Prog 1 :Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.

```
set ns [new Simulator]
set tracefile [open prog1.tr w]
$ns trace-all $tracefile
set namfile [open prog1.nam w]
$ns namtrace-all $namfile
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$ns duplex-link $n0 $n1 0.25Mb 10ms DropTail
$ns queue-limit $n0 $n1 5
$ns duplex-link $n1 $n2 100Mb 10ms DropTail
#$ns queue-limit $n1 $n2 3
$ns duplex-link-op $n0 $n1 orient right-down
$ns duplex-link-op $n1 $n2 orient left-down
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n2 $sink
$ns connect $tcp $sink
$tcp set PacketSize_ 2500
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.5 "$ftp start"
$ns at 2.0 "$ftp stop"
$ns at 2.5 "Finish"
proc Finish {} {
global ns tracefile namfile
$ns flush-trace
close $tracefile
close $namfile
exec nam prog1.nam &
exec awk -f prog1.awk prog1.tr &
exit 0
}
puts "simulation starts..."
$ns run
```

prog1.awk

```
BEGIN{
count=0;
}
{
event=$1;
if(event == "d")
{
count++;
}
}
END{
printf("\nNumber of packets dropped is %d\n",count);
}
```

prog 2: Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

```
set ns [new Simulator]
set tracefile [open prog2.tr w]
$ns trace-all $tracefile
set namfile [open prog2.nam w]
$ns namtrace-all $namfile
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$ns color 1 red
$ns color 2 blue
$ns duplex-link $n0 $n1 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n1 4
$ns duplex-link $n1 $n2 50.0Mb 10ms DropTail
$ns queue-limit $n1 $n2 4
$ns duplex-link $n2 $n3 1.0Mb 10ms DropTail
$ns queue-limit $n2 $n3 5
$ns duplex-link $n3 $n4 1.0Mb 10ms DropTail
$ns duplex-link $n4 $n5 10.0Mb 10ms DropTail
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient right-down
$ns duplex-link-op $n3 $n4 orient left
$ns duplex-link-op $n4 $n5 orient left
Agent/Ping instproc recv { from rtt } {
$self instvar node_
puts "node [$node_ id] recived ping answer from \#$from with round-trip-time $rtt ms"
}
set p0 [new Agent/Ping]
$ns attach-agent $n0 $p0
$p0 set packetSize_ 50000
$p0 set fid_ 1
set p5 [new Agent/Ping]
$ns attach-agent $n5 $p5
$p5 set packetSize_ 50000
$p5 set fid_ 2
$ns connect $p0 $p5
$ns at 0.1 "$p0 send"
$ns at 0.2 "$p0 send"
$ns at 0.3 "$p0 send"
$ns at 0.4 "$p0 send"
$ns at 0.5 "$p0 send"
$ns at 0.6 "$p0 send"
$ns at 0.7 "$p0 send"
$ns at 0.9 "$p0 send"
$ns at 1.0 "$p0 send"
```

```
$ns at 0.1 "$p5 send"
$ns at 0.2 "$p5 send"
$ns at 0.3 "$p5 send"
$ns at 0.4 "$p5 send"
$ns at 0.5 "$p5 send"
$ns at 0.6 "$p5 send"
$ns at 0.7 "$p5 send"
$ns at 0.9 "$p5 send"
$ns at 1.0 "$p5 send"

proc finish { } {
global ns tracefile namfile
$ns flush-trace
close $tracefile
close $namfile
exec nam prog2.nam &
exit 0
}
$ns at 10.0 "finish"
$ns run
```

prog2.awk

```
BEGIN{
count=0;
}
{
event=$1;
if(event == "d")
{
count++;
}
}
END{
printf("\nNumber of packets dropped is %d\n",count);
}
```

