Assignment 1

1. implement frame sorting
-> each frame consists sequence number , time , message
->Sequence number must be a random number of 2 digits , regenerate the repeated sequence number
->sort the frames based on sequence number using bubble sort technique
->Display all the info of the packet before sort and after sort

Code:

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

struct packet
{
    int seqno;
    float time;
    char msg[100];
} p[100], temp;

void bubble_sort(int n, struct packet p[])
{
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - 1 - i; j++)
        {
            if (p[j].seqno > p[j + 1].seqno)
            {
                temp = p[j];
                p[j] = p[j + 1];
                p[j + 1] = temp;
            }
        }
    }
}

void display(int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("%d\t%f\t%s\n", p[i].seqno, p[i].time, p[i].msg);
    }
```

```
}

int randum()
{
    return rand() % 100;
}

int main()
{
    int n, num;
    printf("Enter the number of Frames:\n");
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        num = randum();
        for (int j = 0; j < n; j++)
        {
            if (p[j].seqno == num)
            {
                num = randum();
            }
        }
        p[i].seqno = num;
        printf("Enter the Message:\n");
        scanf("%s", p[i].msg);
        p[i].time = rand();
    }
    printf("Array Before sorting:\n");
    display(n);
    bubble_sort(n, p);
    printf("\nArray After sorting:\n");
    display(n);
    return 0;
}
```

Output:

```
Enter the number of Frames:
3
Enter the Message:
asd
```

```
Enter the Message:
aaa
Enter the Message:
bbb
Array Before sorting:
83      846930880.000000    asd
77      1714636928.000000   aaa
93      424238336.000000    bbb

Array After sorting:
77      1714636928.000000   aaa
83      846930880.000000    asd
93      424238336.000000    bbb
```

2)program to implement frame sorting.Each frame contains sequence number,time and message.Sequence number must be a random number of 4 digits.Sort the frames based on sequence number using quick sort technique

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

struct packet
{
   int seqno;
   float time;
   char msg[100];
} p[100], temp;


void quick_sort(struct packet p[],int l,int r)
{
   if (l < r)
   {
      int s = partition(p, l, r);
      quick_sort(p, l, s - 1);
      quick_sort(p, s + 1, r);
   }
}

int partition(struct packet p[], int l, int r)
```

```
{
    int x = p[r].seqno;
    int i = l - 1;

    for (int j = l; j < r; j++)
    {
        if (p[j].seqno  <= x)
        {
            i++;

            temp = p[i];
            p[i] = p[j];
            p[j] = temp;
        }
    }

    temp = p[i + 1];
    p[i + 1] = p[r];
    p[r] = temp;

    return i + 1;
}

void display(int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("%d\t%f\t%s\n", p[i].seqno, p[i].time, p[i].msg);
    }
}

int randum()
{
    return rand() % 10000;
}

int main()
{
    int n, num;
    printf("Enter the number of Frames:\n");
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        num = randum();
```

```
        for (int j = 0; j < n; j++)
        {
            if (p[j].seqno == num)
            {
                num = randum();
            }
        }
        p[i].seqno = num;
        printf("Enter the Message:\n");
        scanf("%s", p[i].msg);
        p[i].time = rand();
    }
    printf("Array Before sorting:\n");
    display(n);
    quick_sort(p,0,n-1);
    printf("\nArray After sorting:\n");
    display(n);
    return 0;
}
```

Output:

```
Enter the number of Frames:
4
Enter the Message:
abc
Enter the Message:
cde
Enter the Message:
feg
Enter the Message:
kjh
Array Before sorting:
9383    846930880.000000    abc
2777    1714636928.000000   cde
7793    424238336.000000    feg
5386    1649760512.000000   kjh

Array After sorting:
2777    1714636928.000000   cde
5386    1649760512.000000   kjh
```

7793 424238336.000000 feg
9383 846930880.000000 abc

3) program to implement frame sorting.Each frame contains sequence number,packet id , source ip address , destination ip address,port number and message.
->Sequence number must be a random number of 3 digits regenerate the repeated sequence number
->Sort the frames based on sequence number using insertion

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

struct packet
{
    int seqno;
    int packet_id;
    int source_ip;
    int destination_ip;
    int port_no;
    char msg[100];
} p[100], temp;

void insertion_sort(int n,struct packet p[])
{   int j;
    for(int i=1;i<n;i++){
        temp=p[i];
        for(j=i-1;j>=0 && temp.seqno<p[j].seqno;j--){
                p[j+1]=p[j];
        }
        p[j+1]=temp;
    }
}

void display(int n)
{
    for (int i = 0; i < n; i++)
    {
```

```
        printf("%d\t%d\t%d\t%d\t%d\t%s\n", p[i].seqno,p[i].packet_id,p[i].source_ip,
p[i].destination_ip,p[i].port_no,p[i].msg);
    }
}


int randum()
{
    return rand() % 1000;
}

int main()
{
    int n, num;
    printf("Enter the number of Frames:\n");
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        num = randum();
        for (int j = 0; j < n; j++)
        {
            if (p[j].seqno == num)
            {
                num = randum();
            }
        }
        p[i].seqno = num;
        printf("Enter the Message:\n");
        scanf("%s", p[i].msg);
        printf("Enter the Source Ip Address:\n");
        scanf("%d", &p[i].source_ip);
        printf("Enter the Destination Ip Address:\n");
        scanf("%d", &p[i].destination_ip);
        printf("Enter the Port Number:\n");
        scanf("%d", &p[i].port_no);
        p[i].packet_id = rand();
    }
    printf("Array Before sorting:\n");
    display(n);
    insertion_sort(n, p);
    printf("\nArray After sorting:\n");
    display(n);
    return 0;
}
```

Output:

