NETWORK AND COMMUNICATION

LAB DIGITAL ASSIGNMENT 1

ARSHDEEP SINGH
19BCB0086

AIM

To create a c program that implements the basic error detection and correction mechanisms.

Theory used

- Even parity
- Odd parity
- CRC
- Check sum
- Hamming code (on 4 bits)

Code logic / flow

1. User is given a menu
2. User can select any 1 method of the choices and enters the number of the choice to select it
3. The respective functions are run and implement the algorithms based on the logic for the topics **listed in theory.**
4. Each of the choice invokes the corresponding user defined function and generates output.
5. Program terminates

Code(IDE VSCODE)

```
#include<stdio.h>
#include<string.h>
void check(){

   printf("THIS IS CHECKSUM FUNCTION\n");
   char a[20],b[20];
   char sum[20],complement[20];
   int i,length;

   printf("Enter first binary string\n");
   scanf("%s",a);
   printf("Enter second binary string\n");
   scanf("%s",b);

   if(strlen(a)==strlen(b)){
       length = strlen(a);
       char carry='0';

       for(i=length-1;i>=0;i--)
       {
           if(a[i]=='0' && b[i]=='0' && carry=='0')
           {
               sum[i]='0';
               carry='0';
           }
           else if(a[i]=='0' && b[i]=='0' && carry=='1')
           {
               sum[i]='1';
               carry='0';
           }
           else if(a[i]=='0' && b[i]=='1' && carry=='0')
           {
               sum[i]='1';
               carry='0';
           }
           else if(a[i]=='0' && b[i]=='1' && carry=='1')
```

```
        {
            sum[i]='0';
            carry='1';
        }
        else if(a[i]=='1' && b[i]=='0' && carry=='0')
        {
            sum[i]='1';
            carry='0';
        }
        else if(a[i]=='1' && b[i]=='0' && carry=='1')
        {
            sum[i]='0';
            carry='1';
        }
        else if(a[i]=='1' && b[i]=='1' && carry=='0')
        {
            sum[i]='0';
            carry='1';
        }
        else if(a[i]=='1' && b[i]=='1' && carry=='1')
        {
            sum[i]='1';
            carry='1';
        }
        else
            break;
    }

    printf("\nSum=%c%s",carry,sum);

    for(i=0;i<length;i++)
    {
        if(sum[i]=='0')
            complement[i]='1';
        else
            complement[i]='0';
    }

    if(carry=='1')
        carry='0';
```

```
        else
            carry='1';

        printf("\nChecksum=%c%s\n",carry,complement);
    }
    else {
        printf("\nWrong input strings");
    }
}

void crc(){

    printf("THIS IS CRC FUNCTION\n");
    int i,j,keylen,msglen;
    char input[100], key[30],temp[30],quot[100],rem[30],key1[30];
    printf("Enter Data: ");
    scanf("%s",input);
    printf("Enter Key divisor: ");
    scanf("%s",key);
    keylen=strlen(key);
    msglen=strlen(input);
    strcpy(key1,key);
    //appending 0
    for (i=0;i<keylen-1;i++) {
        input[msglen+i]='0';
    }

    //storing input in temp
    for (i=0;i<keylen;i++)
     temp[i]=input[i];


    for (i=0;i<msglen;i++) {
        quot[i]=temp[0];
        if(quot[i]=='0')
         for (j=0;j<keylen;j++)
         key[j]='0';
        else
         for (j=0;j<keylen;j++)
         key[j]=key1[j];
```

```
        for (j=keylen-1;j>0;j--) {
            if(temp[j]==key[j])
             rem[j-1]='0';
            else
             rem[j-1]='1';
        }
        rem[keylen-1]=input[i+keylen];
        strcpy(temp,rem);
    }
    strcpy(rem,temp);
    printf("\nQuotient is ");
    for (i=0;i<msglen;i++)
     printf("%c",quot[i]);

    printf("\nRemainder/CRC bits is ");
    for (i=0;i<keylen-1;i++)
     printf("%c",rem[i]);
    printf("\nFinal data is: ");
    for (i=0;i<msglen;i++)
     printf("%c",input[i]);
    for (i=0;i<keylen-1;i++)
     printf("%c",rem[i]);
     printf("\n");
}
void ham_check(){
    {
    int data[10];

    printf("THIS IS HAMMING CODE FUNCTION\n");
    int dataatrec[10],c,c1,c2,c3,i;
    printf("Enter 4 bits of data one by one\n");
    scanf("%d",&data[0]);
    scanf("%d",&data[1]);
    scanf("%d",&data[2]);
    scanf("%d",&data[4]);
    data[6]=data[0]^data[2]^data[4];
    data[5]=data[0]^data[1]^data[4];
    data[3]=data[0]^data[1]^data[2];
    printf("\nEncoded data is\n");
```

```
   for(i=0;i<7;i++)
       printf("%d",data[i]);
   printf("\n\nEnter received data bits one by one\n");
   for(i=0;i<7;i++)
       scanf("%d",&dataatrec[i]);
   c1=dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];
   c2=dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];
   c3=dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0];
   c=c3*4+c2*2+c1 ;
   if(c==0) {
       printf("\nNo error while transmission of data\n");
   }
   else {
       printf("\nError on position %d",c);

       printf("\nData sent : ");
       for(i=0;i<7;i++)
           printf("%d",data[i]);

       printf("\nData received : ");
       for(i=0;i<7;i++)
           printf("%d",dataatrec[i]);

       printf("\nCorrect message is\n");
       if(dataatrec[7-c]==0)
           dataatrec[7-c]=1;
       else
           dataatrec[7-c]=0;

       for (i=0;i<7;i++) {
           printf("%d",dataatrec[i]);
       }
   }
}
printf("\n");

}

void odd(){
```

```
   printf("THIS IS  ODD PARITY FUNCTION\n");
   printf("enter the size: ");
   int k;
   scanf("%d",&k);
   printf("ENTER %d BINARY DIGITS TO REPRESENT THE DATA : ",k);
   char data[k+1];
   int i=0;
   int count=0;
   scanf("%s",data);
   i=0;
   for(i=0;i<k;i++)
   {
       if(data[i]=='1')count++;
   }

   if(count%2==0){
       data[k]='1';
   }
   else data[k]='0';

   printf("The data bit appended is %c \n",data[k]);
   printf("The data created and sent in odd-parity mode is ");
   for(i=0;i<k+1;i++)printf("%c",data[i]);
   printf("\n");
}

void even(){

   printf("THIS IS EVEN PARITY FUNCTION\n");
   printf("enter the size: ");
   int k;
   scanf("%d",&k);
   printf("ENTER %d BINARY DIGITS TO REPRESENT THE DATA : ",k);
   char data[k+1];
   int i=0;
   int count=0;
   scanf("%s",data);
   i=0;
   for(i=0;i<k;i++)
   {
```

```
        if(data[i]=='1')count++;
    }

    if(count%2==1){
        data[k]='1';
    }
    else data[k]='0';

    printf("The data bit appended is %c \n",data[k]);
    printf("The data created and sent in even-parity mode is ");
    for(i=0;i<k+1;i++)printf("%c",data[i]);
    printf("\n");
}

int main(){
    printf("choose a parity check to be demonstrated\n1.even parity\n2.odd
parity\n3.CRC\n4.checksum\n5.hamming \nEnter the choice: ");
    int n;
    scanf("%d",&n);

    if (n==1){
        even();
    }

    else if(n==2){
        odd();
    }

    else if(n==3){
        crc();
    }
    else if (n==4){
        check();
    }
    else if(n==5){
        ham_check();
    }
}
```

OUTPUT

1. EVEN PARITY

```
arshdeep@arshdeep-HP-Laptop-14s-cr1xxx:~/Desktop/netcom$ ./ex1
choose a parity check to be demonstrated
1.even parity
2.odd parity
3.CRC
4.checksum
5.hamming
Enter the choice: 1
THIS IS EVEN PARITY FUNCTION
enter the size: 5
ENTER 5 BINARY DIGITS TO REPRESENT THE DATA : 10100
The data bit appended is 0
The data created and sent in even-parity mode is 101000
```

2. ODD PARITY(WITH SAME NUMBER FOR EASY VERIFICATION)

```
The data created and sent in even-parity mode is 101000
arshdeep@arshdeep-HP-Laptop-14s-cr1xxx:~/Desktop/netcom$ ./ex1
choose a parity check to be demonstrated
1.even parity
2.odd parity
3.CRC
4.checksum
5.hamming
Enter the choice: 2
THIS IS  ODD PARITY FUNCTION
enter the size: 5
ENTER 5 BINARY DIGITS TO REPRESENT THE DATA : 10100
The data bit appended is 1
The data created and sent in odd-parity mode is 101001
```

3. CRC

```
arshdeep@arshdeep-HP-Laptop-14s-cr1xxx:~/Desktop/netcom$ ./ex1
choose a parity check to be demonstrated
1.even parity
2.odd parity
3.CRC
4.checksum
5.hamming
Enter the choice: 3
THIS IS CRC FUNCTION
Enter Data: 101011
Enter Key divisor: 101

Quotient is 100011
Remainder/CRC bits is 11
Final data is: 10101111
arshdeep@arshdeep-HP-Laptop-14s-cr1xxx:~/Desktop/netcom$ ./ex1
choose a parity check to be demonstrated
1.even parity
2.odd parity
3.CRC
4.checksum
5.hamming
Enter the choice: 3
THIS IS CRC FUNCTION
Enter Data: 101001
Enter Key divisor: 1001

Quotient is 101100
Remainder/CRC bits is 100
Final data is: 101001100
arshdeep@arshdeep-HP-Laptop-14s-cr1xxx:~/Desktop/netcom$ ./ex1
choose a parity check to be demonstrated
1.even parity
2.odd parity
3.CRC
4.checksum
5.hamming
Enter the choice: 3
THIS IS CRC FUNCTION
Enter Data: 1000101
Enter Key divisor: 1000

Quotient is 1000101
Remainder/CRC bits is 000
Final data is: 1000101000
```

4. CHECKSUM

```
arshdeep@arshdeep-HP-Laptop-14s-cr1xxx:~/Desktop/netcom$ ./ex1
choose a parity check to be demonstrated
1.even parity
2.odd parity
3.CRC
4.checksum
5.hamming
Enter the choice: 4
THIS IS CHECKSUM FUNCTION
Enter first binary string
100101
Enter second binary string
100100

Sum=1001001
Checksum=0110110
arshdeep@arshdeep-HP-Laptop-14s-cr1xxx:~/Desktop/netcom$ ./ex1
choose a parity check to be demonstrated
1.even parity
2.odd parity
3.CRC
4.checksum
5.hamming
Enter the choice: 4
THIS IS CHECKSUM FUNCTION
Enter first binary string
1001
Enter second binary string
0001

Sum=01010
Checksum=10101
arshdeep@arshdeep-HP-Laptop-14s-cr1xxx:~/Desktop/netcom$ ./ex1
choose a parity check to be demonstrated
1.even parity
2.odd parity
3.CRC
4.checksum
5.hamming
Enter the choice: 4
THIS IS CHECKSUM FUNCTION
Enter first binary string
10001
Enter second binary string
01110

Sum=011111
Checksum=100000
```