prog 3 : Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.

```
set ns [new Simulator]
set nf [open prog3.nam w]
$ns namtrace-all $nf

proc finish {} {
global ns nf
$ns flush-trace
close $nf
exec nam prog3.nam &
exit 0
}

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

$ns duplex-link $n0 $n1 1Mb 5ms DropTail

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set PacketSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n1 $null0

$ns connect $udp0 $null0
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0  stop"
$ns at 5.0 "finish"

$ns run
```

prog3.awk

```
BEGIN{
count=0;
}
{
event=$1;
if(event == "d")
{
count++;
}
}
END{
printf("\nNumber of packets dropped is %d\n",count);
}
```

prog 4: Implement a network with four nodes to simulate the working of TCP and UDP protocols.

```
set ns [new Simulator]

$ns color 1 Blue
$ns color 2 Red

set nf [open prog4.nam w]
$ns namtrace-all $nf

proc finish {} {
global ns nf
$ns flush-trace

close $nf

exec nam prog4.nam &
exit 0
}

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

$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

$ns queue-limit $n2 $n3 10
$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

$ns duplex-link-op $n2 $n3 queuePos 0.5

set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

set udp [new Agent/UDP]
```

prog4.awk

```
BEGIN{
count=0;
}
{
event=$1;
if(event == "d")
{
count++;
}
}
END{
printf("\nNumber of packets dropped is %d\n",count);
}
```

```
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 2000
$cbr set rate_ 1mb
$cbr set random_ false

$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

$ns at 5.0 "finish"

puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

$ns run
```

prog 5: Implement a network with eight nodes to simulate the working of TCP and UDP protocols.

```
set ns [new Simulator]

$ns color 1 Green
$ns color 2 Blue

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

proc finish {} {
global ns nf
$ns flush-trace

close $nf

exec nam out.nam &
exit 0}

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]

$ns duplex-link $n0 $n2 0.5Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail
$ns duplex-link $n3 $n4 2Mb 10ms DropTail


$ns duplex-link $n4 $n5 1Mb 30ms DropTail
$ns duplex-link $n4 $n6 1.7Mb 10ms DropTail
$ns duplex-link $n5 $n7 2.2Mb 20ms DropTail
$ns duplex-link $n6 $n7 2.7Mb 10ms DropTail

$ns queue-limit $n2 $n3 10
$ns queue-limit $n4 $n5 10
$ns queue-limit $n6 $n7 20

$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-down
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n4 $n5 orient right-up
$ns duplex-link-op $n4 $n6 orient right-down
```

prog5.awk

```
BEGIN{
count=0;
}
{
event=$1;
if(event == "d")
{
count++;
}
}
END{
printf("\nNumber of packets dropped is %d\n",count);
}
```

```
$ns duplex-link-op $n5 $n7 orient right-down
$ns duplex-link-op $n6 $n7 orient right-up

$ns duplex-link-op $n2 $n3 queuePos 0.5
$ns duplex-link-op $n4 $n5 queuePos 0.4
$ns duplex-link-op $n6 $n7 queuePos 0.2
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n6 $null
$ns connect $udp $null
$udp set fid_ 2

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false

$ns at 0.1 "$cbr start"
$ns at 0.5 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n5 $sink"
$ns at 5.0 "finish"

puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

$ns run
```