```
Enter the number of Frames:
3
Enter the Message:
hello
Enter the Source Ip Address:
123
Enter the Destination Ip Address:
456
Enter the Port Number:
11

Enter the Message:
World
Enter the Source Ip Address:
999
Enter the Destination Ip Address:
888
Enter the Port Number:
10

Enter the Message:
 Welcome
Enter the Source Ip Address:
333
Enter the Destination Ip Address:
444
Enter the Port Number:
5


Array Before sorting:
383     846930886     123     456     11      hello
777     1714636915    999     888     10      World
793     424238335     333     444     5       Welcome

Array After sorting:
383     846930886     123     456     11      hello
777     1714636915    999     888     10      World
```

793 424238335 333 444 5 Welcome

NETWORK ASSIGNMENTS : 2 - NS2

1. Simulate a four node point-to-point network, and connect the links as follows: n0-n2, n1-n2, n2-n3 and n1-n3.

Apply UDP traffic between n0-n3.

Apply relevant applications over UDP agents changing the parameter.

Set the queue size vary the bandwidth and find the number of packets dropped by UDP.

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

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

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

$ns duplex-link $n0 $n2 100Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 5ms DropTail
$ns duplex-link $n2 $n3 1Mb 5ms DropTail
$ns duplex-link $n1 $n3 1Mb 5ms DropTail
```

```
$ns queue-limit $n0 $n2 50
$ns queue-limit $n1 $n2 50
$ns queue-limit $n2 $n3 50
$ns queue-limit $n1 $n3 50

set udp1 [new Agent/UDP]
$ns attach-agent $n0 $udp1
set null [new Agent/Null]
$ns attach-agent $n3 $null

$ns connect $udp1 $null
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packet-size 200
$cbr1 attach-agent $udp1

$ns at 0.5 "$cbr1 start"
$ns at 5.0 "$cbr1 stop"
$ns at 5.5 "finish"
$ns run
```

2. Simulate a six node point-to-point network, and connect the links as follows: n1-n5, n2-n3, n2-n4, n1-n6 and n3-n6.

Apply FTP agent between n3-n5 and TELNET between n4-n6.

Apply relevant applications over TCP agents changing the parameter.

Set the queue size vary the bandwidth and find the number of packets dropped by TCP.

Change node color, Link Color, Packet Color, change node position and shape.

```
set ns [new Simulator]
$ns color 1 violet
set nf [open out.nam w]
$ns namtrace-all $nf
set tf [open out.tr w]
```

```
$ns trace-all $tf

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

set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

$ns duplex-link $n1 $n5 0.011Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 5ms DropTail
$ns duplex-link $n2 $n4 1Mb 5ms DropTail
$ns duplex-link $n1 $n6 1Mb 5ms DropTail
$ns duplex-link $n3 $n6 1Mb 5ms DropTail

$ns duplex-link-op $n1 $n5 color red
$ns duplex-link-op $n2 $n3 color blue

$ns duplex-link-op $n1 $n5 orient left
$ns duplex-link-op $n2 $n3 orient right

$ns queue-limit $n1 $n5 50
$ns queue-limit $n2 $n3 0.011
$ns queue-limit $n2 $n4 50
$ns queue-limit $n1 $n6 50
$ns queue-limit $n3 $n6 50

$n1 color blue
$n2 color red

$n1 shape circle
$n2 shape box

set tcp1 [new Agent/TCP]
$ns attach-agent $n3 $tcp1
```

```
$tcp1 set fid_ 1
set tcpsink0 [new Agent/TCPSink]
$ns  attach-agent $n5 $tcpsink0
$ns connect $tcp1 $tcpsink0

set tcp2 [new Agent/TCP]
$ns attach-agent $n4 $tcp2
set tcpsink1 [new Agent/TCPSink]
$ns attach-agent $n6 $tcpsink1
$ns connect $tcp2 $tcpsink1

set ftp0 [new Application/FTP]
set telnet0 [new Application/Telnet]

$ftp0 set packet-size 200
$ftp0 set Interval 0.005
$ftp0 attach-agent $tcp1
$ns at 0.5 "$ftp0 start"
$ns at 4.5 "$ftp0 stop"

$telnet0 set packet-size 200
$telnet0 set interval 0.005
$telnet0 attach-agent $tcp2
$ns at 0.5 "$telnet0 start"
$ns at 4.5 "$telnet0 stop"

$ns at 5.5 "finish"
$ns run
```

C code:

```
#include<stdio.h>

void main(){
        FILE *fp = fopen("/home/adminmca/out.tr", "r");
        char x = 'd';
        char c;
        int count = 0;
        if(!fp){
                printf("can't open the file");
        }
```

```
        while((c=getc(fp)) != EOF){
                if(c == 'd'){
                        count++;
                }
        }

        printf("Packet dropped %d", count);
        fclose(fp);
}
```

Output:

Packet dropped 2

NETWORK ASSIGNMENTS : 3

1. **Simulate an Ethernet LAN with 10 nodes consisting of TCP traffic, Telnet traffic generator.**

Code:

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

proc finish {} {
global ns nf tf
$ns flush-trace
close $nf
close $tf
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 [$ns node]
set n8 [$ns node]
set n9 [$ns node]