5. HAMMING CODE

```
arshdeep@arshdeep-HP-Laptop-14s-cr1xxx:~/Desktop/netcom$ ./ex1
choose a parity check to be demonstrated
1.even parity
2.odd parity
3.CRC
4.checksum
5.hamming
Enter the choice: 5
THIS IS HAMMING CODE FUNCTION
Enter 4 bits of data one by one
1 0 1 0

Encoded data is
1010010

Enter received data bits one by one
1 1 1 0 0 1 0

Error on position 6
Data sent : 1010010
Data received : 1110010
Correct message is
1010010
arshdeep@arshdeep-HP-Laptop-14s-cr1xxx:~/Desktop/netcom$ ./ex1
choose a parity check to be demonstrated
1.even parity
2.odd parity
3.CRC
4.checksum
5.hamming
Enter the choice: 5
THIS IS HAMMING CODE FUNCTION
Enter 4 bits of data one by one
1 1 1 0

Encoded data is
1111000

Enter received data bits one by one
1 1 1 1 0 1 0

Error on position 2
Data sent : 1111000
Data received : 1111010
Correct message is
1111000
arshdeep@arshdeep-HP-Laptop-14s-cr1xxx:~/Desktop/netcom$
```

NETWORKS AND COMMUNICATION DA2(LAB)

DONE BY ARSHDEEP SINGH
19BCB0086

TOPIC : FLOW CONTROL MECHANISMS

STOP AND WAIT PROTOCOL

Code

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
   int i,j,noframes,x,x1=10,x2;
   for(i=0;i<200;i++)rand();
   scanf("%d",&noframes);
   i=1;j=1;
   printf("\n number of frames is %d",noframes);
   while(noframes>0){
         printf("\n sending frame %d",i);
         srand(x1++);
         x = rand()%10;
         if(x%2 == 0){
               for (x2=1; x2<2; x2++){
                  printf("\nwaiting for %d seconds\n", x2);
                  }
               printf("\nsending frame %d",i);
```

```
            srand(x1++);
            x = rand()%10;
            }
        printf("\n ack for frame %d",j);
        noframes-=1;i++;j++;
        }
   printf("\nend of stop and wait protocol");
}
```

OUTPUT

```
 number of frames is 4
 sending frame 1
 ack for frame 1
 sending frame 2
waiting for 1 seconds

sending frame 2
 ack for frame 2
 sending frame 3
 ack for frame 3
 sending frame 4
waiting for 1 seconds

sending frame 4
 ack for frame 4
end of stop and wait protocol
```

```
 number of frames is 7
 sending frame 1
 ack for frame 1
 sending frame 2
waiting for 1 seconds

sending frame 2
 ack for frame 2
 sending frame 3
 ack for frame 3
 sending frame 4
waiting for 1 seconds

sending frame 4
 ack for frame 4
 sending frame 5
waiting for 1 seconds

sending frame 5
 ack for frame 5
 sending frame 6
 ack for frame 6
 sending frame 7
waiting for 1 seconds

sending frame 7
 ack for frame 7
end of stop and wait protocol
```

```
 number of frames is 11
 sending frame 1
 ack for frame 1
 sending frame 2
waiting for 1 seconds

sending frame 2
 ack for frame 2
 sending frame 3
 ack for frame 3
 sending frame 4
waiting for 1 seconds

sending frame 4
 ack for frame 4
 sending frame 5
waiting for 1 seconds

sending frame 5
 ack for frame 5
 sending frame 6
 ack for frame 6
 sending frame 7
waiting for 1 seconds

sending frame 7
 ack for frame 7
 sending frame 8
 ack for frame 8
 sending frame 9
waiting for 1 seconds

sending frame 9
 ack for frame 9
 sending frame 10
waiting for 1 seconds

sending frame 10
 ack for frame 10

 ack for frame 11
end of stop and wait protocol
```

GO BACK N PROTOCOL

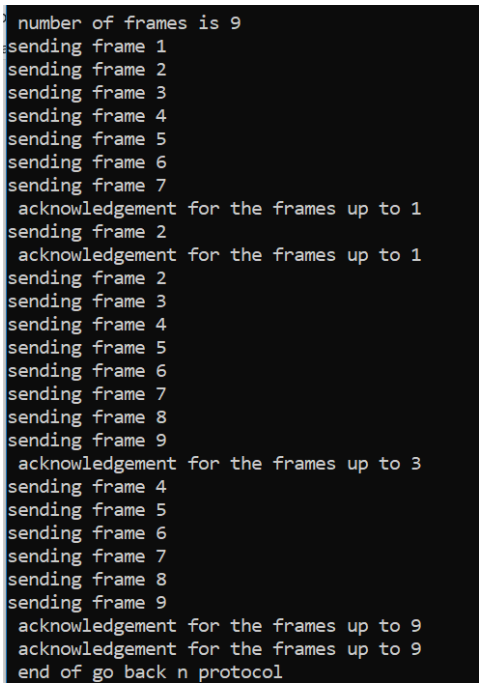
Code

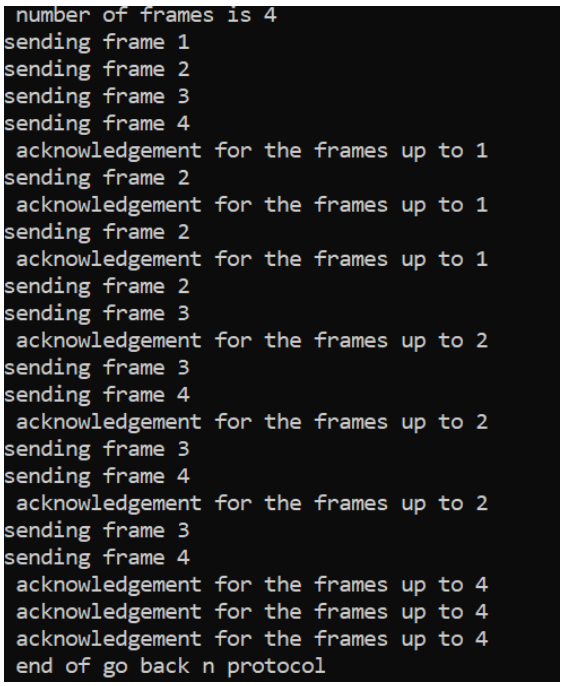
```
#include<stdio.h>


#include<stdlib.h>
int main()
{
   int temp1,temp2,temp3,temp4,i,winsize=8,noframes,moreframes;
   char c;
   int reciever(int);
   int simulate(int);


   temp4=0,temp1=0,temp2=0,temp3=0;
   for(i=0;i<200;i++)
   rand();
   scanf("%d",&noframes);


   printf("\n number of frames is %d",noframes);
   moreframes=noframes;
   while(moreframes>=0)
   {
      temp1=simulate(winsize);
      winsize-=temp1;
      temp4+=temp1;
      if(temp4 >noframes)
      temp4 = noframes;
```

```
        for(i=temp3+1;i<=temp4;i++)
            printf("\nsending frame %d",i);
        temp2=reciever(temp1);
        temp3+=temp2;
        if(temp3 > noframes)
        temp3 = noframes;
        printf("\n acknowledgement for the frames up to %d",temp3);
        moreframes-=temp2;
        temp4=temp3;
        if(winsize<=0)
        winsize=8;
    }
    printf("\n end of go back n protocol");
}

int reciever(int temp1)
{
    int i;
    for(i=1;i<100;i++)
        rand();
    i=rand()%temp1;
    return i;
}
int simulate(int winsize){
    int temp1,i;
    for(i=1;i<50;i++)
```

```
    temp1=rand();
  if(temp1==0)
    temp1=simulate(winsize);

  i = temp1%winsize;
  if(i==0)
    return winsize;
  else
    return temp1%winsize;
}
```

OUTPUT





```
 number of frames is 6
sending frame 1
sending frame 2
sending frame 3
sending frame 4
 acknowledgement for the frames up to 1
sending frame 2
 acknowledgement for the frames up to 1
sending frame 2
 acknowledgement for the frames up to 1
sending frame 2
sending frame 3
 acknowledgement for the frames up to 2
sending frame 3
sending frame 4
sending frame 5
sending frame 6
 acknowledgement for the frames up to 2
sending frame 3
sending frame 4
sending frame 5
 acknowledgement for the frames up to 2
sending frame 3
sending frame 4
sending frame 5
sending frame 6
 acknowledgement for the frames up to 4
sending frame 5
 acknowledgement for the frames up to 4
sending frame 5
sending frame 6
 acknowledgement for the frames up to 5
sending frame 6
 acknowledgement for the frames up to 5
sending frame 6
 acknowledgement for the frames up to 6
 acknowledgement for the frames up to 6
 end of go back n protocol
```

SELECTIVE REPEAT PROTOCOL

Code

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

int main(){
   int temp1,temp2,temp3,temp4,temp5,i,winsize=8,noframes,moreframes;
   char c;
   int reciever(int);
   int simulate(int);
   int nack(int);

   temp4=0,temp1=0,temp2=0,temp3=0,temp5 = 0;
   for(i=0;i<200;i++)
      rand();

   scanf("%d",&noframes);

   printf("\n number of frames is %d",noframes);

   moreframes=noframes;
   while(moreframes>=0){
         temp1=simulate(winsize);
         winsize-=temp1;
         temp4+=temp1;
         if(temp4 >noframes)
```

```
    temp4 = noframes;
  for(i=noframes - moreframes;i<=temp4;i++)
    printf("\nsending frame %d",i);

  temp2=reciever(temp1);
  temp3+=temp2;
  if(temp3 > noframes)
    temp3 = noframes;
    temp2 = nack(temp1);
    temp5+=temp2;
    if (temp5 !=0){
        printf("\n No acknowledgement for the frame %d",temp5);
        for(i=1;i<temp5;i++);
        printf("\n Retransmitting frame %d",temp5);

      }
      moreframes-=temp1;
      if(winsize<=0)
        winsize=8;
  }
  printf("\n end of sliding window protocol Selective Repeat");

}
```

```
int reciever(int temp1){
   int i;for(i=1;i<100;i++)rand();
   i=rand()%temp1;
   return i;
   }

int nack(int temp1){
   int i;
   for(i=1;i<100;i++)rand();
   i=rand()%temp1;
   return i;
   }

int simulate(int winsize){
   int temp1,i;
   for(i=1;i<50;i++)temp1=rand();
   if(temp1==0)temp1=simulate(winsize);
   i = temp1%winsize;
   if(i==0)return winsize;
   else
      return temp1%winsize;
   }
```