prog 6: Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and plot ongestion window for different source / destination.

```
set ns [new Simulator]
set namfile [open prog6.nam w]
$ns namtrace-all $namfile
set tracefile [open prog6.tr w]
$ns trace-all $tracefile
proc finish {} {
global ns namfile tracefile
$ns flush-trace
close $namfile
close $tracefile
exec nam prog6.nam &
exit 0
}
set n0 [$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 [$ns node]
set n8 [$ns node]
$ns color 1 Blue
$ns color 2 Red
$n7 shape box
$n7 color Blue
$n8 shape hexagon
$n8 color Red
$ns duplex-link $n1 $n0 2Mb 10ms DropTail
$ns duplex-link $n2 $n0 2Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 20ms DropTail
$ns make-lan "$n3 $n4 $n5 $n6 $n7 $n8" 512Kb 40ms LL Queue/DropTail Mac/802_3
$ns duplex-link-op $n1 $n0 orient right-down
$ns duplex-link-op $n2 $n0 orient right-up
$ns duplex-link-op $n0 $n3 orient right
$ns queue-limit $n0 $n3 20
set tcp1 [new Agent/TCP/Vegas]
$ns attach-agent $n1 $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n7 $sink1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns connect $tcp1 $sink1
$tcp1 set class_ 1
$tcp1 set packetSize_ 55
set tfile1 [open cwnd1.tr w]
$tcp1 attach $tfile1
$tcp1 trace cwnd_
set tcp2 [new Agent/TCP/Reno]
```

prog6.awk

```
BEGIN{
}
{
if($6==”cwnd_” )
{
printf("\n%f\t %f\n",$1,$7);
}
}
END{
}
```

```
$ns attach-agent $n2 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $n8 $sink2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ns connect $tcp2 $sink2
$tcp2 set class_ 2
$tcp2 set packetSize_ 55
set tfile2 [open cwnd2.tr w]
$tcp2 attach $tfile2
$tcp2 trace cwnd_
$ns at 0.5 "$ftp1 start"
$ns at 1.0 "$ftp2 start"
$ns at 5.0 "$ftp2 stop"
$ns at 5.0 "$ftp1 stop"
$ns at 5.5 "finish"
$ns run
```

Prog 7: Simulate a simple ESS and with transmitting nodes in wireless LAN by simulation and determine the performance with respect to transmission of packets.

AWK :

```
BEGIN{
PacketRcvd = 0;
Throughput = 0.0;
}
{
if(($1=="r")&&($3=="_3_")&&($4=="AGT")&&($7=="tcp")&&($8>1000))
{
PacketRcvd++;
}
}
END {
Throughput=((PacketRcvd*1000*8)/(95.0*1000000));
printf("\nThe throughput is:%f\n",Throughput);
}
```

TCL :

```
if {$argc != 1} {
error "Command: ns <ScriptName.tcl><Number_of_Nodes>"
exit 0
}
set ns [new Simulator]
set tracefile [open prog7.tr w]
$ns trace-all $tracefile
set namfile [open prog7.nam w]
$ns namtrace-all-wireless $namfile 750 750
```