$ns make-lan "$n0 $n1 $n2 $n3 $n4 $n5 $n6 $n7 $n8 $n9" 0.1Mb 0.1ms LL Queue/DropTail Mac/802_3

set tcp0 [new Agent/TCP]
$ns attach-agent $n3 $tcp0
$tcp0 set fid_ 1
set tcpsink0 [new Agent/TCPSink]
$ns attach-agent $n5 $tcpsink0
$ns connect $tcp0 $tcpsink0

set telnet0 [new Application/Telnet]
$telnet0 set packet-size 500
$telnet0 set interval 0.005
$telnet0 attach-agent $tcp0
$ns at 0.5 "$telnet0 start"
$ns at 2.0 "$telnet0 stop"

set tcp1 [new Agent/TCP]
$ns attach-agent $n3 $tcp1
$tcp1 set fid_ 2
set tcpsink0 [new Agent/TCPSink]
$ns attach-agent $n5 $tcpsink0
$ns connect $tcp1 $tcpsink0

set telnet1 [new Application/Telnet]
$telnet1 set packet-size 500
$telnet1 set interval 0.005
$telnet1 attach-agent $tcp1
$ns at 0.5 "$telnet1 start"
$ns at 2.0 "$telnet1 stop"
```

```
$ns at 5.5 "finish"
$ns run
```

2. **Simulate an Ethernet LAN with 12 nodes consisting of multiple traffic(TCP and UDP).**
   Apply different path for multiple traffic
   Apply CBR,FTP,TELNET application
   Find the number of packets dropped.
   change the shape and color of every source and destination node.
   Change the color of packet for multiple Traffic

Code:

```
set ns [new Simulator]
$ns color 1 brown
$ns color 2 green
set nf [open out.nam w]
$ns namtrace-all $nf
set tf [open out.tr w]
$ns trace-all $tf

proc finish {} {
global ns nf tf
$ns flush-trace
close $nf
close $tf
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]
set n8 [$ns node]
set n9 [$ns node]
```

```
set n10 [$ns node]
set n11 [$ns node]

$ns make-lan "$n0 $n1 $n2 $n3 $n4 $n5 $n6 $n7 $n8 $n9 $n10 $n11"  0.22Mb 10ms LL
Queue/DropTail Mac/802_3

$n0 color red
$n1 color red
$n2 color red
$n3 color red
$n4 color blue
$n5 color blue
$n6 color blue
$n7 color blue
$n8 color blue
$n9 color purple
$n10 color purple
$n11 color purple


$n0 shape box
$n2 shape box

set udp1 [new Agent/UDP]
$ns attach-agent $n0 $udp1
set null1 [new Agent/Null]
$ns attach-agent $n3 $null1
$ns connect $udp1 $null1

set tcp1 [new Agent/TCP]
$ns attach-agent $n4 $tcp1
$tcp1 set fid_ 1
set tcpsink0 [new Agent/TCPSink]
$ns attach-agent $n8 $tcpsink0
$ns connect $tcp1 $tcpsink0

set tcp2 [new Agent/TCP]
$ns attach-agent $n9 $tcp2
$tcp2 set fid_ 2
set tcpsink1 [new Agent/TCPSink]
$ns attach-agent $n11 $tcpsink1
$ns connect $tcp2 $tcpsink1

set cbr1 [new Application/Traffic/CBR]
```

```
$cbr1 set packet-size 200
$cbr1 set interval 0.005
$cbr1 attach-agent $udp1
$ns at 0.5 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"

set ftp0 [new Application/FTP]
$ftp0 set packet-size 200
$ftp0 set interval 0.005
$ftp0 attach-agent $tcp1
$ns at 0.5 "$ftp0 start"
$ns at 4.0 "$ftp0 stop"

set telnet0 [new Application/Telnet]
$telnet0 set packet-size 200
$telnet0 set interval 0.005
$telnet0 attach-agent $tcp2
$ns at 0.5 "$telnet0 start"
$ns at 4.5 "$telnet0 stop"

$ns at 5.0  "finish"
$ns run
```

NETWORK ASSIGNMENTS 4 – Error Detection Code

1. **Write a program for error detection code using Parity check technique.**

**************SENDER**************

Enter Message : 1011
1. ODD parity (0)
2. even parity (1)

Enter a option : 1
Message to be Transmitted : 1011**0**

Do You Want to Introduce error(Y/N) : Y
Enter the Position : 1 (condition : position between 1 to T size)

**************RECEIVER**************

Message received at the Receiver : **0**0110

ERROR in MESSAGE

Code:

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

void main()
{
   char msg[20];
   int option, position,count=0,i;

   printf("***** Sender*****");
   printf("\nEnter Message: ");
   scanf("%s", msg);

   printf("1. ODD parity (0)\n2. Even parity (1)\n");
   printf("Enter an option: ");
   scanf("%d", &option);
     int pos = strlen(msg);
   if(option==1)
   msg[pos] = '0';
   else if(option==2)
   msg[pos] = '1';
   printf("Message to be Transmitted: %s ", msg);

   printf("\nDo You Want to Introduce error (Y/N): ");
   char choice;
   scanf(" %c", &choice);

   if (choice == 'Y'|| choice == 'y') {
      printf("Enter the Position (condition: position between 1 to %d size): ", pos);
      scanf("%d", &position);


      if (position > 0 && position < pos) {
         if (msg[position-1] == '0') {
            msg[position-1] = '1';
         }
         else {
            msg[position-1] = '0';
         }
      }
   }
```

```
   printf("\n************RECEIVER************\n");
   printf("Message received at the Receiver: %s\n", msg);

      for(i=0;i<pos;i++)
                  {
                  if(msg[i]=='1')
                  count++;
                  }
          if(msg[pos]=='0')
          {
                  if(count%2!=0)
                  printf("\nMESSAGE WITHOUT ERROR");
            else
            printf("\nERROR IN MESSAGE");
          }
          if(msg[pos]=='1')
          {
                  if(count%2==0)
                  printf("\nMESSAGE WITHOUT ERROR");
            else
            printf("\nERROR IN MESSAGE");
          }
}
```

