1a) count number of bytes

```
#include<stdio.h>
#include<string.h>
char frames[100];
char sendingFrames[1000];
int main() {
   int n;
   char l[100];
   printf("How many frames do you want to send\n");
   scanf("%d", &n);
   int len;
   //int len2 = strlen(sendingFrames);
   for(int i = 0; i < n; i++){
      printf("Enter the frame %d\n", i + 1);
      scanf("%s", frames);
      len = strlen(frames);
      sprintf(l, "%d", len);
      strcat(l, frames);
      strcat(sendingFrames, l);

   }
   printf("Sending frames: \n");
   printf("%s\n", sendingFrames);

   printf("Received frames\n");
   for(int i = 0; i < strlen(sendingFrames); i++) {
      int length = sendingFrames[i] - '0';
      for(int j = i + 1; j <= (i + length); j++) {
         printf("%c", sendingFrames[j]);
      }
```

```
        printf("\n");
        i = i + length;
    }
    return 0;

}
```

1b) bit stuff

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

void sender();
void receiver(int *message, int l2);

int main()
{
        sender();
        return 0;
}

void sender()
{
        int i, j, n, count = 0, onecounter = 0, l1, zero = 0;
        int msg[50];
        int result[50];

        printf("Enter the number of bits to be sent: ");
        scanf("%d", &n);
        printf("Enter the bits: ");
```

```
for (i = 0; i < n; i++)
{
	scanf("%d", &msg[i]);
}
result[0] = 0;
result[1] = 1;
result[2] = 1;
result[3] = 1;
result[4] = 1;
result[5] = 1;
result[6] = 1;
result[7] = 0;
j = 8;

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

	if (count == 5 && zero == 1)
	{
		result[j++] = 0;
		onecounter++;
		count = 0;
	}
	if (msg[i] == 0)
	{
		result[j++] = msg[i];
		count = 0;
		zero = 1;
	}
	else
	{
```

```
                result[j++] = msg[i];
                count++;
            }
        }
        if (count == 5)
        {
            result[j++] = 0;
            onecounter++;
            count = 0;
        }
        result[j++] = 0;
        result[j++] = 1;
        result[j++] = 1;
        result[j++] = 1;
        result[j++] = 1;
        result[j++] = 1;
        result[j++] = 1;
        result[j++] = 0;
        l1 = 16 + n + onecounter;
        for (i = 0; i < l1; i++)
        {
            printf("%d", result[i]);
        }
        receiver(result, l1);
}

void receiver(int *result, int l2)
{
        int i, j, counter = 0, l3;
        int mesg[100];
        l3 = l2 - 8;
```

```
	j = 0;
	for (i = 8; i < l3; i++)
	{

		if (counter == 5)
		{
			i++;
			counter = 0;
		}

		if (result[i] == 0)
		{
			mesg[j] = result[i];
			j++;
			counter = 0;
		}
		else
		{
			mesg[j] = result[i];
			j++;
			counter++;
		}
	}
	printf("\n Receiver side message is:");
	for (i = 0; i < j; i++)
	{
		printf("%d", mesg[i]);
	}
}
```

2) crc

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#define N strlen(g)
char t[120],cs[120],g[]="100000111";
int a,c,e;
void xor()
{
 for(c=1;c<N;c++)
 cs[c]=((cs[c]==g[c])?'0':'1');
}
void crc()

{
for(e=0;e<N;e++)
 cs[e]=t[e];
 do
  {
   if(cs[0]=='1')
    xor();
   for(c=0;c<N-1;c++)
cs[c]=cs[c+1];
cs[c]=t[e++];
}
while(e<=a+N-1);
}
void main()
{
printf("enter the polynomial\n");
```

```
scanf("%s",t);
printf("generating polynomial is %s\n",g);
a=strlen(t);
for(e=a;e<a+N-1;e++)
t[e]='0';
printf("modified t[u] is %s\n",t);
crc();
printf("checksum is :%s\n",cs);
for(e=a;e<a+N-1;e++)
t[e]=cs[e-a];
printf("final codeword is :%s\n",t);
printf("test error detection 0(yes)1(no)?:\n");
scanf("%d",&e);
if(e==0)
 {
 do
  {
   printf("enter position where error has to be inserted\n");
   scanf("%d",&e);
  }
while(e==0 || e>a+N-1);
 t[e-1]=(t[e-1]=='0')?'1':'0';
 printf("errorneous data %s\n",t);
}

crc();

for(e=0;(e<N-1)&&(cs[e]!='1');e++);
 if(e<N-1)
  printf("error detected\n");
 else
```

```
 printf("error is not detected\n");
}
```

