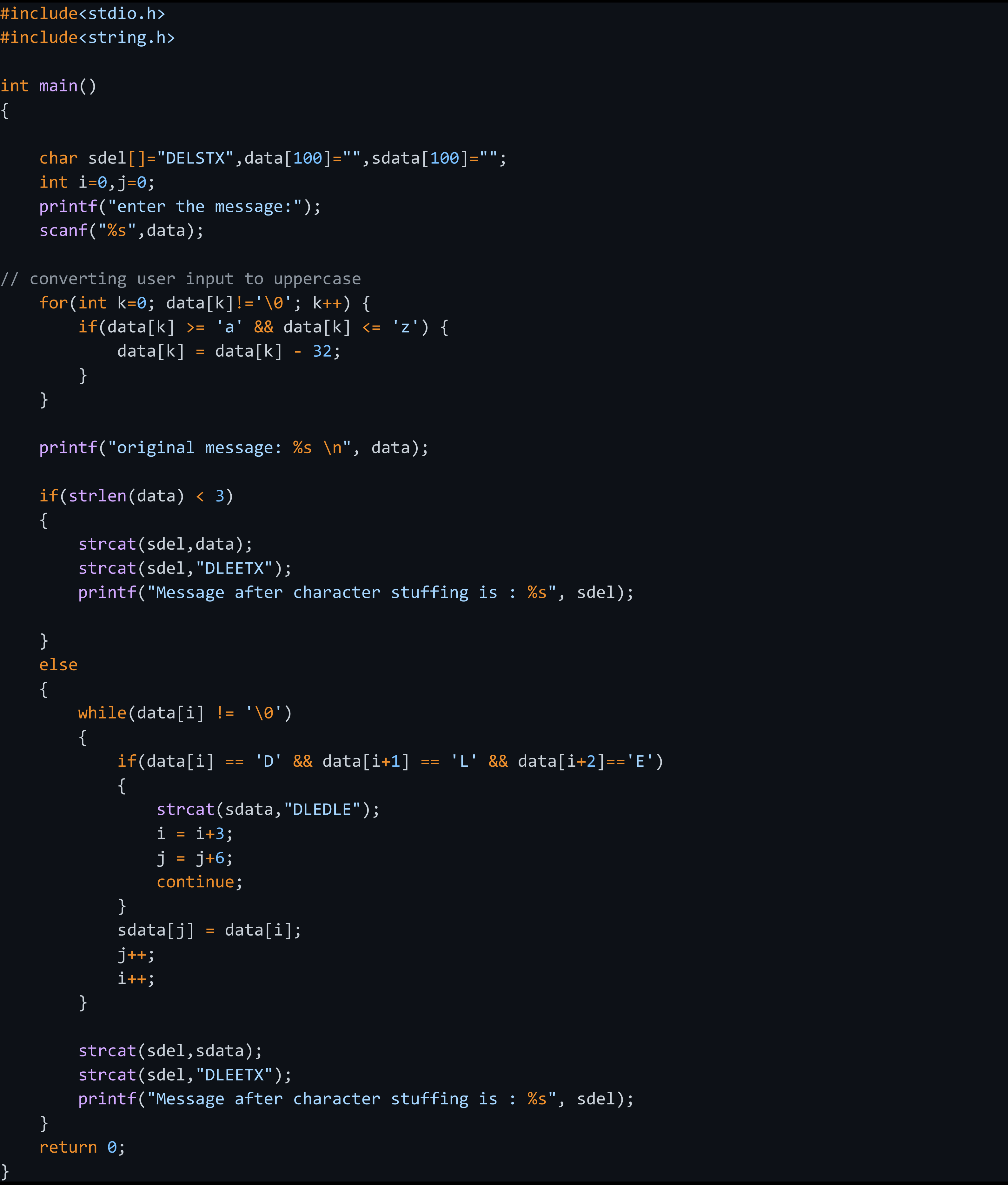
```c
#include<stdio.h>
int main()
{
    int w,i,f,frames[50];

    printf("Enter window size: ");
    scanf("%d",&w);
    printf("\nEnter number of frames to transmit: ");
    scanf("%d",&f);

    printf("\nEnter %d frames: ",f);
    for(i=1; i<=f; i++)
        scanf("%d",&frames[i]);

    printf("\nWith sliding window protocol the frames will be sent in the following manner
                                                                 (assuming no corruption of frames)\n\n");
    printf("After sending %d frames at each stage sender waits for acknowledgement sent by the receiver\n\n",w);

    for(i=1; i<=f; i++)
    {
        if(i%w==0)
        {
            printf("%d\n",frames[i]);
            printf("Acknowledgement of above frames sent is received by sender\n\n");
        }
        else
            printf("%d ",frames[i]);
    }

    if(f%w!=0)
        printf("\nAcknowledgement of above frames sent is received by sender\n");

    return 0;
}
```

PROGRAM2-A) Bit Stuffing Program in C



```c
#include<stdio.h>
#include<string.h>
int main()
{
    int a[20],b[30],i,j,k,count,n;
    printf("Enter frame size (Example: 8):");
    scanf("%d",&n);
    printf("Enter the frame in the form of 0 and 1 :");
    for(i=0; i<n; i++)
        scanf("%d",&a[i]);
    i=0;
    count=1;
    j=0;
    while(i<n)
    {
        if(a[i]==1)
        {
            b[j]=a[i];
            for(k=i+1; a[k]==1 && k<n && count<5; k++)
            {
                j++;
                b[j]=a[k];
                count++;
                if(count==5)
                {
                    j++;
                    b[j]=0;
                }
                i=k;
            }
        }
        else
        {
            b[j]=a[i];
        }
        i++;
        j++;
    }
    printf("After Bit Stuffing :");
    for(i=0; i<j; i++)
        printf("%d",b[i]);
    return 0;
}
```

PROGRAM2-B) Byte Stuffing Program in C



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

int main()
{

    char sdel[]="DELSTX",data[100]="",sdata[100]="";
    int i=0,j=0;
    printf("enter the message:");
    scanf("%s",data);

// converting user input to uppercase
    for(int k=0; data[k]!='\0'; k++) {
        if(data[k] >= 'a' && data[k] <= 'z') {
            data[k] = data[k] - 32;
        }
    }

    printf("original message: %s \n", data);

    if(strlen(data) < 3)
    {
        strcat(sdel,data);
        strcat(sdel,"DLEETX");
        printf("Message after character stuffing is : %s", sdel);

    }
    else
    {
        while(data[i] != '\0')
        {
            if(data[i] == 'D' && data[i+1] == 'L' && data[i+2]=='E')
            {
                strcat(sdata,"DLEDLE");
                i = i+3;
                j = j+6;
                continue;
            }
            sdata[j] = data[i];
            j++;
            i++;
        }

        strcat(sdel,sdata);
        strcat(sdel,"DLEETX");
        printf("Message after character stuffing is : %s", sdel);
    }
    return 0;
}
```