Output:

```
Enter Message: 1011
1. ODD parity (0)
2. Even parity (1)
Enter an option: 1
Message to be Transmitted: 10110
Do You Want to Introduce error (Y/N): Y
Enter the Position (condition: position between 1 to 4 size): 1
```

**********RECEIVER**********
Message received at the Receiver: 00110

ERROR IN MESSAGE

Output 2:

***** Sender*****
Enter Message: 1110
1. ODD parity (0)
2. Even parity (1)
Enter an option: 1
Message to be Transmitted: 11100
Do You Want to Introduce error (Y/N): N

**********RECEIVER**********
Message received at the Receiver: 11100

MESSAGE WITHOUT ERROR

2. **Write a program for error detecting code using CRC technique**

**********SENDER**********
Enter Message : 101010
Enter Pattern : 1111 (Condition : pattern size should be less than Message or else re enter pattern)

Given Message is : 101010
Given Pattern is : 1111
Message size is : 6
Pattern Size is : 4
FCS : 3bit
After Appending Q is :101010000
Remainder R is :100
Message to be Transmitted T is :101010**100**

Do You Want to Introduce error(Y/N) : Y
Enter the Position : 2 (condition : position between 1 to T size)

**********RECEIVER**********
Message received at the Receiver : 1**1**1010100
Remainder R is : 100
ERROR IN MESSAGE

Code:

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

int main() {
   char message[50], pattern[50], msg[50], reminder[20];
   int i, j, msglen, patlen, position, error_flag = 0;
   printf("**********SENDER**********");
   printf("\nEnter Message: ");
   scanf("%s", message);
   printf("Enter Pattern: ");
   scanf("%s", pattern);

   msglen = strlen(message);
   patlen = strlen(pattern);

   if (patlen >= msglen) {
      printf("Pattern size should be less than Message size. Please re-enter the pattern.\n");
      return 0;
   }

   printf("\nGiven Message is: %s\n", message);
   printf("Given Pattern is: %s\n", pattern);
   printf("Message size is: %d\n", msglen);
   printf("Pattern Size is: %d\n", patlen);
   printf("FCS: %dbit\n", patlen - 1);

   strcpy(msg, message);
   for (i = 0; i < patlen - 1; i++) {
      msg[msglen + i] = '0';
   }
```

```
    msg[msglen + i] = '\0';

    printf("After Appending Q is: %s\n", msg);

    for (i = 0; i < msglen; i++) {
        if (msg[i] == '1') {
            for (j = 0; j < patlen; j++) {
                if (msg[i + j] == pattern[j]) {
                    msg[i + j] = '0';
                } else {
                    msg[i + j] = '1';
                }
            }
        }
    }

    printf("Remainder R is: %s\n", &msg[msglen]);
    strcat(message, msg + msglen);
    printf("Message to be Transmitted T is: %s\n", message);

    char choice;
    printf("\nDo You Want to Introduce error (Y/N): ");
    scanf(" %c", &choice);

    if (choice == 'Y' || choice == 'y') {
        printf("Enter the Position (between 1 to T size): ");
        scanf("%d", &position);
    if (position > 0 && position <= strlen(message)) {
    if (message[position - 1] == '0') {
        message[position - 1] = '1';
    } else {
        message[position - 1] = '0';
    }
} else {
    printf("Invalid position.\n");
    return 0;
}

    }

    printf("\n************RECEIVER************\n");
    printf("Message received at the Receiver: %s\n", message);

   for (i = 0; i < msglen; i++) {
```

```
        if (message[i] == '1') {
            for (j = 0; j < patlen; j++) {
                if (message[i + j] == pattern[j]) {
                    message[i + j] = '0';
                } else {
                    message[i + j] = '1';
                }
            }
        }
    }
        strcpy(reminder,message+msglen);


        printf("Remainder R is: %s\n", reminder);

        for (i = 0; i < strlen(reminder); i++) {
            if (reminder[i] == '1') {
                error_flag = 1;
                break;
            }
        }

        if (error_flag == 1) {
            printf("\nERROR IN MESSAGE\n");
        } else {
            printf("\nNO ERROR IN MESSAGE\n");
        }

        return 0;
    }
```

Output:1

```
**********SENDER**********
Enter Message: 1010001101
Enter Pattern: 110101

Given Message is: 1010001101
Given Pattern is: 110101
Message size is: 10
Pattern Size is: 6
```

FCS: 5bit
After Appending Q is: 101000110100000
Remainder R is: 01110
Message to be Transmitted T is: 101000110101110

Do You Want to Introduce error (Y/N): n

**********RECEIVER**********
Message received at the Receiver: 101000110101110
Remainder R is: 00000

NO ERROR IN MESSAGE

Output:2

**********SENDER**********
Enter Message: 1101011011
Enter Pattern: 10011

Given Message is: 1101011011
Given Pattern is: 10011
Message size is: 10
Pattern Size is: 5
FCS: 4bit
After Appending Q is: 11010110110000
Remainder R is: 1110
Message to be Transmitted T is: 11010110111110

Do You Want to Introduce error (Y/N): y
Enter the Position (between 1 to T size): 11

**********RECEIVER**********
Message received at the Receiver: 11010110110110
Remainder R is: 1000

ERROR IN MESSAGE

NETWORK ASSIGNMENTS : 5 NS2-PING

1. **Simulate the transmission of ping messages over a network topology consisting of 6 nodes.**

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

proc finish {} {
global ns nf tf
$ns flush-trace
close $nf
close $tf
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]

$ns duplex-link $n0 $n2 100Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 5ms DropTail
$ns duplex-link $n2 $n3 1Mb 5ms DropTail
$ns duplex-link $n1 $n3 1Mb 5ms DropTail
$ns duplex-link $n4 $n5 1Mb 5ms DropTail
$ns duplex-link $n3 $n5 1Mb 5ms DropTail

$ns queue-limit $n0 $n2 50
```

```
$ns queue-limit $n1 $n2 50
$ns queue-limit $n2 $n3 50
$ns queue-limit $n1 $n3 50
$ns queue-limit $n4 $n5 50
$ns queue-limit $n3 $n5 50


Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] has received ping answer $from with round trip time $rtt
ms"}

set p1 [new Agent/Ping]
$ns attach-agent $n1 $p1

set p2 [new Agent/Ping]
$ns attach-agent $n3 $p2
$ns connect $p1 $p2

$ns at 0.2 "$p1 send"
$ns at 0.6 "$p2 send"
$ns at 0.4 "$p1 send"
$ns at 0.8 "$p2 send"

$ns at 1.0 "finish"
$ns run
```