3a) TCP Client-Server

TCP SERVER

```
#include<stdio.h>
#include<stdlib.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<errno.h>
#include<unistd.h>
#include<netinet/in.h>
#include<string.h>
int main()
{
        struct sockaddr_in server_addr;
        struct sockaddr_in client_addr;
        FILE *fptr;
        int sock,connected,bytes_recv;
        char ch,send_data[1024],recv_data[1024];
        int sin_size,flag = 0;


        if((sock=socket(AF_INET,SOCK_STREAM,0))==-1)
        {
                perror("socket");
                exit(1);
        }


        server_addr.sin_family=AF_INET;
        server_addr.sin_port=htons(6119);
        server_addr.sin_addr.s_addr=inet_addr("127.0.0.1");

        if(bind(sock,(struct sockaddr *)&server_addr, sizeof(struct
sockaddr))==-1)
        {
                perror("unable to bind");
                exit(1);
        }

        if(listen(sock,5)==-1)
        {
                perror("lsten");
                exit(1);
        }

        printf("tcp server is waiting for client on port XXXX\n");
        sin_size=sizeof(struct sockaddr_in);
        connected=accept(sock,(struct sockaddr
*)&client_addr,&sin_size);
```

```
        while(1)
        {

                bytes_recv=recv(connected,recv_data,1024,0);
                recv_data[bytes_recv]='\0';

                printf("reciecved data is %s\n\n\n",recv_data);


                fptr=fopen(recv_data,"r");
                if(fptr==NULL)
                {
                        strcpy(send_data,"FILE");
                        send(connected,send_data,strlen(send_data),0);
                }
                ch = fgetc(fptr);

                while(ch != EOF)//this loop searches the for the current
word
                {
                   // fscanf(fptr,"%s",send_data);
                        send_data[flag] = ch;
                        flag++;
                        ch = fgetc(fptr);
                   //send(connected,send_data,strlen(send_data),0);
                }
                        send(connected,send_data,strlen(send_data),0);
                        //send_data[0] = 'q';
                        //strcpy(send_data,"q");
                        //send(connected,send_data,strlen(send_data),0);
                        close(connected);
                        break;
        }
}
```

TCP CLIENT

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<arpa/inet.h>
#include<netdb.h>
#include<netinet/in.h>
#include<errno.h>
#include<string.h>
int main()
{
        int sock,bytes_recv;
        struct sockaddr_in server_addr;
        char recv_data[1024],send_data[1024];
        struct hostent *host;
        host=gethostbyname("127.0.0.1");
        if((sock=socket(AF_INET,SOCK_STREAM,0))==-1)
        {
                perror("socket");
```

```
                exit(1);
        }
        server_addr.sin_family=AF_INET;
        server_addr.sin_port=htons(6119);
        server_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
        if(connect(sock,(struct sockaddr *)&server_addr,sizeof(struct
sockaddr))==-1)
        {
                perror("connect");
                exit(1);
        }
                printf("send Filename to send\n");
                gets(send_data);

                if(strcmp(send_data,"q")!=0)
                        send(sock,send_data,strlen(send_data),0);

                while((bytes_recv=recv(sock,recv_data,1024,0))>0)
                {
                        recv_data[bytes_recv]='\0';
                        //printf("%s\n\n", recv_data);
                        //if(strcmp(recv_data,"q")==0)
                //      {
                //      close(sock);
                //      break;
                //      }
                        printf("%s\n", recv_data);
                }
        close(sock);
        return 0;
}
```

3b) UDP Client-Server

UDP SERVER