OUTPUT

```
 number of frames is 4
sending frame 0
sending frame 1
sending frame 2
sending frame 3
sending frame 4
 No acknowledgement for the frame 3
 Retransmitting frame 3
sending frame 4
 No acknowledgement for the frame 3
 Retransmitting frame 3
 end of sliding window protocol Selective Repeat
```

```
 number of frames is 7
sending frame 0
sending frame 1
sending frame 2
sending frame 3
sending frame 4
 No acknowledgement for the frame 3
 Retransmitting frame 3
sending frame 4
sending frame 5
 No acknowledgement for the frame 3
 Retransmitting frame 3
sending frame 5
sending frame 6
sending frame 7
 No acknowledgement for the frame 3
 Retransmitting frame 3
sending frame 7
 No acknowledgement for the frame 3
 Retransmitting frame 3
 end of sliding window protocol Selective Repeat
```

```
 number of frames is 25
sending frame 0
sending frame 1
sending frame 2
sending frame 3
sending frame 4
 No acknowledgement for the frame 3
 Retransmitting frame 3
sending frame 4
sending frame 5
 No acknowledgement for the frame 3
 Retransmitting frame 3
sending frame 5
sending frame 6
sending frame 7
 No acknowledgement for the frame 3
 Retransmitting frame 3
sending frame 7
sending frame 8
 No acknowledgement for the frame 3
 Retransmitting frame 3
sending frame 8
sending frame 9
sending frame 10
sending frame 11
sending frame 12
 No acknowledgement for the frame 6
 Retransmitting frame 6
sending frame 12
sending frame 13
sending frame 14
sending frame 15
 No acknowledgement for the frame 8
 Retransmitting frame 8
sending frame 15
sending frame 16
 No acknowledgement for the frame 8
 Retransmitting frame 8
sending frame 16
sending frame 17
sending frame 18
sending frame 19
sending frame 20
sending frame 21
sending frame 22
 No acknowledgement for the frame 13
 Retransmitting frame 13
sending frame 22
sending frame 23
sending frame 24
 No acknowledgement for the frame 14
 Retransmitting frame 14
sending frame 24
sending frame 25
 No acknowledgement for the frame 14
 Retransmitting frame 14
 end of sliding window protocol Selective Repeat
```

NETWORK AND COMMUNICATION LAB DA 3

19BCB0086

ARSHDEEP SINGH BHATIA

Q1

a) Identify the class of IP Address for the given binary notation.

b) Identify the class of IP Address for the given decimal notation.

c) Identify the default subnet mask for the given IP Address.

d) Identify the first address and last address for the given address.

A1

Code

```
#include <bits/stdc++.h>

using namespace std;

int dectofind(){
   cout<<"\nenter decimal ip address (dotted format)\n";
      string ip;
      cin>>ip;
      int pos=0;
      for(int i=0;i<ip.size();i++){
         if(ip[i]=='.'){
            pos=i;break;
         }
      }
      string bit=ip.substr(0,pos);
      int k=stoi(bit);
      if(k>=0 && k<=127)return 0;
      else if(k>=128 && k<=191)return 1;
      else if(k>=192 && k<=223)return 2;
      else if(k>=224 && k<=239)return 3;
      else if(k>=240 && k<=255)return 4;
      else return -1;
}

int main(){
   cout<<"=====================================================\n";
```

```
   cout<<"Welcome to network da 3"<<endl;
   cout<<" 1. Identify the class of IP Address for the given binary notation. \n 2. Identify the class of IP Address for the given decimal notation. \n 3. Identify the default subnet mask for the given IP Address. \n 4. Identify the first address and last address for the given address. \n 5. exit is -1 \n enter your choice : " ;
   int choice;
   cin>>choice;
   if (choice ==-1){
      exit(0);
   }
   if(choice==1){
      cout<<"\nenter binary ip address (dotted format)\n";
      string ip;
      cin>>ip;
      if (ip[0]=='0')cout<<"class A\n";
      else if(ip.substr(0,2)=="10")cout<<"Class B\n";
      else if (ip.substr(0,3)=="110")cout<<"Class C\n";
      else if (ip.substr(0,4)=="1110")cout<<"Class D\n";
      else if (ip.substr(0,4)=="1111")cout<<"Class E\n";
      else cout<<"INVALID INPUT";
   }
   if(choice==2){
      cout<<"\nenter decimal ip address (dotted format)\n";
      string ip;
      cin>>ip;
      int pos=0;
      for(int i=0;i<ip.size();i++){
         if(ip[i]=='.'){
            pos=i;break;
         }
      }
      string bit=ip.substr(0,pos);
      int k=stoi(bit);
      if(k>=0 && k<=127)cout<<"class A\n";
      else if(k>=128 && k<=191)cout<<"class B\n";
      else if(k>=192 && k<=223)cout<<"class C\n";
      else if(k>=224 && k<=239)cout<<"class D\n";
      else if(k>=240 && k<=255)cout<<"class E\n";
      else cout<<"INVALID INPUT\n";
   }
   if(choice==3){
      int k=dectofind();
      if (k==0)cout<<"\n default subnet mask = 255.0.0.0";
      else if (k==1)cout<<"\n default subnet mask = 255.255.0.0";
      else if (k==2)cout<<"\n default subnet mask = 255.255.255.0";
      else if (k==3)cout<<"\n default subnet mask is not defined for class d";
      else if (k==4)cout<<"\n default subnet mask is not defined for class e";
      else if (k==-1)cout<<"\n invaild input ";
```

```
    cout <<endl;
}
if(choice==4){
    cout<<"\nenter decimal ip address (dotted format)\n";
    string ip;
    cin>>ip;
    int pos;
    int q;
    for(int i=0;i<ip.size();i++){
        if(ip[i]=='.'){
            pos=i;
            break;
        }
    }
    string bit=ip.substr(0,pos);
    int k=stoi(bit);
    if(k>=0 && k<=127)q=0;
    else if(k>=128 && k<=191)q=1;
    else if(k>=192 && k<=223)q=2;
    else if(k>=224 && k<=239)q=3;
    else if(k>=240 && k<=255)q=4;
    else q=-1;
    if (q==0)
    {
        cout<<ip.substr(0,pos)<<".0.0.0 to "<<ip.substr(0,pos)<<".255.255.255 ";
    }
    else if (q==1){
        for(int i=pos+1;i<ip.size();i++){
            if(ip[i]=='.'){
                pos=i;
                break;
        }
    }
    cout<<ip.substr(0,pos)<<".0.0 to "<<ip.substr(0,pos)<<".255.255 ";
    }
    else if(q==2){
        for(int j=0;j<2;j++){
            for(int i=pos+1;i<ip.size();i++){
            if(ip[i]=='.'){
                pos=i;
                break;
                }
            }
        }
        cout<<ip.substr(0,pos)<<".0 to "<<ip.substr(0,pos)<<".255 \n";
    }

    else cout<<"Undefined for this \n";
```

```
    }

    main();
}
```

Output

Choice 1

```
=========================================================
Welcome to network da 3
 1. Identify the class of IP Address for the given binary notation.
 2. Identify the class of IP Address for the given decimal notation.
 3. Identify the default subnet mask for the given IP Address.
 4. Identify the first address and last address for the given address.
 5. exit is -1
 enter your choice : 1

enter binary ip address (dotted format)
00001010.00001011.00001100.00001101
class A
=========================================================
```

```
=========================================================
Welcome to network da 3
 1. Identify the class of IP Address for the given binary notation.
 2. Identify the class of IP Address for the given decimal notation.
 3. Identify the default subnet mask for the given IP Address.
 4. Identify the first address and last address for the given address.
 5. exit is -1
 enter your choice : 1

enter binary ip address (dotted format)
11010001.01101011.01101001.01000110
Class C
=========================================================
Welcome to network da 3
 1. Identify the class of IP Address for the given binary notation.
 2. Identify the class of IP Address for the given decimal notation.
 3. Identify the default subnet mask for the given IP Address.
 4. Identify the first address and last address for the given address.
 5. exit is -1
 enter your choice : 1

enter binary ip address (dotted format)
10011011.10101001.10101000.01101011
Class B
---------------------------------------------------------
```

Choice 2

```
=====================================================
Welcome to network da 3
 1. Identify the class of IP Address for the given binary notation.
 2. Identify the class of IP Address for the given decimal notation.
 3. Identify the default subnet mask for the given IP Address.
 4. Identify the first address and last address for the given address.
 5. exit is -1
 enter your choice : 2

enter decimal ip address (dotted format)
192.168.29.210
class C
=====================================================
Welcome to network da 3
 1. Identify the class of IP Address for the given binary notation.
 2. Identify the class of IP Address for the given decimal notation.
 3. Identify the default subnet mask for the given IP Address.
 4. Identify the first address and last address for the given address.
 5. exit is -1
 enter your choice : 2

enter decimal ip address (dotted format)
10.10.1.9
class A
=====================================================
```

Choice 3

```
=====================================================
Welcome to network da 3
 1. Identify the class of IP Address for the given binary notation.
 2. Identify the class of IP Address for the given decimal notation.
 3. Identify the default subnet mask for the given IP Address.
 4. Identify the first address and last address for the given address.
 5. exit is -1
 enter your choice : 3

enter decimal ip address (dotted format)
10.10.1.9

 default subnet mask = 255.0.0.0
=====================================================
Welcome to network da 3
 1. Identify the class of IP Address for the given binary notation.
 2. Identify the class of IP Address for the given decimal notation.
 3. Identify the default subnet mask for the given IP Address.
 4. Identify the first address and last address for the given address.
 5. exit is -1
 enter your choice : 3

enter decimal ip address (dotted format)
192.168.29.210

 default subnet mask = 255.255.255.0
=====================================================
```

Choice 4

```
==========================================================
Welcome to network da 3
 1. Identify the class of IP Address for the given binary notation.
 2. Identify the class of IP Address for the given decimal notation.
 3. Identify the default subnet mask for the given IP Address.
 4. Identify the first address and last address for the given address.
 5. exit is -1
 enter your choice : 4

enter decimal ip address (dotted format)
10.10.19.8
10.0.0.0 to 10.255.255.255 ==========================================================
Welcome to network da 3
 1. Identify the class of IP Address for the given binary notation.
 2. Identify the class of IP Address for the given decimal notation.
 3. Identify the default subnet mask for the given IP Address.
 4. Identify the first address and last address for the given address.
 5. exit is -1
 enter your choice : 4

enter decimal ip address (dotted format)
192.134.1.2
192.134.1.0 to 192.134.1.255

==========================================================
```

```
==========================================================
Welcome to network da 3
 1. Identify the class of IP Address for the given binary notation.
 2. Identify the class of IP Address for the given decimal notation.
 3. Identify the default subnet mask for the given IP Address.
 4. Identify the first address and last address for the given address.
 5. exit is -1
 enter your choice : 4

enter decimal ip address (dotted format)
192.168.11.2
192.168.11.0 to 192.168.11.255
==========================================================
Welcome to network da 3
 1. Identify the class of IP Address for the given binary notation.
 2. Identify the class of IP Address for the given decimal notation.
 3. Identify the default subnet mask for the given IP Address.
 4. Identify the first address and last address for the given address.
 5. exit is -1
 enter your choice : -1


...Program finished with exit code 0
Press ENTER to exit console.
```