2.Write a Program to implement a Ring topology using less number of statements in TCL Language apply PING Agent

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

```
$ns trace-all $tf

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

set numnodes 5

for { set i 0 } {$i<$numnodes} {incr i} {
set nodes($i) [$ns node]
}

for { set i 0 } {$i<$numnodes} {incr i} {
set j [expr {($i+1)%$numnodes}]
$ns duplex-link $nodes($i) $nodes($j) 1Mb 10ms DropTail }

Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] has received ping answer $from with round trip time $rtt
ms"}

set p1 [new Agent/Ping]
$ns attach-agent $nodes(0) $p1

set p2 [new Agent/Ping]
$ns attach-agent $nodes(2) $p2
$ns connect $p1 $p2

$ns at 0.2 "$p1 send"
$ns at 0.6 "$p2 send"
```

```
$ns at 0.4 "$p1 send"
$ns at 0.8 "$p2 send"

$ns at 1.0 "finish"
$ns run
```

3.Write a Program to implement a Star topology using less number of statements in TCL Language using UDP and TCP Agent

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

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

set numnodes 7

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

for { set i 1 } {$i<$numnodes} {incr i} {
$ns duplex-link $n(0) $n($i) 1Mb 10ms DropTail }

$ns duplex-link-op $n(0) $n(1) orient left-up
$ns duplex-link-op $n(0) $n(2) orient right-up
```

```
$ns duplex-link-op $n(0) $n(3) orient left
$ns duplex-link-op $n(0) $n(4) orient left-down

$ns duplex-link-op $n(0) $n(5) orient right
$ns duplex-link-op $n(0) $n(6) orient right-down


set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
$udp0 set fid_ 1
set null1 [new Agent/Null]
$ns attach-agent $n(2) $null1
$ns connect $udp0 $null1

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packet-size 500
$cbr0 set interval 0.005
$cbr0 attach-agent $udp0
$ns at 0.3 "$cbr0 start"
$ns at 1.5 "$cbr0 stop"

set tcp0 [new Agent/TCP]
$ns attach-agent $n(3) $tcp0
$tcp0 set fid_ 2
set tcpsink0 [new Agent/TCPSink]
$ns attach-agent $n(5) $tcpsink0
$ns connect $tcp0 $tcpsink0

set ftp0 [new Application/FTP]
$ftp0 set packet-size 500
$ftp0 set interval 0.01
$ftp0 attach-agent $tcp0
$ns at 0.5 "$ftp0 start"
$ns at 2.0 "$ftp0 stop"
```

$ns at 2.5 "finish"
$ns run

NETWORK ASSIGNMENTS :6

1. **Write a program to implement Leaky Bucket algorithm.**

Enter the Size of the Bucket : 20
Enter the out rate : 5
Enter the time Interval : 5
Enter the number of Packet : 5
Enter the time and Size of Packet 1 : 2 6
Enter the time and Size of Packet 2 : 3 10
Enter the time and Size of Packet 3 : 14 4
Enter the time and Size of Packet 4 : 16 15
Enter the time and Size of Packet 5 : 21 10

Operation	Time	Filled	Free-Space
Insert	2	6	14
Insert	3	16	4
Remove	5	11	9
Remove	10	6	14
Insert	14	10	10
Remove	15	5	15
Insert	16	20	0
Remove	20	15	5
Insert	21	overflow	
Remove	25	10	10
Remove	30	5	15
Remove	35	0	20

Code:

```
#include <stdio.h>

struct Packet {
    int time;
    int size;
```

```
} P[10];

int main() {
    int no_packets, bucket_size, out_rate, time_interval, free_space;

    printf("Enter the Size of the Bucket: ");
    scanf("%d", &bucket_size);
    free_space = bucket_size;

    printf("Enter the out rate: ");
    scanf("%d", &out_rate);

    printf("Enter the time Interval: ");
    scanf("%d", &time_interval);

    printf("Enter the number of Packets: ");
    scanf("%d", &no_packets);

    printf("Enter the time and Size of Packets:\n");
    for (int i = 0; i < no_packets; i++) {
        printf("Packet %d: ", i + 1);
        scanf("%d %d", &P[i].time, &P[i].size);
    }
    int filled=0;
    int time=1;//time multiplication
    int i = 0; // Index for packets

    printf("\nOperation\tTime\tFilled\tFree-Space\n");

        while (i < no_packets) {
        // Check if current time is a multiple of time interval
        if (P[i].time >= (time_interval * time)) {
            //  remove operation
            filled -= out_rate;
            free_space += out_rate;
            printf("Remove\t\t%d\t\t%d\t\t%d\n", time_interval * time, filled,free_space);
            time++; // Increment time
        }