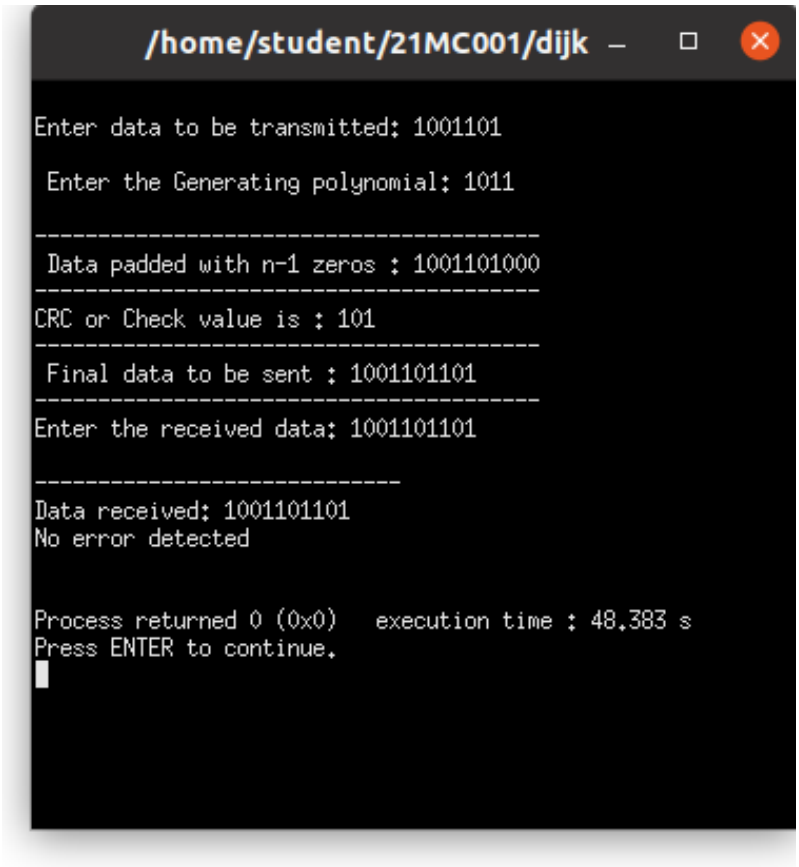
```
proc finish {} {
global ns tracefile namfile
$ns flush-trace
close $tracefile
close $namfile
exec nam prog7.nam &
exec awk -f prog7.awk prog7.tr &
exit 0
}

set val(nn) [lindex $argv 0]

set topo [new Topography]
$topo load_flatgrid 750 750

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

set god_ [create-god $val(nn)]

for {set i 0} {$i < $val(nn)} {incr i} {
set n($i) [$ns node]
}

$n(1) label "TCPSource"
$n(3) label "Sink"

for {set i 0} {$i < $val(nn)} {incr i} {
set XX [expr rand()*750]
set YY [expr rand()*750]
$n($i) set X_ $XX
$n($i) set Y_ $YY
}

for {set i 0} {$i < $val(nn)} {incr i} {
$ns initial_node_pos $n($i) 100
}

proc destination {} {
global ns val n
set now [$ns now]
```

```
set time 5.0
for {set i 0} {$i< $val(nn)} {incr i} {
set XX [expr rand()*750]
set YY [expr rand()*750]
$ns at [expr $now + $time] "$n($i) setdest $XX $YY 20.0"
}
$ns at [expr $now + $time] "destination"
}
set tcp [new Agent/TCP]
$ns attach-agent $n(1) $tcp
set ftp [new Application/FTP]
$ftp attach-agent $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n(3) $sink
$ns connect $tcp $sink
$ns at 0.0 "destination"
$ns at 1.0 "$ftp start"
$ns at 100 "finish"
$ns run
```