Q2)

1. Subnetting
2. customized subnet mask
3. first and last address of each subnet
4. no of usable hosts

Code

```
#include <bits/stdc++.h>
using namespace std;

int getOctetsIP(string ip, vector<int> &octetsIP) {
	stringstream sip(ip);
	string temp;
	octetsIP.clear();
	vector<bool> ipInRange;
	while (getline(sip,temp,'.'))
		octetsIP.push_back(atoi(temp.c_str()));
	if (octetsIP.size() == 4) {
		for(int i = 0; i < octetsIP.size(); i++){
			if (octetsIP[i] >= 0 && octetsIP[i] <= 255)
				ipInRange.push_back(true);
			else
				ipInRange.push_back(false);
		}
		if (ipInRange[0]==true&&ipInRange[1]==true&&ipInRange[2]==true&&ipInRange[3]==true){
			return 0;
		}else{
			cout << endl << "There are only 255 bits per octet. Please re-enter IP." << endl << endl;
			return 1;
		}
	}else{
		cout << endl << "Please enter four octets in dot notation." << endl << endl;
		return 1;
	}
}



int getOctetsMask(string mask,  vector<int> &octetsMask) {
	stringstream smask(mask);
	string temp;
	octetsMask.clear();
	vector<bool> maskInRange;
	while (getline(smask,temp,'.'))
		octetsMask.push_back(atoi(temp.c_str()));
	if (octetsMask.size() == 4){
		for(int i = 0; i < octetsMask.size(); i++){
			if (octetsMask[i] == 0 || octetsMask[i] == 128 || octetsMask[i] == 192 || octetsMask[i] ==
224 || octetsMask[i] == 240 || octetsMask[i] == 248 || octetsMask[i] == 252 || octetsMask[i] == 254 || octetsMask[i] ==
255)
				maskInRange.push_back(true);
			else
				maskInRange.push_back(false);
		}

	if(maskInRange[0]==true&&maskInRange[1]==true&&maskInRange[2]==true&&maskInRange[3]==true){
```

```
                        return 0;
                }else{
                        cout << endl << "Subnet masks only use 2^[0-7]. Please re-enter mask." << endl << endl;
                        return 1;
                }
        }else{
                cout << endl << "Please enter four octets in dot notation." << endl << endl;
                return 1;
        }
}



int calcClass(vector<int> &octetsIP) {
        if (octetsIP[0] == 10) {
                return 1;
        }else if (octetsIP[0] == 172 && octetsIP[1] >= 16 && octetsIP[1] <= 31) {
                return 2;
        }else if (octetsIP[0] == 192 && octetsIP[1] == 168) {
                return 3;
        }else if (octetsIP[0] == 127) {
                return 4;
        }else if (octetsIP[0] >= 0 && octetsIP[0] < 127) {
                return 5;
        }else if (octetsIP[0] > 127 && octetsIP[0] < 192) {
                return 6;
        }else if (octetsIP[0] > 191 && octetsIP[0] < 224) {
                return 7;
        }else if (octetsIP[0] > 223 && octetsIP[0] < 240) {
                return 8;
        }else if (octetsIP[0] > 239 && octetsIP[0] <= 255) {
                return 9;
        }else{
                return 0;
        }
}



int getNHBits(vector<int> &octetsIP, vector<int> &octetsMask, vector<int> &octetsIPBits, vector<int> &octetsMaskBits){


cout << "-----------------------------------------" << endl;
cout << "===== Binary Representation =====/" << endl;
cout << "-----------------------------------------" << endl;
        for (int j=0; j < octetsIP.size(); j++)
   {
                if (j>0)
                        cout << ".";

      int mask = 128;
      while (mask)
      {
         octetsIPBits.push_back((octetsIP[j] & mask) != 0);
                        cout << ((octetsIP[j] & mask) != 0);
         mask >>= 1;
      }
```

```
  }
        cout << "  : IP Address" << endl;
        for (int j=0; j < octetsMask.size(); j++)
  {
                if (j>0)
                        cout << ".";
    int mask = 128;
    while (mask)
    {
      octetsMaskBits.push_back((octetsMask[j] & mask) != 0);
                        cout << ((octetsMask[j] & mask) != 0);
      mask >>= 1;
    }
  }
        cout << "  : Subnet Mask" << endl;
        cout << "----------------------------------------" << endl;

return 0;
}


vector<int> getNetID(vector<int> &octetsIPBits, vector<int> &octetsMaskBits){
        vector<int> netID;
  for (int j=0; j < octetsIPBits.size(); j++)
  {
    if ((j > 0) && (j%8 == 0))
      cout << ".";

                netID.push_back(octetsIPBits[j] & octetsMaskBits[j]);
  }
return netID;
}
string toString(vector<int> octets){
        ostringstream octStrm;

        for(int j = 0; j < octets.size(); j++)
        {
                if (j>0)
                        octStrm << '.';

                octStrm << octets[j];
        }

        return octStrm.str();
}

vector<int> toDecimal(vector<int> octets, vector<int> &decimals){
        stringstream octStrm;
        decimals.clear();
        for(int j = 0; j < octets.size(); j++)
        {
                if (j>0)
                        octStrm << '.';

                octStrm << octets[j];
        }

        string temp;
```

```
	while (getline(octStrm, temp, '.'))
		decimals.push_back(atoi(temp.c_str()));

	return decimals;
}
int getIncrement(vector<int> decimalMask, vector<int> decimalNetID){
	int increment = 0;
	for (int i=0; i<decimalMask.size(); i++){
		if (decimalMask[i] == 255){
			increment = 1;
		}else if(decimalMask[i] == 254){
			increment = 2;
			break;
		}else if(decimalMask[i] == 252){
			increment = 4;
			break;
		}else if(decimalMask[i] == 248){
			increment = 8;
			break;
		}else if(decimalMask[i] == 240){
			increment = 16;
			break;
		}else if(decimalMask[i] == 224){
			increment = 32;
			break;
		}else if(decimalMask[i] == 192){
			increment = 64;
			break;
		}else if(decimalMask[i] == 128){
			increment = 128;
			break;
		}
	}
return increment;
}

vector<int> getNetIDRange(vector<int> &decimalNetID, int &netInc, vector<int> &decimalMask) {
	vector<int> netIDEnd;
	for (int i=0; i<decimalNetID.size(); i++){
		if (decimalMask[i] == 255){
			netIDEnd.push_back(decimalNetID[i]);
		}else if (decimalMask[i] < 255 && decimalMask[i] > 0){
			netIDEnd.push_back( (decimalNetID[i] + netInc) - 1 );
		}else{
			netIDEnd.push_back(255);
		}
	}
	return netIDEnd;
}
int getSubnets(vector<int> &decimalMask, int &ipClass, vector<int> &subClassMask){
	int netBits = 0;
	subClassMask.clear();
		if (ipClass==1){
			subClassMask.push_back(255);
			subClassMask.push_back(0);
			subClassMask.push_back(0);
			subClassMask.push_back(0);
		}else if(ipClass==2){
```

```
                        subClassMask.push_back(255);
                        subClassMask.push_back(255);
                        subClassMask.push_back(0);
                        subClassMask.push_back(0);
                }else if(ipClass==3){
                        subClassMask.push_back(255);
                        subClassMask.push_back(255);
                        subClassMask.push_back(255);
                        subClassMask.push_back(0);
                }else if(ipClass==4 || ipClass==5){
                        subClassMask.push_back(decimalMask[0]);
                        subClassMask.push_back(decimalMask[1]);
                        subClassMask.push_back(decimalMask[2]);
                        subClassMask.push_back(decimalMask[3]);
                }

        for (int i=0; i<decimalMask.size(); i++){
                if (decimalMask[i] != subClassMask[i]){
                        if (decimalMask[i] == 255){
                                netBits += 8;
                                continue;
                        }else if (decimalMask[i] == 254){
                                netBits += 7;
                                continue;
                        }else if (decimalMask[i] == 252){
                                netBits += 6;
                                continue;
                        }else if (decimalMask[i] == 248){
                                netBits += 5;
                                continue;
                        }else if (decimalMask[i] == 240){
                                netBits += 4;
                                continue;
                        }else if (decimalMask[i] == 224){
                                netBits += 3;
                                continue;
                        }else if (decimalMask[i] == 192){
                                netBits += 2;
                                continue;
                        }else if (decimalMask[i] == 128){
                                netBits += 1;
                                continue;
                        }else if (decimalMask[i] == 0){
                                netBits += 0;
                                continue;
                        }else{
                                netBits += 0;
                        }
                }
        }
        int subnets = pow(2.0,netBits);
        return subnets;
}

int getHostsPerSubnet(vector<int> &decimalMask){
        int hostBits = 0;
        for (int i=0; i<decimalMask.size(); i++){
                if (decimalMask[i] == 255){
```

```
                hostBits += 0;
                continue;
            }else if (decimalMask[i] == 254){
                hostBits += 1;
                continue;
            }else if (decimalMask[i] == 252){
                hostBits += 2;
                continue;
            }else if (decimalMask[i] == 248){
                hostBits += 3;
                continue;
            }else if (decimalMask[i] == 240){
                hostBits += 4;
                continue;
            }else if (decimalMask[i] == 224){
                hostBits += 5;
                continue;
            }else if (decimalMask[i] == 192){
                hostBits += 6;
                continue;
            }else if (decimalMask[i] == 128){
                hostBits += 7;
                continue;
            }else if (decimalMask[i] == 0){
                hostBits += 8;
                continue;
            }else{
                hostBits = 0;
                break;
            }
        }
        int hostsPerSubnet = pow(2.0,hostBits)-2;
        return hostsPerSubnet;
}


int main() {
char resp = 'y';
while (resp == 'y') {
        cout << endl << endl;
            string ip;
            vector<int> octetsIP;
            while (getOctetsIP(ip, octetsIP) == 1) {
            cout << "Enter IPv4 Address -> ";
            (getline(cin, ip));
            }
            string mask;
            vector<int> octetsMask;
            while (getOctetsMask(mask, octetsMask) == 1) {
            cout << endl << "Enter subnet mask for " << ip << " -> ";
            (getline(cin, mask));
            }
            cout << endl << endl << endl << endl << endl;
            vector<int> decimals;
            cout << "======================" << endl;
            cout << "=== IP Address: " << toString(octetsIP) << endl;
            vector<int> decimalMask = toDecimal(octetsMask, decimals);
            cout << "=== Subnet Mask: " << toString(octetsMask) << endl;
```

```
cout << "=======================" << endl << endl;
vector<int> octetsIPBits;
vector<int> octetsMaskBits;
getNHBits(octetsIP, octetsMask, octetsIPBits, octetsMaskBits);
vector<int> netID = getNetID(octetsIP, octetsMask);
vector<int> decimalNetID = toDecimal(netID, decimals);
int netInc = getIncrement(decimalMask, decimalNetID);
cout << endl;
        cout << "----------------------------------------" << endl;
        cout << "======== Class Information =======" << endl;
        cout << "----------------------------------------" << endl;
        int classResult = calcClass(octetsIP);
        int ipClass = 0;
        switch (classResult){
                case 1:
                        cout << "IP Class: Private block, Class 'A' " << endl;
                        ipClass = 1;
                        break;
                case 2:
                        cout << "IP Class: Private block, Class 'B'" << endl;
                        ipClass = 2;
                        break;
                case 3:
                        cout << "IP Class: Private block, Class 'C'" << endl;
                        ipClass = 3;
                        break;
                case 4:
                        cout << "IP Class: Reserved block, System Loopback Address" << endl;
                        ipClass = 1;
                        break;
                case 5:
                        cout << "IP Class: A" << endl;
                        ipClass = 1;
                        break;
                case 6:
                        cout << "IP Class: B" << endl;
                        ipClass = 2;
                        break;
                case 7:
                        cout << "IP Class: C" << endl;
                        ipClass = 3;
                        break;
                case 8:
                        cout << "IP Class: D" << endl;
                        ipClass = 4;
                        cout << "!! This is a reserved Class D Multicast IP Address Block" << endl;
                        break;
                case 9:
                        cout << "IP Class: E" << endl;
                        ipClass = 5;
                        cout << "!! This is a reserved Class E Multicast IP Address Block" << endl;
                        break;
                default :
                        cout << "Not in Range" << endl;
                        break;
        }
vector<int> subClassMask;
getSubnets(decimalMask, ipClass, subClassMask);
```

```
                cout << "Default Class Subnet Mask: " << toString(subClassMask) << endl;
                cout << "-----------------------------------------" << endl << endl;

                cout << "-----------------------------------------" << endl;
                cout << "======== Subnet Details ========" << endl;
                cout << "-----------------------------------------" << endl;
                vector<int> netIDRange = getNetIDRange(decimalNetID, netInc, decimalMask);
                cout << "Network ID:          -          Broadcast ID: " << endl;
                        cout << "-------------------------------------------------" << endl;
                        cout << toString(netID) << " - [ usable hosts ] - ";
                cout << toString(netIDRange) << endl << endl;
                cout << "Network Increment: " << getIncrement(decimalMask, decimalNetID) << endl;
                cout << "Number of Subnets: " << getSubnets(decimalMask, ipClass, subClassMask) << endl;
                cout << "Usable hosts per subnet: " << getHostsPerSubnet(decimalMask) << endl;
                cout << "-----------------------------------------" << endl << endl;

                cout << "Would you like to enter another IP Address to subnet? (y or n): ";
                cin >> resp;
                cout << endl << endl << endl << endl;
}
        return 0;
}
```

