## 01] Bit Stufing - 29 | 05

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

void main()
{
    char a[20],fs[50]="",t[6],r[5];
    int i,j,p=0,q=0;

    printf("enter bit string: ");
    scanf("%s", a);

    strcat(fs,"01111110");
    if(strlen(a)<5)
    {
        strcat(fs,a);
    }
    else
    {
        for(i=0;i<strlen(a)-4;i++)
        {
            for(j=i;j<i+5;j++)
            {
                t[p++]=a[j];
            }
            t[p]='\0';
            if(strcmp(t,"11111")==0)
            {
                strcat(fs,"111110");
                i=j-1;
            }
            else
            {
                r[0]=a[i];
                r[1]='\0';
                strcat(fs,r);
            }
            p=0;
        }
    for(q=i;q<strlen(a);q++)
    {
        t[p++]=a[q];
    }
        t[p]='\0';
        strcat(fs,t);
    }
    strcat(fs,"01111110");
    printf("After stuffing: %s", fs);
}
```

**02] Character Stuffing - 28 | 03**

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

void main()
{
        char a[30],fs[50000]="",t[3],sd[3],ed[3],x[3],s[3],d[3],y[3];
        int i,j,p=0,q=0;

        printf("Enter the characters to be stuffed: ");
        scanf("%s",a);

        printf("\n Enter the starting delimiter character: ");
        scanf("%s",sd);

        printf("\n Enter the ending delimiter character:: ");
        scanf("%s",ed);

        x[0]=s[0]=s[1]=sd[0];
        x[1]=s[2]='\0';

        y[0]=d[0]=d[1]=ed[0];
        d[2]=y[1]='\0';

        strcat(fs,x);
        for(i=0;i<strlen(a);i++)
        {
                t[0] = a[i];
                t[1] = '\0';
                if(t[0]==sd[0])
                        strcat(fs,s);
                else
                if(t[0]==ed[0])
                        strcat(fs,d);
                else
                        strcat(fs,t);
        }
        strcat(fs,y);
        printf("\n After stuffing: %s",fs);
}
```

## 03] Leaky Bucket Algorithm - 34 | 9 - 9 - 16 | 03

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

#define MIN(x, y) ((x > y) ? y : x)

int main()
{
    int orate, drop = 0, cap, x, count = 0,inp[10] = {0},i = 0, nsec, ch;

    printf("\nEnter bucket size: ");
    scanf("%d", &cap);

    printf("\nEnter output rate: ");
    scanf("%d", &orate);

    do {
        printf("\n Enter number of packets coming at second %d: ", i + 1);
        scanf("%d", &inp[i]);
        i++;
        printf("\n Enter 1 to continue or 0 to quit:");
        scanf("%d", &ch);
    } while (ch);

    nsec = i;
    printf("\nSecond\tSent\tRecieved\tDropped\tRemained\n");

    for (i = 0; count >0 || i < nsec; i++) {
    printf("%d", i + 1);
    printf("\t%d\t", inp[i]);
    printf("\t%d\t", MIN((inp[i] + count),orate));
    if ((x = inp[i] + count - orate) > 0)
    {
        if (x > cap)
        {
            count = cap;
            drop = x - cap;
        }
        else
        {
            count = x;
            drop = 0;
        }
        } else
        {
        drop = 0;
        count = 0;
        }
        printf("\t%d\t%d\n", drop, count);
    }
    return 0;
}
```

## 04] CRC Error Control - 52 | 7 - 10 -  8 - 8 - 10 - 9 | 06

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

char t[30], cs[30], g[10];
int a, i, j, N;

void xor1()
{
        for (j = 1; j < N; j++)
        cs[j] = ((cs[j] == g[j]) ? '0' : '1');
}

void crc()
{
        for (i = 0; i < N; i++)
        cs[i] = t[i];
        do {
                if (cs[0] == '1')
                xor1();
                for (j = 0; j < N - 1; j++)
                cs[j] = cs[j + 1];
                cs[j] = t[i++];
        } while (i <= a + N - 1);
}

int main()
{
        printf("\n Enter data: ");
        scanf("%s", t);
        printf("\n-----------------------------------");
        printf("\n Enter the generating polynomial data: ");
        scanf("%s", g);
        N = strlen(g);
        a = strlen(t);

        if ((N - 1) < a && (g[0] == '1' && g[N - 1] == '1'))
        {
                for (i = a; i < a + N - 1; i++)
                t[i] = '0';
                t[i] = '\0';
                printf("\n------------------------------------");
                printf("\n Modified data is: %s", t);
                printf("\n-----------------------------------");
                crc();
                for (i = a; i < a + N - 1; i++)
                t[i] = cs[i - a];
                t[i] = '\0';
                printf("\n Checksum is: %s", cs);
                printf("\n------------------------------------");
                printf("\n Transmitting codeword is: %s", t);
                printf("\n------------------------------------");
                printf("\nEnter received message: ");
                scanf("%s", t);
                crc();
                for (i = 0; i < N - 1 && cs[i] != '1'; i++);
                        if (i < N - 1)
                                printf("\n Error detected\n\n");
                        else
                                printf("\n No error detected\n\n");
                                printf("\n------------------------------------\n");
        }
        else
        {
                printf("Wrong generating polynomial\n");
        }
        return 0;
}
```