        // Check for overflow
        else if (filled + P[i].size > bucket_size) {
            printf("Insert\t\t%d\t\toverflow\n",P[i].time);
            break;
        }
```

```
        else {
            // Insert packet
            filled += P[i].size;
            free_space -= P[i].size;
            printf("Insert\t\t%d\t\t%d\t\t%d\n", P[i].time, filled, free_space);
            i++;
        }
    }
    //remianing packet
while(filled!=0)
{
            filled -= out_rate;
            free_space += out_rate;
            printf("Remove\t\t%d\t\t%d\t\t%d\n", time_interval * time, filled, free_space);
            time++;
}
    return 0;
}
```

Output:

```
Enter the Size of the Bucket: 20
Enter the out rate: 5
Enter the time Interval: 5
Enter the number of Packets: 5
Enter the time and Size of Packets:
Packet 1: 2 6
Packet 2: 3 10
Packet 3: 14 4
Packet 4: 16 15
Packet 5: 21 10
```

Operation	Time	Filled	Free-Space
Insert	2	6	14
Insert	3	16	4
Remove	5	11	9
Remove	10	6	14
Insert	14	10	10
Remove	15	5	15
Insert	16	20	0
Remove	20	15	5

Insert	21	overflow	
Remove	25	10	10
Remove	30	5	15
Remove	35	0	20

2. Implement hamming distance method for error correction. Input binary data and convert input data into code words in transmitted message. Display received message and display corrected message in case of error.

**********SENDER**********
Enter Data : 1011001

Out Put
Code Word :0110
Transmission data T : 101**0**1101**1**1**0**

Do You Want to Introduce error(Y/N) : Y
Enter the Position : 2 (condition : position between 1 to T size)

**********RECEIVER**********
Message received at the Receiver : 10101101**0**0
Code Word :0010 Error in 2nd position

After Correction of Data : 10101101**1**0

Code:

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

int check(char msg[], int n) {
   int len = strlen(msg);
   int flag = 0, i = n - 1, count = 0;
   while (i < len) {
      int k = 0;
      while (k < n) {
         if (msg[i++] == '1')
            count++;
         k++;
      }
      k = 0;
```

```
        while (k < n) {
            k++;
            i++;
        }
    }
    return count;
}

int pow1(int a, int b) {
    int r = 1;
    for (int i = 0; i < b; i++) {
        r = r * a;
    }
    return r;
}

int main() {
    char msg[20], nmsg[20], hcode[20];
    int i = 0, j = 0, k = 0, count;

    printf("*** SENDER ***\n");
    printf("Enter Data:");
    scanf("%s", msg);

    int len = strlen(msg);
    while (msg[i]!='\0') {
        if ((j + 1) == pow1(2, k)) {
            nmsg[j++]=' ';
            nmsg[j]='\0';
            k++;
        } else {
            nmsg[j++] = msg[i++];
            nmsg[j]='\0';
        }
    }

    printf("\nAfter appending Empty space for message: %s\n", nmsg);

    len = strlen(nmsg);
    int l = 0;
    j = 0;
    while (pow1(2, l) <= len) {
        count = check(nmsg, pow1(2, l));
        if (count % 2 == 0)
            hcode[j++] = '0';
        else
            hcode[j++] = '1';
        l++;
    }
    hcode[j] = '\0';
```

```
printf("\nCode Word: %s\n", hcode);

j = 0;
i = 0;
k = 0;
while (nmsg[j] != '\0') {
   if ((j + 1) == pow1(2, k)) {
      nmsg[j] = hcode[i++];
      k++;
   }
   j++;
}

int pos;
char ch;
printf("\nTransmission Data: %s\n", nmsg);
printf("Do you want to introduce error (y/n): ");
scanf(" %c", &ch);

if (ch == 'y' || ch == 'Y') {
   printf("Enter the Position: ");
   scanf("%d", &pos);
   if (nmsg[pos - 1] == '1') {
      nmsg[pos - 1] = '0';
   } else {
      nmsg[pos - 1] = '1';
   }
}

l = 0;
j = 0;
printf("\n*** Receiver ***\n");
printf("Message received at the receiver: %s\n", nmsg);

while (pow1(2, l) <= len) {
   count = check(nmsg, pow1(2, l));
   if (count % 2 == 0)
      hcode[j++] = '0';
   else
      hcode[j++] = '1';
   l++;
}
len = strlen(hcode);
for (i = 0; i < len/2; i++) {
   int temp = hcode[i];
   hcode[i] = hcode[len - i - 1];
   hcode[len - i - 1] = temp;
}
```

```
    printf("Code Word: %s\n", hcode);

    int result = 0;
    pos = 0;
    for (i = len - 1; i >= 0; i--, pos++) {
        if (hcode[i] == '1')
            result += pow1(2, pos);
    }

    printf("Error in position: %d\n", result);

    if (nmsg[result - 1] == '1')
        nmsg[result - 1] = '0';
    else
        nmsg[result - 1] = '1';

    printf("Corrected Data: %s\n", nmsg);

    return 0;
}
```

Output:1

```
*** SENDER ***
Enter Data:1011001

After appending Empty space for message:   1 011 001

Code Word: 1001

Transmission Data: 10100111001
Do you want to introduce error (y/n): n

*** Receiver ***
Message received at the receiver: 10100111001
Code Word: 0000
Error in position: 0
Corrected Data: 10100111001
```