PROG 8 : Program to implement CRC

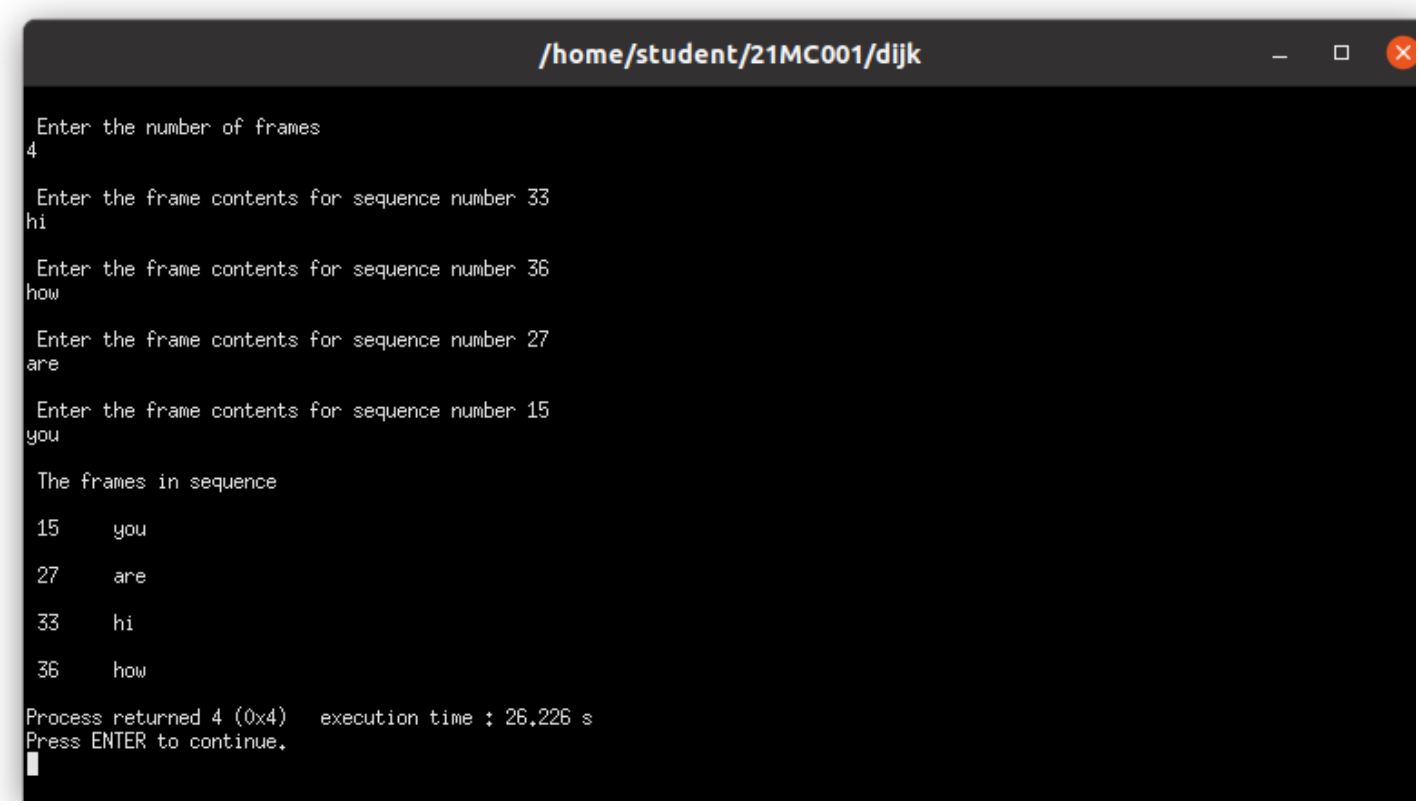
```
#include<stdio.h>
#include<string.h>
#define N strlen(gen_poly)
char data[28];
char check_value[28];
char gen_poly[10];
int data_length,i,j;
void XOR(){
   for(j = 1;j < N; j++)
   check_value[j] = (( check_value[j] == gen_poly[j])?'0':'1');
}
void receiver(){
   printf("Enter the received data: ");
   scanf("%s", data);
   printf("\n-----------------------------\n");
   printf("Data received: %s", data);
   crc();
   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(){
   for(i=0;i<N;i++)
      check_value[i]=data[i];
   do{
      if(check_value[0]=='1')
         XOR();
      for(j=0;j<N-1;j++)
         check_value[j]=check_value[j+1];
      check_value[j]=data[i++];
   }while(i<=data_length+N-1);
}

int main()
{
   printf("\nEnter data to be transmitted: ");
   scanf("%s",data);
   printf("\n Enter the Generating polynomial: ");
   scanf("%s",gen_poly);
   data_length=strlen(data);
   for(i=data_length;i<data_length+N-1;i++)
      data[i]='0';
   printf("\n----------------------------------------");
   printf("\n Data padded with n-1 zeros : %s",data);
   printf("\n----------------------------------------");
   crc();
   printf("\nCRC or Check value is : %s",check_value);
   for(i=data_length;i<data_length+N-1;i++)
```

```
    data[i]=check_value[i-data_length];
  printf("\n----------------------------------------");
  printf("\n Final data to be sent : %s",data);
  printf("\n----------------------------------------\n");

  receiver();
    return 0;
}
```