```
#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;

  /*Create UDP socket*/
  udpSocket = socket(PF_INET, SOCK_DGRAM, 0);

  /*Configure settings in address struct*/
  serverAddr.sin_family = AF_INET;
```

```
  serverAddr.sin_port = htons(8893);
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

  /*Bind socket with address struct*/
  bind(udpSocket, (struct sockaddr *) &serverAddr, sizeof(serverAddr));

  /*Initialize size variable to be used later on*/
  addr_size = sizeof serverStorage;

  while(1){
    /* Try to receive any incoming UDP datagram. Address and port of
 *       requesting client will be stored on serverStorage variable */
    nBytes = recvfrom(udpSocket,buffer,1024,0,(struct sockaddr
*)&serverStorage, &addr_size);

    /*Convert message received to uppercase*/
    for(i=0;i<nBytes-1;i++)
      buffer[i] = toupper(buffer[i]);

    /*Send uppercase message back to client, using serverStorage as the
address*/
    sendto(udpSocket,buffer,nBytes,0,(struct sockaddr
*)&serverStorage,addr_size);
  }

  return 0;
}
```

UDP CLIENT

```
#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;

  /*Create UDP socket*/
  clientSocket = socket(PF_INET, SOCK_DGRAM, 0);

  /*Configure settings in address struct*/
  serverAddr.sin_family = AF_INET;
  serverAddr.sin_port = htons(8893);
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

  /*Initialize size variable to be used later on*/
  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;

    /*Send message to server*/
    sendto(clientSocket,buffer,nBytes,0,(struct sockaddr
*)&serverAddr,addr_size);

    /*Receive message from server*/
                nBytes = recvfrom(clientSocket,buffer,1024,0,NULL,
NULL);

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

  }

  return 0;
}
```

4) Distant Vector

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

```
                {
                        if(rt[i].dist[j]>dmat[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<n;i++)
{
        printf("\n\nState value for router %d is \n",i+1);
        printf("\nNode \t Via \t Dist. ");
        for(j=0;j<n;j++)
        {
                printf("\n%d \t %d \t %d
",j+1,rt[i].from[j]+1,rt[i].dist[j]);
        }
}
printf("\n\n");
}
```

5) Leaky bucket

```
#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("\n enter bucket size : ");
        scanf("%d",&cap);
        printf("\n enter output rate :");
        scanf("%d",&orate);
        do{
        printf("\n enter number of packets coming at second %d :",i+1);
        scanf("%d",&inp[i]);
        if(inp[i]>cap)
        {
                printf("Bucket overflow\n");
                printf("Packet Discarded\n");
                exit(0);
        }
        i++;
        printf("\n enter 1 to contiue or 0 to quit.........");
        scanf("%d",&ch);
}
while(ch);
nsec=i;
printf("\n Second \t Recieved \t Sent \t Dropped \tRemained \n");
for(i=0;count || i<nsec;i++)
{
        printf("  %d",i+1);
```

```
        printf(" \t\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;
}
```

PART B

1) Udp p2p

```
#include "ns3/netanim-module.h"

#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/internet-module.h"

#include "ns3/point-to-point-module.h"

#include "ns3/applications-module.h"


using namespace ns3;

int

main (int argc, char *argv[])
```

```
{
  Time::SetResolution (Time::NS);
  NodeContainer nodes;
  nodes.Create (2);

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);

  InternetStackHelper stack;
  stack.Install (nodes);

  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");

  Ipv4InterfaceContainer interfaces = address.Assign (devices);

  UdpEchoServerHelper echoServer (9);

  ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
```

```
  ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));
 AnimationInterface anim ("first.xml");
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