Output:2

```
*** SENDER ***
Enter Data:1011001

After appending Empty space for message:   1 011 001
```

Code Word: 1001

Transmission Data: 10100111001
Do you want to introduce error (y/n): Y
Enter the Position: 10

*** Receiver ***
Message received at the receiver: 10100111011
Code Word: 1010
Error in position: 10
Corrected Data: 10100111001

NETWORK ASSIGNMENTS : 7

1. Write a program to find the shortest path using **Dijkstra's Algorithm**

Enter a number of Node : 4

Enter distance to A to A : 0

Enter distance to A to B : 10

Enter distance to A to C : 0

Enter distance to A to D : 0

Enter distance to B to A : 10

Enter distance to B to B : 0

Enter distance to B to C : 3

Enter distance to B to D : 20

Enter distance to C to A : 0

Enter distance to C to B :3

Enter distance to C to C :0

Enter distance to C to D :10

Enter distance to D to A :0

Enter distance to D to B :20

Enter distance to D to C :10

Enter distance to D to D :0

Enter starting Node – A

Distance of Node B – 10

Route A -> B

Distance of Node C – 13

Route A -> B→ C

Distance of Node D – 23

Route A -> B ->C->D

Enter a Destination Node : C

The shortest route is - A -> B →C

Distance is 13

Code:

```
#include <stdio.h>
#define INFINITY 9999
#define MAX 100

void Dijkstra(int cost[MAX][MAX], int n, int start, int destination) {
   int 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 (cost[i][j] == 0)
            cost[i][j] = INFINITY;
         else
            cost[i][j] = cost[i][j];
      }
   }

   for (i = 0; i < n; i++) {
      distance[i] = cost[start][i];
```

```
        pred[i] = start;
        visited[i] = 0;
    }

    distance[start] = 0;
    visited[start] = 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 != start) {
            printf("Distance of Node %d: %d\n", i, distance[i]);
            printf("Route: %d", start);
            int path = pred[i];

            while (path != start) {
                printf("-->%d", path);
                path = pred[path];
            }
            printf("-->%d\n", i);
        }
    }

    printf("Shortest distance from source %d to destination %d: %d\n", start, destination,
distance[destination]);
    printf("Shortest path: %d ", start);
```

```
    int path = pred[destination];

    while (path != start) {
        printf("-->%d ", path);
        path = pred[path];
    }
    printf("--> %d\n", destination);
}

int main() {
    int cost[MAX][MAX], i, j, n, u, destination;
    printf("Enter the number of nodes:");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (i > j) {
                printf("Distance between %d and %d: %d\n", i, j, cost[j][i]);
                cost[i][j] = cost[j][i];
            } else {
                printf("Distance between %d and %d:", i, j);
                scanf("%d", &cost[i][j]);
            }
        }
    }
    printf("Enter the starting node:");
    scanf("%d", &u);

    printf("Enter the ending node:");
    scanf("%d", &destination);

    Dijkstra(cost, n, u, destination);
    return 0;
}
```

Output:1

```
Enter the number of nodes:4
Distance between 0 and 0:0
Distance between 0 and 1:10
Distance between 0 and 2:0
Distance between 0 and 3:0
```

Distance between 1 and 0: 10
Distance between 1 and 1:0
Distance between 1 and 2:3
Distance between 1 and 3:20
Distance between 2 and 0: 0
Distance between 2 and 1: 3
Distance between 2 and 2:0
Distance between 2 and 3:10
Distance between 3 and 0: 0
Distance between 3 and 1: 20
Distance between 3 and 2: 10
Distance between 3 and 3:0
Enter the starting node:0
Enter the ending node:2
Distance of Node 1: 10
Route: 0-->1
Distance of Node 2: 13
Route: 0-->1-->2
Distance of Node 3: 23
Route: 0-->2-->1-->3
Shortest distance from source 0 to destination 2: 13
Shortest path: 0 -->1 --> 2

Output 2:

Enter the number of nodes:9
Distance between 0 and 0:0
Distance between 0 and 1:4
Distance between 0 and 2:0
Distance between 0 and 3:0
Distance between 0 and 4:0
Distance between 0 and 5:0
Distance between 0 and 6:0
Distance between 0 and 7:8
Distance between 0 and 8:0

Distance between 1 and 0: 4
Distance between 1 and 1:0
Distance between 1 and 2:8
Distance between 1 and 3:0
Distance between 1 and 4:0
Distance between 1 and 5:0
Distance between 1 and 6:0

Distance between 1 and 7:11
Distance between 1 and 8:0

Distance between 2 and 0: 0
Distance between 2 and 1: 8
Distance between 2 and 2:0
Distance between 2 and 3:7
Distance between 2 and 4:0
Distance between 2 and 5:4
Distance between 2 and 6:0
Distance between 2 and 7:0
Distance between 2 and 8:2

Distance between 3 and 0: 0
Distance between 3 and 1: 0
Distance between 3 and 2: 7
Distance between 3 and 3:0
Distance between 3 and 4:9
Distance between 3 and 5:14
Distance between 3 and 6:0
Distance between 3 and 7:0
Distance between 3 and 8:0

Distance between 4 and 0: 0
Distance between 4 and 1: 0
Distance between 4 and 2: 0
Distance between 4 and 3: 9
Distance between 4 and 4:0
Distance between 4 and 5:10
Distance between 4 and 6:0
Distance between 4 and 7:0
Distance between 4 and 8:0

Distance between 5 and 0: 0
Distance between 5 and 1: 0
Distance between 5 and 2: 4
Distance between 5 and 3: 14
Distance between 5 and 4: 10
Distance between 5 and 5:0
Distance between 5 and 6:2
Distance between 5 and 7:0
Distance between 5 and 8:0