## 05] Dijkstra's Alogrithm - 46 | 14 - 9 - 7 - 7 - 9 | 05

```
#include<stdio.h>
#define INFINITY 99
#define startnode 2

void dijkstra(int cost[10][10],int n);

int main()
{
        int cost[10][10],i,j,n,u;
        printf("enter the no. of vertices: ");
        scanf("%d",&n);
        printf("\n Enter the cost matrix:\n");
            for(i=0;i<n;i++)
             for(j=0;j<n;j++)
               scanf("%d",&cost[i][j]);
            dijkstra(cost,n);
            return 0;
}

void dijkstra(int cost[10][10], int n)
{
 int distance[10],pred[10],visited[10], count, mindistance, nextnode,i,j;
 for(i=0;i<n;i++)
 {
  distance[i]=cost[startnode][i];
  pred[i]=startnode;
  visited[i]=0;
 }
 distance[startnode]=0;
 visited[startnode]=1;
 count=1;
 while(count<n-1)
 {
  mindistance = INFINITY;
  for(i=0;i<n;i++)
    if(distance[i]<mindistance&&!visited[i])
    {
      mindistance = distance[i];
      nextnode = i;
    }
     visited[nextnode]=1;
     for(i=0;i<n;i++)
      if(!visited[i])
        if(mindistance+cost[nextnode][i]<distance[i])
          {
            distance[i]=mindistance+cost[nextnode][i];
            printf("%d.........%d\n",i,distance[i]);
            pred[i]=nextnode;
          }
         count++;
  }
  for(i=0;i<n;i++)
     if(i!=startnode)
     {
       printf("\n Distance to node %d=%d",i,distance[i]);
       printf("\n through the Path=%d",i);
       j=i;
       do
        {
         j=pred[j];
         printf("<_%d",j);
        }while(j!=startnode);
     }
}
```

## 06] DUPLEX LINKS [TCP-FTP]|[UDP-CBR] - 60 | 19 - 12 - 8 - 7 - 4 - 10 | 06

```
set val(stop) 10.0
set ns [new Simulator]

set tracefile [open exp1.tr w]
$ns trace-all $tracefile

set namfile [open exp1.nam w]
$ns namtrace-all $namfile

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n2 300.0Mb 10ms DropTail
$ns queue-limit $n0 $n2 10
$ns duplex-link $n1 $n2 400.0Mb 10ms DropTail
$ns queue-limit $n1 $n2 20
$ns duplex-link $n2 $n3 10.0Mb 10ms DropTail
$ns queue-limit $n2 $n3 3

$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink2 [new Agent/TCPSink]
$ns attach-agent $n3 $sink2
$ns connect $tcp0 $sink2
$tcp0 set packetSize_ 1500

set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp1 $sink3
$tcp1 set packetSize_ 1500

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 2.0 "$ftp0 stop"

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 1.0 "$ftp1 start"
$ns at 2.0 "$ftp1 stop"

proc finish {} {
global ns tracefile namfile
$ns flush-trace
close $tracefile
close $namfile
exec nam exp1.nam &
exit 0
}

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

```
BEGIN
{
            tcppack=0
            tcppack1=0
}
{
            if($1=="r"&&$4=="3"&&$5=="tcp"&&$6=="1540")
{
            tcppack++;
}
            if($1=="d"&&$3=="2"&&$4=="3"&&$5=="tcp"&&$6=="1540")
{
            tcppack1++;
}
}
END
{
            printf("\n total number of data packets received at Node 3: %d\n", tcppack++);
            printf("\n total number of packets dropped at Node 2: %d\n", tcppack1++);
}
```

## **07] TCP | UDP - 65**

```
set val(stop) 10.0
set ns [new Simulator]

set tracefile [open exp2.tr w]
$ns trace-all $tracefile

set namfile [open exp2.nam w]
$ns namtrace-all $namfile

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n2 200.0Mb 10ms DropTail
$ns queue-limit $n0 $n2 50
$ns duplex-link $n2 $n3 200.0Mb 10ms DropTail
$ns queue-limit $n2 $n3 50
$ns duplex-link $n1 $n2 200.0Mb 10ms DropTail
$ns queue-limit $n1 $n2 50

$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n1 $n2 orient right-up

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp0 $sink3
$tcp0 set packetSize_ 1000
$tcp0 set interval_ 0.1

set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set null2 [new Agent/Null]
$ns attach-agent $n3 $null2
$ns connect $udp1 $null2
$udp1 set packetSize_ 1100
$udp1 set interval_ 0.1
```

```
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 9.0 "$ftp0 stop"

set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 9.0 "$cbr1 stop"

proc finish {} {
global ns tracefile namfile
$ns flush-trace
close $tracefile
close $namfile
exec nam exp2.nam &
exit 0
}