Output

```
=======================
=== IP Address: 192.168.1.10
=== Subnet Mask: 255.255.255.248
=======================

----------------------------------------
===== Binary Representation =====/
----------------------------------------
11000000.10101000.00000001.00001010  : IP Address
11111111.11111111.11111111.11111000  : Subnet Mask
----------------------------------------

----------------------------------------
======== Class Information =======
----------------------------------------
IP Class: Private block, Class 'C'
Default Class Subnet Mask: 255.255.255.0
----------------------------------------

----------------------------------------
======== Subnet Details ========
----------------------------------------
Network ID:           -           Broadcast ID:
-------------------------------------------------
192.168.1.8 - [ usable hosts ] - 192.168.1.15

Network Increment: 8
Number of Subnets: 32
Usable hosts per subnet: 6
----------------------------------------
```

Output 2

```
======================
=== IP Address: 10.12.20.5
=== Subnet Mask: 255.248.0.0
======================

----------------------------------------
===== Binary Representation =====/
----------------------------------------
00001010.00001100.00010100.00000101  : IP Address
11111111.11111000.00000000.00000000  : Subnet Mask
----------------------------------------

----------------------------------------
======== Class Information =======
----------------------------------------
IP Class: Private block, Class 'A'
Default Class Subnet Mask: 255.0.0.0
----------------------------------------

----------------------------------------
======== Subnet Details ========
----------------------------------------
Network ID:           -          Broadcast ID:
------------------------------------------------
10.8.0.0 - [ usable hosts ] - 10.15.255.255

Network Increment: 8
Number of Subnets: 32
Usable hosts per subnet: 524286
----------------------------------------
```

Arshdeep Singh 19BCB0086
Network Lab Midterm

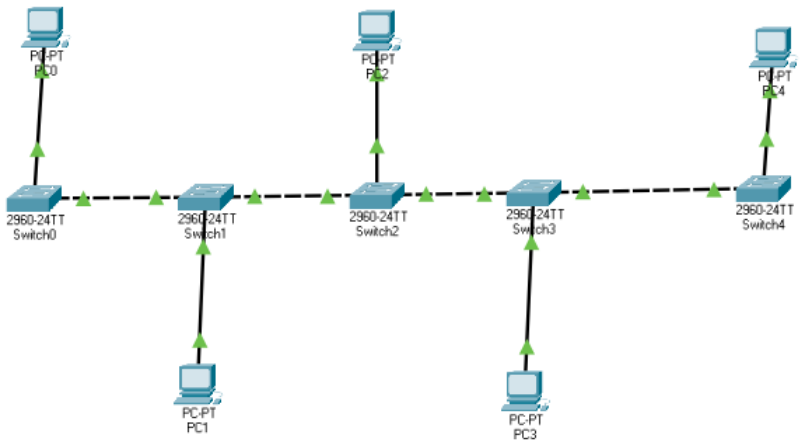
DATE 1-JUNE-2021

Questions

1.Cisco packet tracer Simulate the following bus topology, where the source node sends the FTP application over TCP agent through the intermediate nodes, the bandwidth and delay is given. Generate the trace file to analyse the packet dropped.