PROG 9 : Program to implement frameSorting

```
#include<stdio.h>
struct frame
{
int fslno;
char finfo[20];
};
struct frame arr[20];
int n;
void sort()
{
 int i,j,ex;
 struct frame temp;
 for(i=0;i<n;i++)
 {
 ex=0;
 for(j=0;j<n-i-1;j++)
 if(arr[j].fslno>arr[j+1].fslno)
 {
 temp=arr[j];
 arr[j]=arr[j+1];
 arr[j+1]=temp;
 ex++;
 }
if(ex==0)
break;
 }
}
void main()
{
 int i;
 printf("\n Enter the number of frames \n");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
 arr[i].fslno=rand()%50;
 printf("\n Enter the frame contents for sequence number %d",arr[i].fslno);
 scanf("%s",arr[i].finfo);
 }
 sort();
 printf("\n The frames in sequence \n");
 for(i=0;i<n;i++)
 printf("\n %d\t%s \n",arr[i].fslno,arr[i].finfo);
}
```



PROG 10 : program to implement Distance Vecotr Algorithm

```
#include<stdio.h>
struct node
{
   unsigned dist[20];
   unsigned from[20];
}rt[10];
int main()
{
   int costmat[20][20];
   int nodes,i,j,k,count=0;
   printf("\nEnter the number of nodes : ");
   scanf("%d",&nodes);
   printf("\nEnter the cost matrix :\n");
   for(i=0;i<nodes;i++)
   {
      for(j=0;j<nodes;j++)
      {
         scanf("%d",&costmat[i][j]);
         costmat[i][i]=0;
         rt[i].dist[j]=costmat[i][j];
         rt[i].from[j]=j;
      }
   }
```

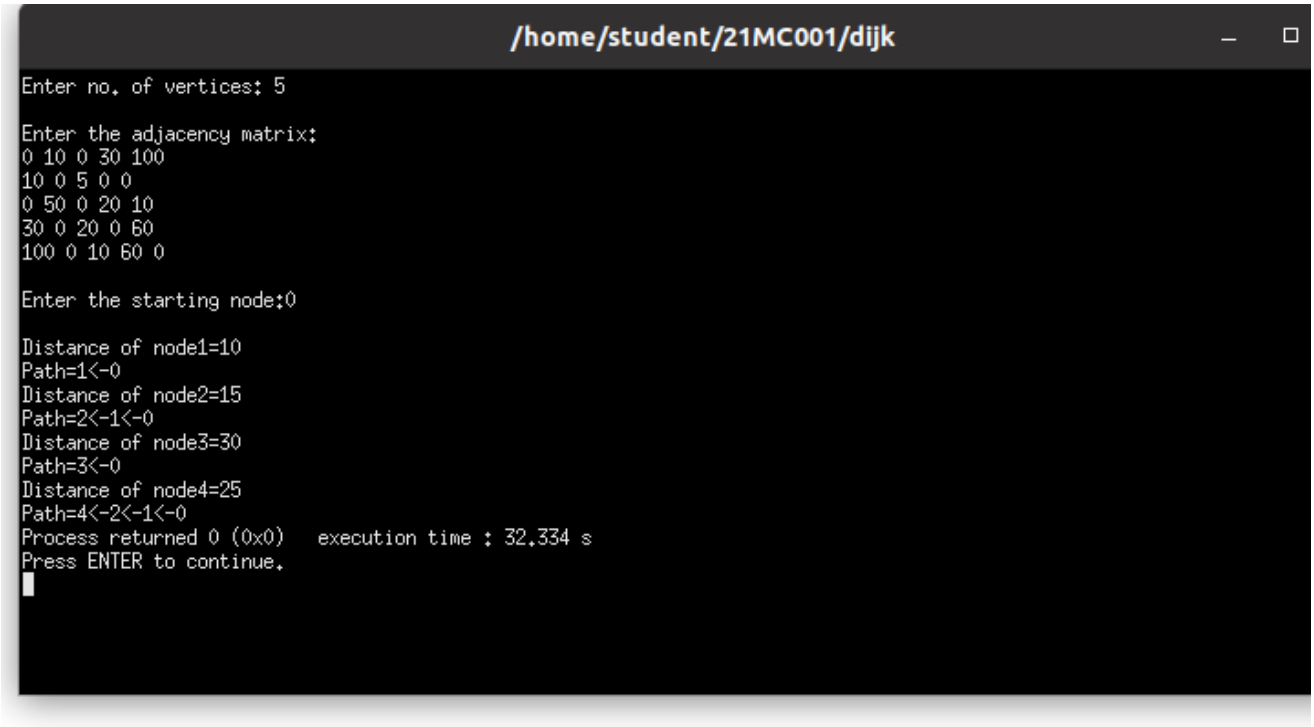
```
        do
        {
           count=0;
           for(i=0;i<nodes;i++)

           for(j=0;j<nodes;j++)
           for(k=0;k<nodes;k++)
              if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
              {
                 rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
                 rt[i].from[j]=k;
                 count++;
              }
        }while(count!=0);
        for(i=0;i<nodes;i++)
        {
           printf("\n\n For router %d\n",i+1);
           for(j=0;j<nodes;j++)
           {
              printf("\t\nnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
           }
        }
     printf("\n\n");
  }
```

```
/home/student/21MC001/dijk

Enter the number of nodes :  4

Enter the cost matrix :
0 2 9999 1
2 0 3 7
9999 3 0 11
1 7 11 0


 For router 1

node 1 via 1 Distance 0
node 2 via 2 Distance 2
node 3 via 2 Distance 5
node 4 via 4 Distance 1

 For router 2

node 1 via 1 Distance 2
node 2 via 2 Distance 0
node 3 via 3 Distance 3
node 4 via 1 Distance 3

 For router 3

node 1 via 2 Distance 5
node 2 via 2 Distance 3
node 3 via 3 Distance 0
node 4 via 2 Distance 6

 For router 4

node 1 via 1 Distance 1
node 2 via 1 Distance 3
node 3 via 1 Distance 6
node 4 via 4 Distance 0


Process returned 0 (0x0)   execution time : 37.214 s
Press ENTER to continue.
```