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run

BEGIN{
tcppack=0
tcppack1=0
}
{
if($1=="r"&&$4=="2"&&$5=="tcp"&&$6=="40")
{
tcppack++;
}
if($1=="r"&&$4=="2"&&$5=="cbr"&&$6=="1000")
{
tcppack1++;
}
}
END{
printf("\n total number of TCP data packets sent between Node 0 and Node 2: %d\n", tcppack++);
printf("\n total number of UDP data packets sent between Node 1 and Node 2: %d\n", tcppack1++);
}
```

## 08] **Ethernet LAN - 59**

```
set ns [new Simulator]

set tf [open lab3.tr w]
$ns trace-all $tf

set nf [open lab3.nam w]
$ns namtrace-all $nf

$ns color 0 blue

set n0 [$ns node]
$n0 color "red"
set n1 [$ns node]
$n1 color "red"
set n2 [$ns node]
$n2 color "red"
```

```
set n3 [$ns node]
$n3 color "red"
set n4 [$ns node]
$n4 color "magenta"
set n5 [$ns node]
$n5 color "magenta"
set n6 [$ns node]
$n6 color "magenta"
set n7 [$ns node]
$n7 color "magenta"

$n1 label "Source/UDP"
$n3 label "Error Node"
$n7 label "Destination"

$ns make-lan "$n0 $n1 $n2 $n3" 100Mb 300ms LL Queue/DropTail Mac/802_3
$ns make-lan "$n4 $n5 $n6 $n7" 100Mb 300ms LL Queue/DropTail Mac/802_3

$ns duplex-link $n3 $n4 100Mb 300ms DropTail
$ns duplex-link-op $n3 $n4 color "green"

set err [new ErrorModel]
$ns lossmodel $err $n3 $n4
$err set rate_ 0.3

set udp [new Agent/UDP]
$ns attach-agent $n1 $udp

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp

$cbr set fid_ 0
$cbr set packetSize_ 1000
$cbr set interval_ 0.1

set null [new Agent/Null]
$ns attach-agent $n7 $null
$ns connect $udp $null

proc finish { } {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam lab3.nam &
exit 0
}

$ns at 0.1 "$cbr start"
$ns at 3.0 "finish"
$ns run

BEGIN{
tcppack=0
tcppack1=0
}
{
if($1=="r"&&$4=="7"&&$5=="cbr"&&$6=="1000")
{
tcppack++;
}
}
END{
printf("\n total number of data packets at Node 7: %d\n", tcppack++);
}
```

**09 ] <u>ESS Implementation - 91</u>**

```
set ns [new Simulator]

set tf [open expt55.tr w]
$ns trace-all $tf

set topo [new Topography]
$topo load_flatgrid 1000 1000
set nf [open expt55.nam w]
$ns namtrace-all-wireless $nf 2000 2000

set chan [new Channel/WirelessChannel];#Create wireless channel

$ns node-config -adhocRouting AODV \
   -llType LL \
   -macType Mac/802_11 \
   -ifqType Queue/DropTail \
   -ifqLen 50 \
   -phyType Phy/WirelessPhy \
   -channel $chan \
   -propType Propagation/TwoRayGround \
   -antType Antenna/OmniAntenna \
   -topoInstance $topo \
   -agentTrace ON \
   -routerTrace ON \
   -macTrace ON

create-god 6
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
#set n5 [$ns node]
set n6 [$ns node]
#set n7 [$ns node]

 $n0 label "tcp-Source"
$n1 label "Access Point1"
$n2 label "Router"
$n3 label "Access Point2"
$n4 label "Destination"
#$n5 label "node1"
$n6 label "node2"
#$n7 label "gateway"

$n0 set X_ 10
$n0 set Y_ 50
$n0 set Z_ 0
$ns initial_node_pos $n0 20

$n1 set X_ 120
$n1 set Y_ 130
$n1 set Z_ 0
$ns initial_node_pos $n1 20
$n2 set X_ 200
$n2 set Y_ 230
$n2 set Z_ 0
$ns initial_node_pos $n2 20
$n3 set X_ 300
$n3 set Y_ 130
$n3 set Z_ 0
$ns initial_node_pos $n3 20
```

```
$n4 set X_ 350
$n4 set Y_ 20
$n4 set Z_ 0
$ns initial_node_pos $n4 20

 $n6 set X_ 600
$n6 set Y_ 20
$n6 set Z_ 0
$ns initial_node_pos $n6 20

 $ns at 0.1 "$n0 setdest 50 50 15"
$ns at 0.1 "$n4 setdest 900 50 20"

 set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

set sink4 [new Agent/TCPSink]
$ns attach-agent $n4 $sink4
$ns connect $tcp0 $sink4

$ns at 5 "$ftp0 start"
$ns at 50 "$ftp0 stop"

proc finish { } {
 global ns nf tf
 $ns flush-trace
 exec nam expt55.nam  &
 close $tf
 exit 0
}

$ns at 80 "finish"
$ns run

BEGIN{
cbrpack=0
cbrpack1=0
}
{
if($1=="r"&&$4=="AGT")
{
cbrpack++;
}
if($1=="s"&&$4=="AGT")
{
cbrpack1++;
}
}
END{
printf("\n total number of packets sent: %d\n", cbrpack1++);
printf("\n total number of packets received: %d\n", cbrpack++);
}
```