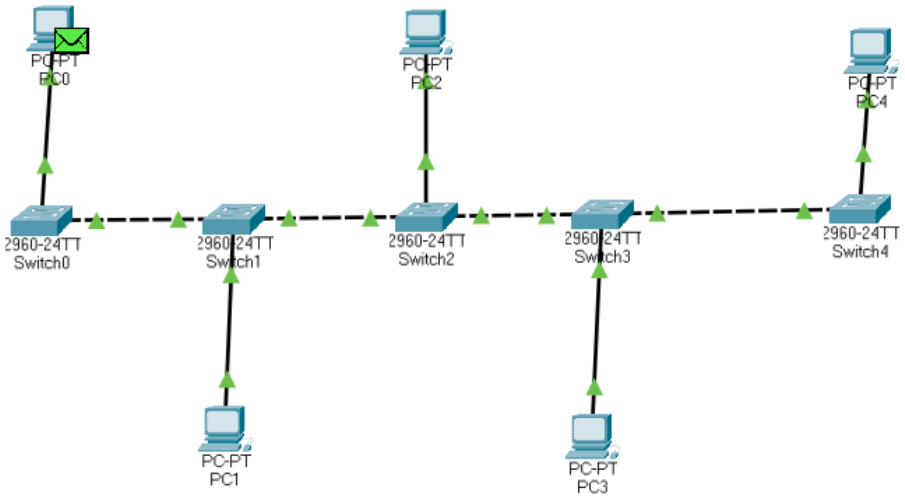
Answer

In the question nothing is given properly so we **assume** a random bus topology

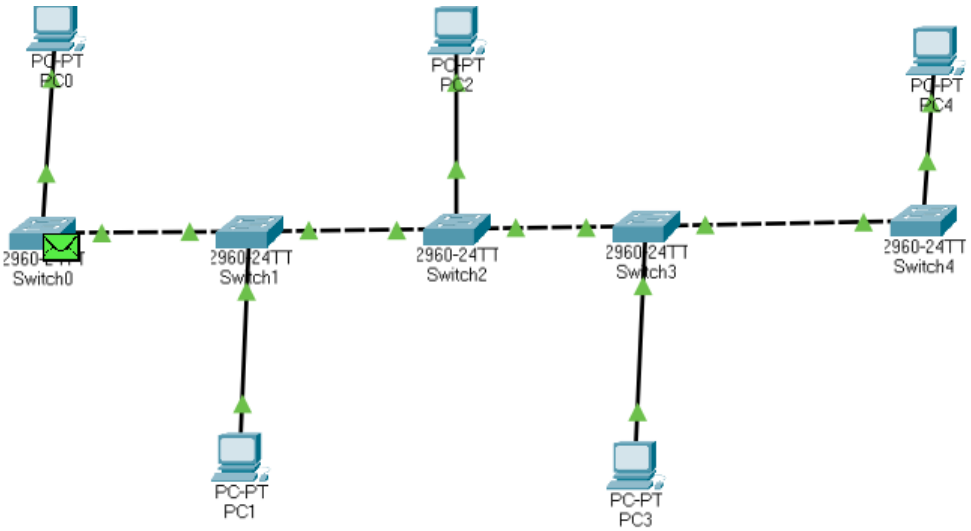


Simulation of a file transfer

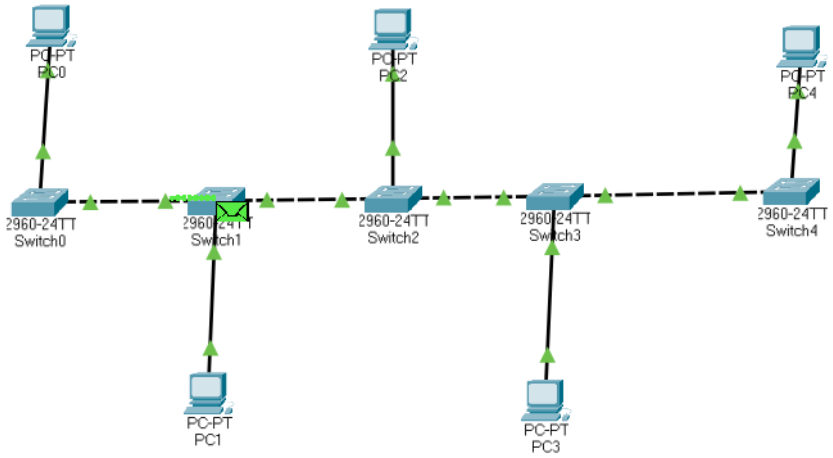
1



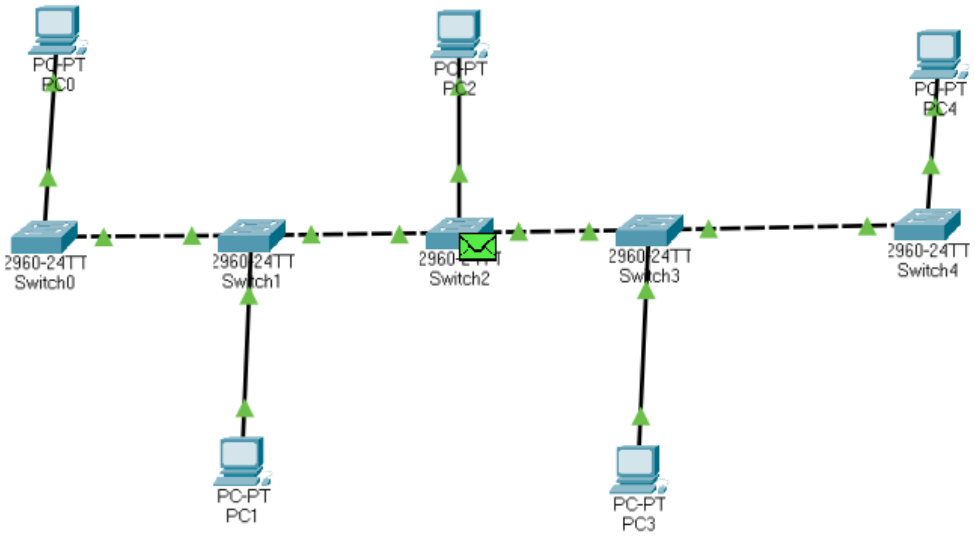
2



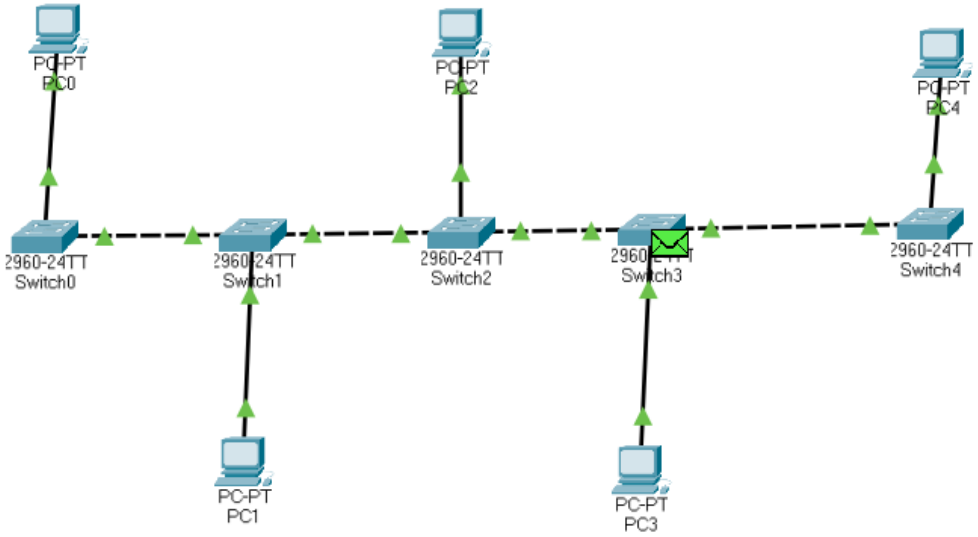
3



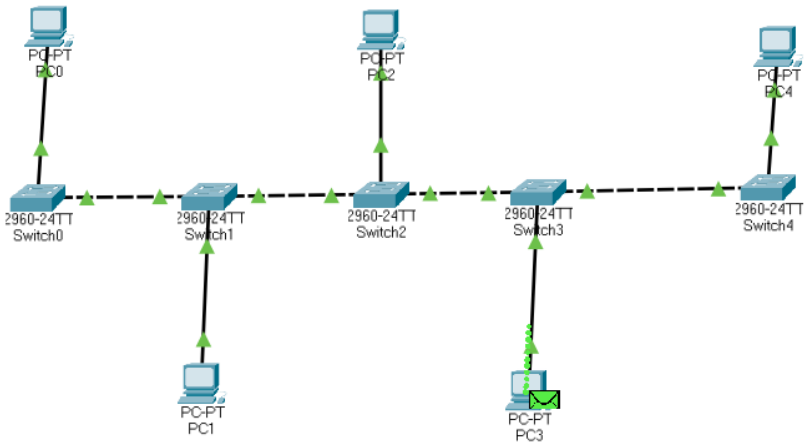
4


PC-PT
PC0
PC-PT
PC2
PC-PT
PC4
2960-24TT
Switch0
2960-24TT
Switch1
2960-24TT
Switch2
2960-24TT
Switch3
2960-24TT
Switch4
PC-PT
PC1
PC-PT
PC3

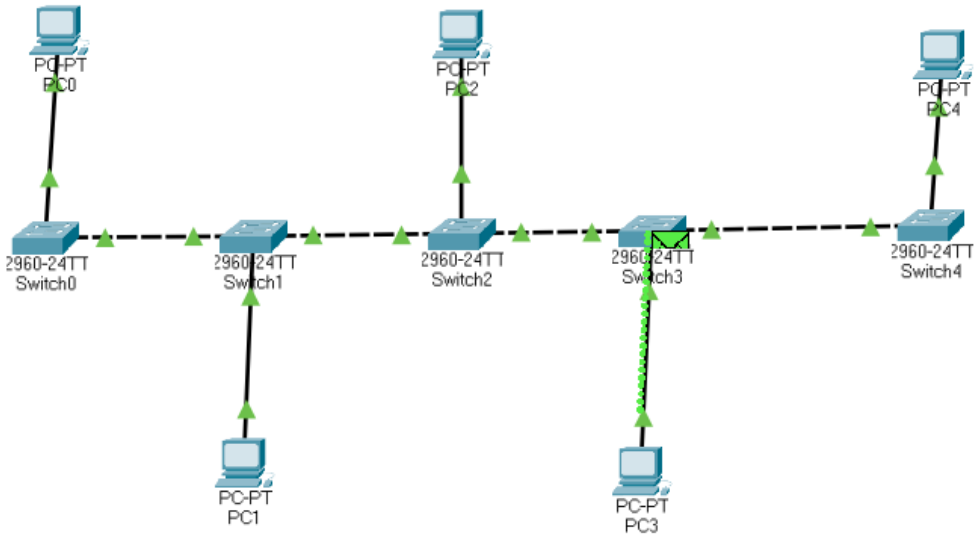
5.


PC-PT
PC0
PC-PT
PC2
PC-PT
PC4
2960-24TT
Switch0
2960-24TT
Switch1
2960-24TT
Switch2
2960-24TT
Switch3
2960-24TT
Switch4
PC-PT
PC1
PC-PT
PC3

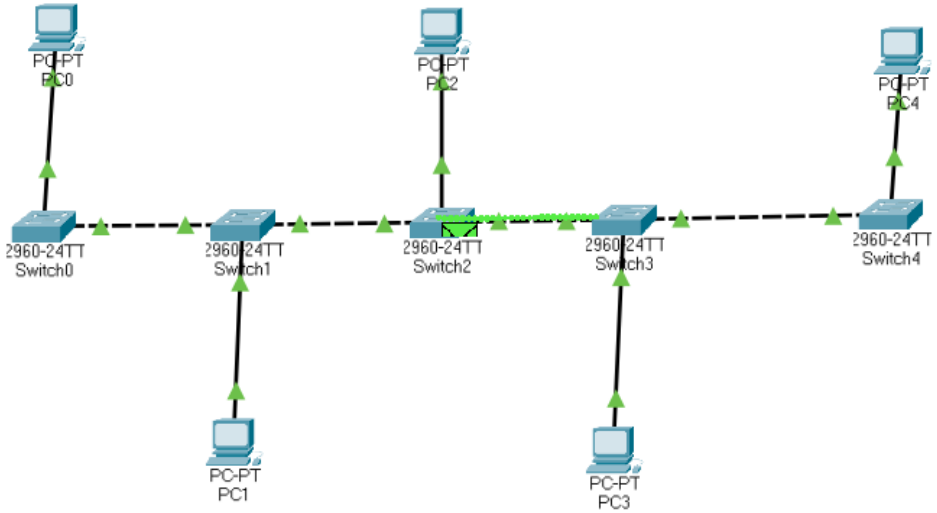
6.


PC-PT
PC0
PC-PT
PC2
PC-PT
PC4
2960-24TT
Switch0
2960-24TT
Switch1
2960-24TT
Switch2
2960-24TT
Switch3
2960-24TT
Switch4
PC-PT
PC1
PC-PT
PC3

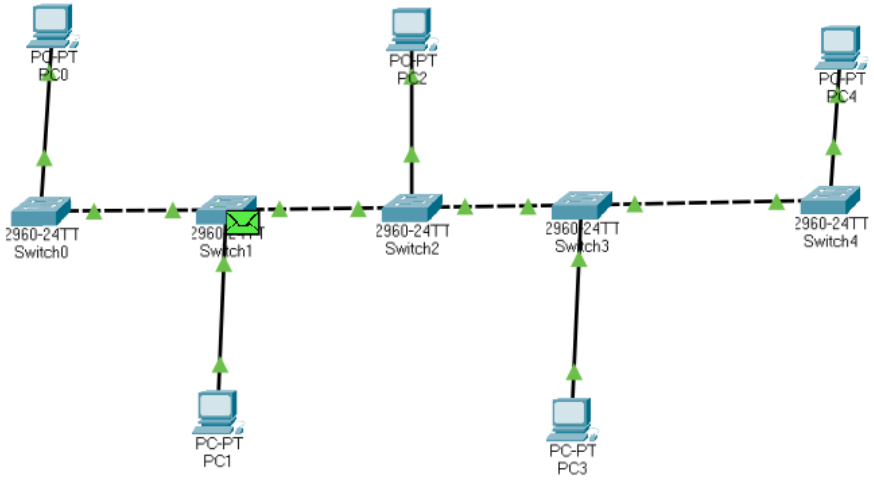
7


PC-PT PC0
PC-PT PC2
PC-PT PC4
2960-24TT Switch0
2960-24TT Switch1
2960-24TT Switch2
2960-24TT Switch3
2960-24TT Switch4
PC-PT PC1
PC-PT PC3

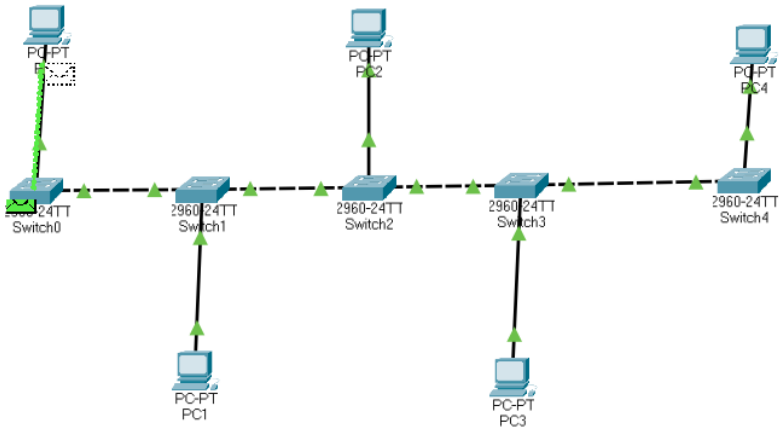
8.