Distance between 6 and 0: 0

Distance between 6 and 1: 0
Distance between 6 and 2: 0
Distance between 6 and 3: 0
Distance between 6 and 4: 0
Distance between 6 and 5: 2
Distance between 6 and 6:0
Distance between 6 and 7:1
Distance between 6 and 8:6

Distance between 7 and 0: 8
Distance between 7 and 1: 11
Distance between 7 and 2: 0
Distance between 7 and 3: 0
Distance between 7 and 4: 0
Distance between 7 and 5: 0
Distance between 7 and 6: 1
Distance between 7 and 7:0
Distance between 7 and 8:7

Distance between 8 and 0: 0
Distance between 8 and 1: 0
Distance between 8 and 2: 2
Distance between 8 and 3: 0
Distance between 8 and 4: 0
Distance between 8 and 5: 0
Distance between 8 and 6: 6
Distance between 8 and 7: 7
Distance between 8 and 8:0

Enter the starting node:1
Enter the ending node:4
Distance of Node 0: 4
Route: 1-->0
Distance of Node 2: 8
Route: 1-->2
Distance of Node 3: 15
Route: 1-->2-->3
Distance of Node 4: 22
Route: 1-->5-->2-->4
Distance of Node 5: 12
Route: 1-->2-->5
Distance of Node 6: 12
Route: 1-->7-->6
Distance of Node 7: 11

Route: 1-->7
Distance of Node 8: 10
Route: 1-->2-->8
Shortest distance from source 1 to destination 4: 22
Shortest path: 1 -->5 -->2 --> 4

NETWORK ASSIGNMENTS : 8

1. Write a program to construct distance vector table for all the router of a given network using Bellman-Ford Distance Vector Routing protocol method.

Example:

Enter the number of nodes : 4
Enter the cost matrix:

	1	2	3	4
1	0	3	5	999
2	3	0	999	1
3	5	4	0	2
4	999	1	2	0

Intial Table
Router 1

Destination	Distance	Next Hop
1	0	1
2	3	2
3	5	3
4	999	-

Similarly construct for Router 2, 3 ,4.

Updated table
Router 1

Destination	Distance	Next Hop
1	0	1
2	3	2
3	5	3
4	4	2

Router 2

Destination	Distance	Next Hop
1	3	1
2	0	2
3	3	4
4	1	4

Router 3

Destination	Distance	Next Hop
1	5	1
2	3	4

3	0	3
4	2	4

Router 4

Destination	Distance	Next Hop
1	4	2
2	1	2
3	2	3
4	0	4

Code:

```
#include <stdio.h>

struct node
{
   int distance[20];
   int from[20];
} router[10];

int main()
{
   int distance_matrix[20][20];
   int n, i, j, k, count = 0;

   printf("Enter the number of nodes: ");
   scanf("%d", &n);

   printf("\nEnter the cost matrix:(Assign value 999 if there is no direct connection)\n");
   for (i = 0; i < n; i++)
```

```
    for (j = 0; j < n; j++)
    {
        scanf("%d", &distance_matrix[i][j]);
        distance_matrix[i][i] = 0;
        router[i].distance[j] = distance_matrix[i][j];
        router[i].from[j] = j;
    }

printf("\nInitial Tables:\n");
for (i = 0; i < n; i++)
{
    printf("\nRouter %d:\n", i + 1);
    printf("Destination\tDistance\tNext Hop\n");

    for (j = 0; j < n; j++)
    {
        if (router[i].distance[j] == 999)
        {
            printf("%d\t\t%d\t\t-\n", j + 1, router[i].distance[j]);
        }
        else
        {
            printf("%d\t\t%d\t\t%d\n", j + 1, router[i].distance[j], router[i].from[j] + 1);
        }
    }
}

do
{
    count = 0;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            for (k = 0; k < n; k++)
                if (router[i].distance[j] > distance_matrix[i][k] + router[k].distance[j])
                {
                    router[i].distance[j] = router[i].distance[k] + router[k].distance[j];
                    router[i].from[j] = k;
                    count++;
                }
} while (count != 0);
printf("\nUpdated Tables:\n");

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

```
        printf("\nRouter %d:\n", i + 1);
        printf("Destination\tDistance\tNext Hop\n");

        for (j = 0; j < n; j++)
        {
            printf("%d\t\t%d\t\t%d\n", j + 1, router[i].distance[j], router[i].from[j] + 1);
        }
    }

    printf("\n");
    return 0;
}
```

Output:

Enter the number of nodes: 4

Enter the cost matrix:(Assign value 999 if there is no direct connection)
0 3 5 999
3 0 999 1
5 4 0 2
999 1 2 0

Initial Tables:

Router 1:

Destination	Distance	Next Hop
1	0	1
2	3	2
3	5	3
4	999	-

Router 2:

Destination	Distance	Next Hop
1	3	1
2	0	2
3	999	-
4	1	4

Router 3:

Destination	Distance	Next Hop

1	5	1
2	4	2
3	0	3
4	2	4

Router 4:

Destination	Distance	Next Hop
1	999	-
2	1	2
3	2	3
4	0	4

Updated Tables:

Router 1:

Destination	Distance	Next Hop
1	0	1
2	3	2
3	5	3
4	4	2

Router 2:

Destination	Distance	Next Hop
1	3	1
2	0	2
3	3	4
4	1	4

Router 3:

Destination	Distance	Next Hop
1	5	1
2	3	4
3	0	3
4	2	4

Router 4:

Destination	Distance	Next Hop
1	4	2
2	1	2
3	2	3
4	0	4