Prog 11 : Program to Implement Dijkastras Algorithm

```
#include<stdio.h>
#define INFINITY 9999
#define MAX 10
void dijkstra(int G[MAX][MAX],int n,int startnode);
int main()
{
int G[MAX][MAX],i,j,n,u;
printf("Enter no. of vertices:");
scanf("%d",&n);
printf("\nEnter the adjacency matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
scanf("%d",&G[i][j]);
printf("\nEnter the starting node:");
scanf("%d",&u);
dijkstra(G,n,u);
return 0;
}
void dijkstra(int G[MAX][MAX],int n,int startnode)
{
int cost[MAX][MAX],distance[MAX],pred[MAX];
int visited[MAX],count,mindistance,nextnode,i,j;
for(i=0;i<n;i++)
for(j=0;j<n;j++)
if(G[i][j]==0)
cost[i][j]=INFINITY;
else
cost[i][j]=G[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];
pred[i]=nextnode;
}
count++;
}
for(i=0;i<n;i++)
if(i!=startnode)
{
printf("\nDistance of node%d=%d",i,distance[i]);
printf("\nPath=%d",i);
j=i;
do
{
j=pred[j];
printf("<-%d",j);
}while(j!=startnode);
}
}
```



PROG 12 : program to implement Hamming Distance btw two strings

```
#include <stdio.h>
int hammingDist(char* str1, char* str2)
{
int i = 0, count = 0;
while (str1[i] != '\0') {
if (str1[i] != str2[i])
count++;
i++;
}
return count;
}
int main()
{
char str1[50], str2[50];
printf("Enter the first string:");
scanf("%s", str1);
printf("Enter the second string:");
scanf("%s", str2);
printf("%d", hammingDist(str1, str2));
return 0;
}
```