PC-PT PC0
PC-PT PC2
PC-PT PC4
2960-24TT Switch0
2960-24TT Switch1
2960-24TT Switch2
2960-24TT Switch3
2960-24TT Switch4
PC-PT PC1
PC-PT PC3

9

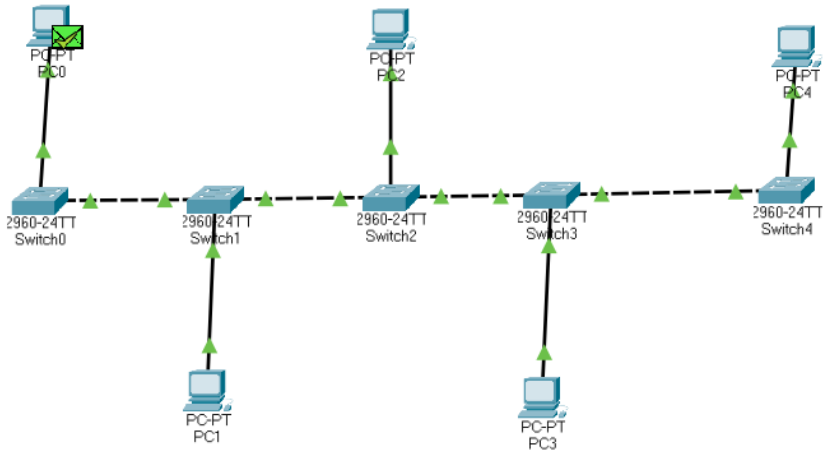

PC-PT PC0
PC-PT PC2
PC-PT PC4
2960-24TT Switch0
2960-24TT Switch1
2960-24TT Switch2
2960-24TT Switch3
2960-24TT Switch4
PC-PT PC1
PC-PT PC3

10



11

file transfer complete with green tick



12

Time(sec)	Last Device	At Device	Type
.001	PC0	Switch0	TCP
.001	–	PC0	TCP
.002	PC0	Switch0	TCP
.002	Switch0	Switch1	TCP
.002	–	PC0	TCP
.003	PC0	Switch0	TCP
.003	Switch0	Switch1	TCP
.003	Switch1	Switch2	TCP
.004	Switch0	Switch1	TCP
.004	Switch1	Switch2	TCP
.004	Switch2	Switch3	TCP
.005	Switch1	Switch2	TCP
.005	Switch2	Switch3	TCP

Packet Tracing

```
Packet Tracer PC Command Line 1.0
C:\>tracert 192.168.1.4

Tracing route to 192.168.1.4 over a maximum of 30 hops:

  1    0 ms      0 ms      0 ms      192.168.1.4

Trace complete.

C:\>tracert 192.168.1.2

Tracing route to 192.168.1.2 over a maximum of 30 hops:

  1    1 ms      0 ms      0 ms      192.168.1.2

Trace complete.

C:\>tracert 192.168.1.3

Tracing route to 192.168.1.3 over a maximum of 30 hops:

  1    0 ms      0 ms      0 ms      192.168.1.3

Trace complete.

C:\>tracert 192.168.1.5

Tracing route to 192.168.1.5 over a maximum of 30 hops:

  1    13 ms     13 ms     0 ms      192.168.1.5

Trace complete.

C:\>tracert 192.168.1.6

Tracing route to 192.168.1.6 over a maximum of 30 hops:

  1    *         *         *         Request timed out.
```

Since 192.168.1.6 is invalid request is timed out

2. Write a program to calculate the LRC to given bits, execute the functions with and without errors 11100111 11011101 00111001 10101001
Answer

Code:

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int a1[8];
    int a2[8];
    int a3[8];
    int a4[8];
    int a5[8];
    int a1_[8];
    int a2_[8];
    int a3_[8];
    int a4_[8];
    int a5_[8];

    cout<<"enter data to send\n";

    for(int i=0;i<8;i++){
        cin>>a1[i];
    }
    for(int i=0;i<8;i++){
        cin>>a2[i];
    }
    for(int i=0;i<8;i++){
        cin>>a3[i];
    }
    for(int i=0;i<8;i++){
        cin>>a4[i];
    }

    for(int i=0;i<8;i++){
        int sum =a1[i]+a2[i]+a3[i]+a4[i];
        a5[i]=sum%2;
    }
    cout<<"the data to send is :";
    for(int i=0;i<8;i++){
        cout<<a1[i];
    }
    cout<<" ";
    for(int i=0;i<8;i++){
        cout<<a2[i];
    }
    cout<<" ";
    for(int i=0;i<8;i++){
        cout<<a3[i];
    }
    cout<<" ";
    for(int i=0;i<8;i++){
        cout<<a4[i];
```

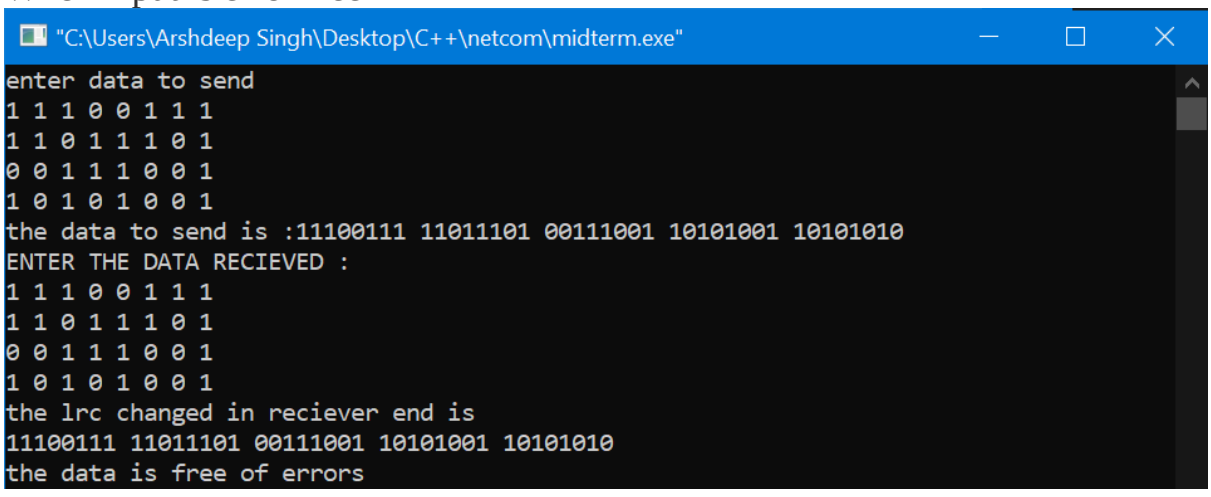
```
}
cout<<" ";
for(int i=0;i<8;i++){
   cout<<a5[i];
}
cout<<"\n";

cout<<"ENTER THE DATA RECIEVED :\n";
 for(int i=0;i<8;i++){
   cin>>a1_[i];
}
for(int i=0;i<8;i++){
   cin>>a2_[i];
}
for(int i=0;i<8;i++){
   cin>>a3_[i];
}
for(int i=0;i<8;i++){
   cin>>a4_[i];
}
for(int i=0;i<8;i++){
   int sum =a1_[i]+a2_[i]+a3_[i]+a4_[i];
   a5_[i]=sum%2;
}
cout<<"the lrc changed in reciever end is \n";
for(int i=0;i<8;i++){
   cout<<a1_[i];
}
cout<<" ";
for(int i=0;i<8;i++){
   cout<<a2_[i];
}
cout<<" ";
for(int i=0;i<8;i++){
   cout<<a3_[i];
}
cout<<" ";
for(int i=0;i<8;i++){
   cout<<a4_[i];
}
cout<<" ";
for(int i=0;i<8;i++){
   cout<<a5_[i];
}
cout<<"\n";
bool k=true;
for(int i=0;i<8;i++){
   if(a1[i]!=a1_[i]){
      k=false;
   }
}
for(int i=0;i<8;i++){
   if(a2[i]!=a2_[i]){
      k=false;
   }
```

```
    }
    for(int i=0;i<8;i++){
        if(a3_[i]!=a3[i]){
            k=false;
        }
    }
    for(int i=0;i<8;i++){
        if(a4[i]!=a4_[i]){
            k=false;
        }
    }
    for(int i=0;i<8;i++){
        if(a5[i]!=a5_[i]){
            k=false;
        }
    }

    if (k){
        cout<<"the data is free of errors";
    }
    else{
        cout<<"errors detected in the reciever end";