2)UDP CSMA

```
#include <fstream>
#include "ns3/core-module.h"
#include "ns3/csma-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"

using namespace ns3;

int
main (int argc, char *argv[])
{
  Address serverAddress;
NodeContainer n;
  n.Create (4);
InternetStackHelper internet;
  internet.Install (n);
CsmaHelper csma;
  csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (5000000)));
  csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
```

```
  csma.SetDeviceAttribute ("Mtu", UintegerValue (1400));
  NetDeviceContainer d = csma.Install (n);
Ipv4AddressHelper ipv4;
     ipv4.SetBase ("10.1.1.0", "255.255.255.0");
     Ipv4InterfaceContainer i = ipv4.Assign (d);
     serverAddress = Address(i.GetAddress (1));

  uint16_t port = 9;  // well-known echo port number
  UdpEchoServerHelper server (port);
  ApplicationContainer apps = server.Install (n.Get (1));
  apps.Start (Seconds (1.0));
  apps.Stop (Seconds (10.0));



  uint32_t packetSize = 1024;
  uint32_t maxPacketCount = 1;
  Time interPacketInterval = Seconds (1.);
  UdpEchoClientHelper client (serverAddress, port);
  client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount));
  client.SetAttribute ("Interval", TimeValue (interPacketInterval));
  client.SetAttribute ("PacketSize", UintegerValue (packetSize));
  apps = client.Install (n.Get (0));
  apps.Start (Seconds (2.0));
  apps.Stop (Seconds (10.0));

#if 0
client.SetFill (apps.Get (0), "Hello World");
client.SetFill (apps.Get (0), 0xa5, 1024);
uint8_t fill[] = { 0, 1, 2, 3, 4, 5, 6};
  client.SetFill (apps.Get (0), fill, sizeof(fill), 1024);
#endif
```

```
AnimationInterface anim ("second.xml");
  Simulator::Run ();
  Simulator::Destroy ();

}
```

3)UDP P2P AND CSMA

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"
using namespace ns3;
int
main (int argc, char *argv[])
{

  uint32_t nCsma = 3;
NodeContainer p2pNodes;
  p2pNodes.Create (2);

  NodeContainer csmaNodes;
  csmaNodes.Add (p2pNodes.Get (1));
  csmaNodes.Create (nCsma);

  PointToPointHelper pointToPoint;
```

```
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);

InternetStackHelper stack;
stack.Install (p2pNodes.Get (0));
stack.Install (csmaNodes);

Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);

address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);

UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
```

```
  UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

  ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  pointToPoint.EnablePcapAll ("second");
  csma.EnablePcap ("second", csmaDevices.Get (1), true);
AnimationInterface anim ("third.xml");
Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

4) TCP P2P

```
#include <string>
#include <fstream>
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/network-module.h"
#include "ns3/packet-sink.h"
#include "ns3/netanim-module.h"
```

```
using namespace ns3;
int
main (int argc, char *argv[])
{



 uint32_t maxBytes = 0;
NodeContainer nodes;
 nodes.Create (2);
PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("500Kbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("5ms"));
NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);
InternetStackHelper internet;
 internet.Install (nodes);
Ipv4AddressHelper ipv4;
 ipv4.SetBase ("10.1.1.0", "255.255.255.0");
 Ipv4InterfaceContainer i = ipv4.Assign (devices);
uint16_t port = 9;  // well-known echo port number
 BulkSendHelper source ("ns3::TcpSocketFactory",
 InetSocketAddress (i.GetAddress (1), port));
  source.SetAttribute ("MaxBytes", UintegerValue (maxBytes));
  ApplicationContainer sourceApps = source.Install (nodes.Get (0));
  sourceApps.Start (Seconds (0.0));
  sourceApps.Stop (Seconds (10.0));
PacketSinkHelper sink ("ns3::TcpSocketFactory",
  InetSocketAddress (Ipv4Address::GetAny (), port));
  ApplicationContainer sinkApps = sink.Install (nodes.Get (1));
  sinkApps.Start (Seconds (0.0));
```

```
  sinkApps.Stop (Seconds (10.0));
Simulator::Stop (Seconds (10.0));
AnimationInterface anim ("fourth.xml");
anim.EnablePacketMetadata(true);
 Simulator::Run ();
  Simulator::Destroy ();
  }
```