    }
}
```

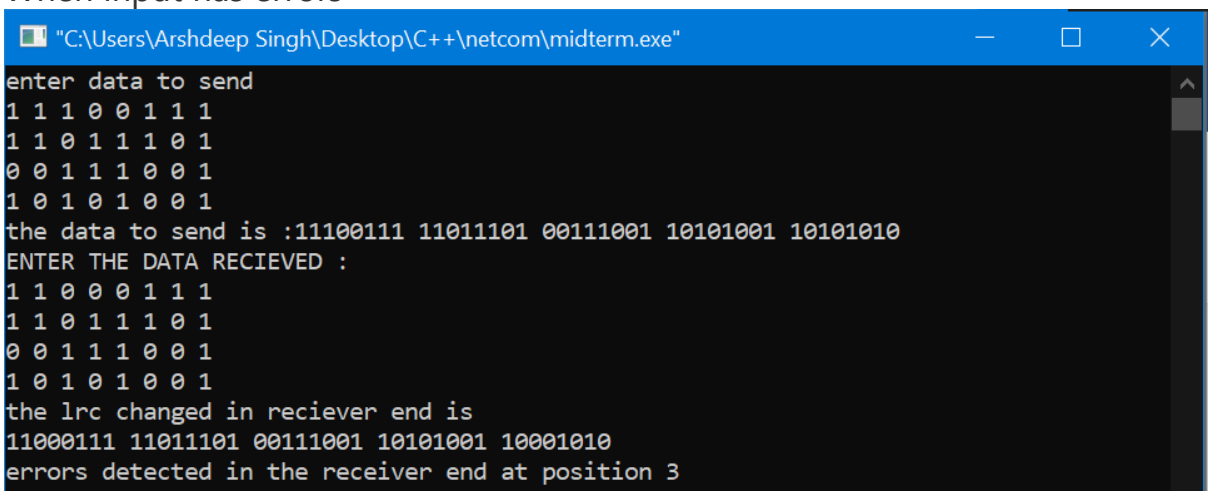
Output

(error message is last line)

When input is error free



When input has errors



Network Lab DA 4

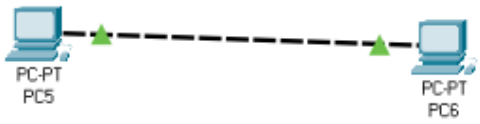
Arshdeep Singh Bhatia

19BCB0086

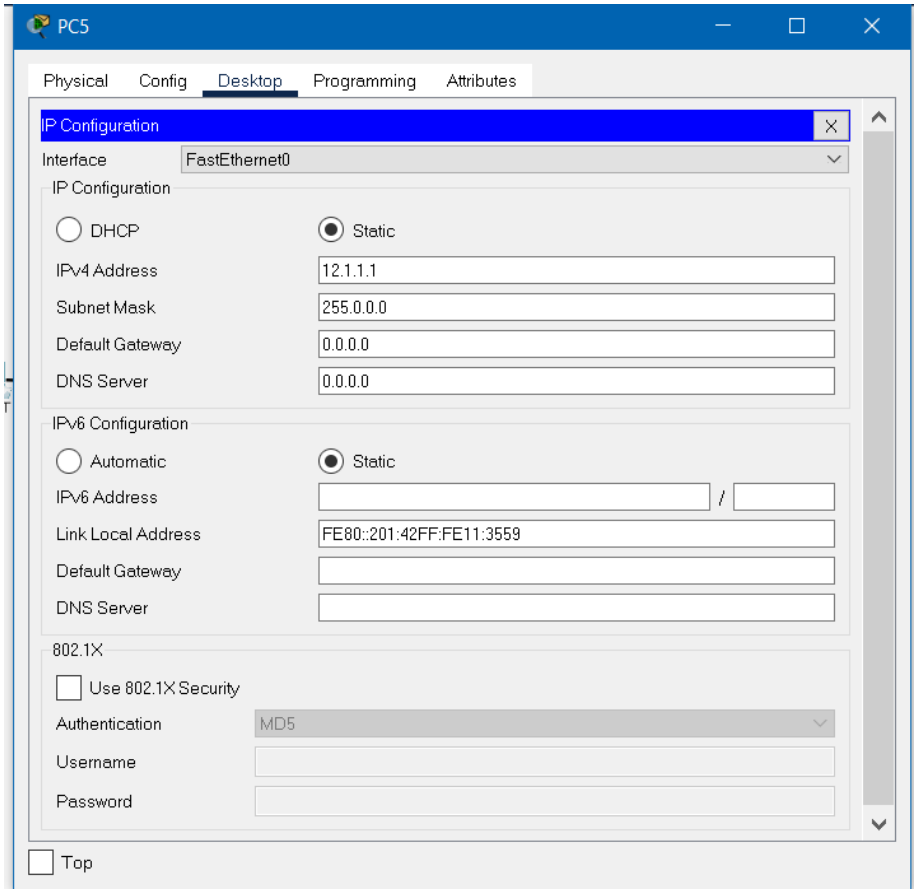
Cisco Packet tracer

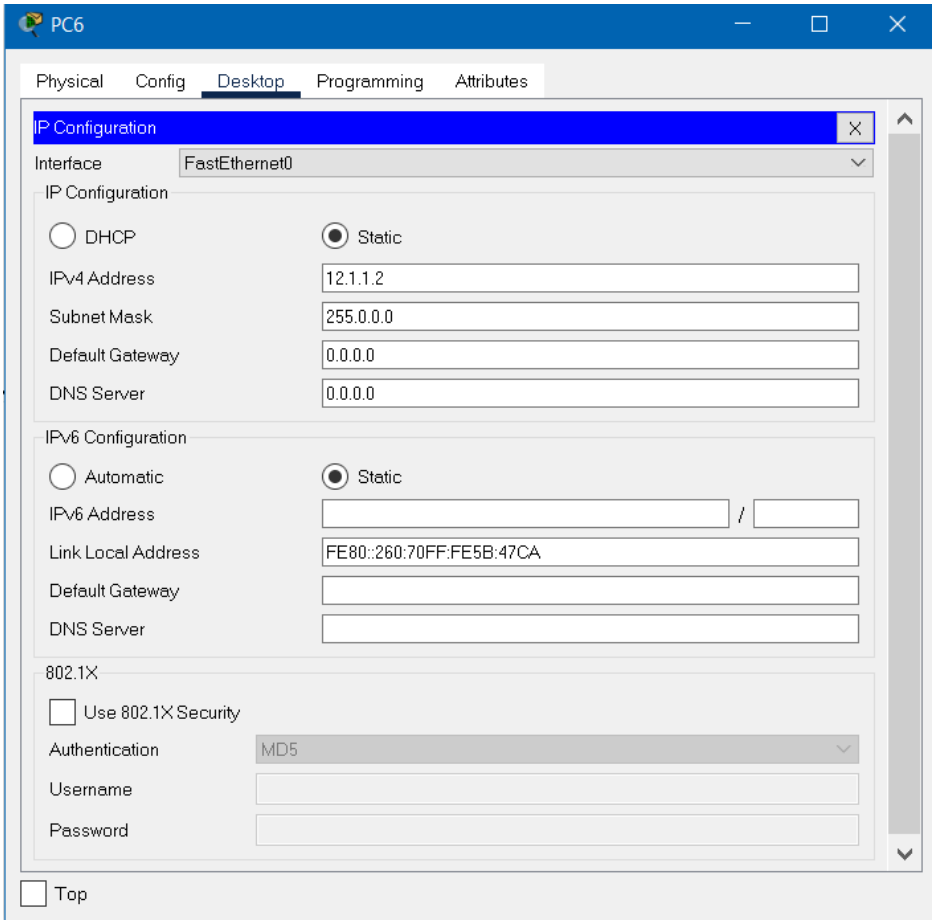
1. P2p Connection

1. Image



2.ip config of pc 1





3.terminal ping commands

Ipconfig

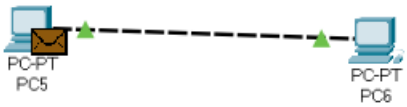
```
C:\>ipconfig

FastEthernet0 Connection:(default port)

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.........:
FE80::201:42FF:FE11:3559
   IPv6 Address....................: ::
   IPv4 Address....................: 12.1.1.1
   Subnet Mask.....................: 255.0.0.0
   Default Gateway.................: ::
                                     0.0.0.0

Bluetooth Connection:

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.........: ::
   IPv6 Address....................: ::
   IPv4 Address....................: 0.0.0.0
   Subnet Mask.....................: 0.0.0.0
   Default Gateway.................: ::
                                     0.0.0.0
```

Ping 12.1.1.2

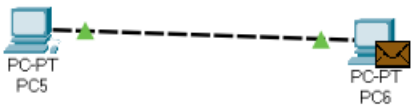
```
C:\>ping 12.1.1.2

Pinging 12.1.1.2 with 32 bytes of data:

Reply from 12.1.1.2: bytes=32 time=114ms TTL=128
Reply from 12.1.1.2: bytes=32 time<1ms TTL=128
Reply from 12.1.1.2: bytes=32 time<1ms TTL=128
Reply from 12.1.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 12.1.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0%
loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 114ms, Average = 28ms
```

4.Simulation

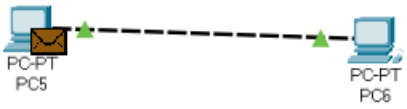
1



2

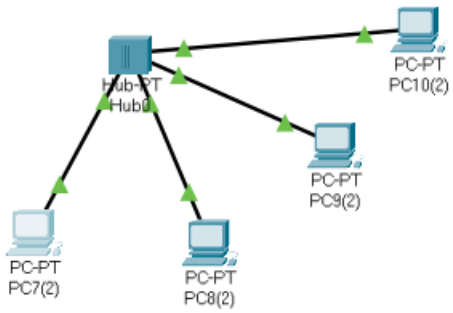


3 ack



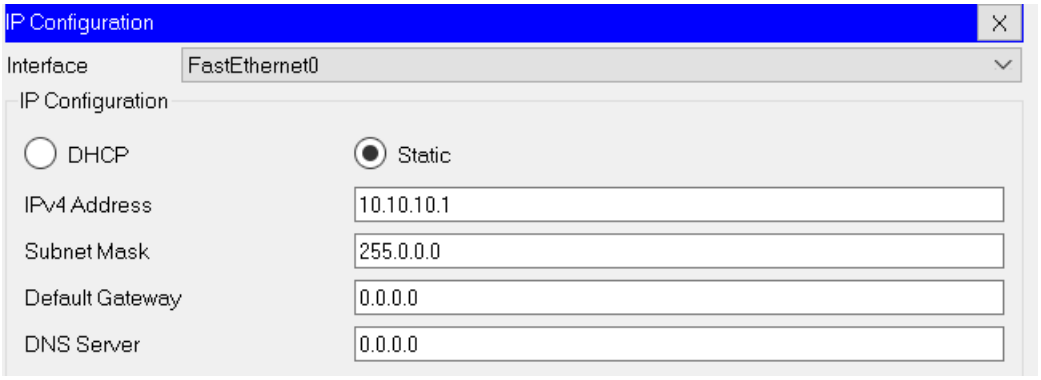
2. Hub Connection

1.image

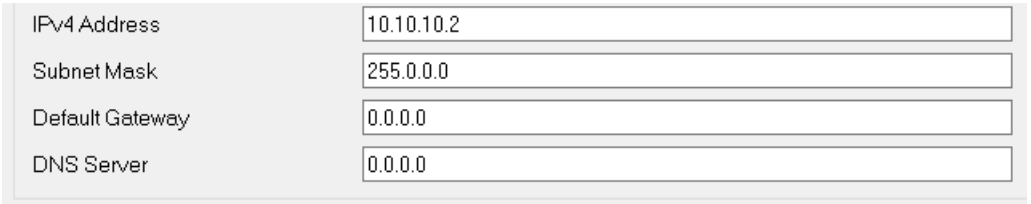


2.configuration

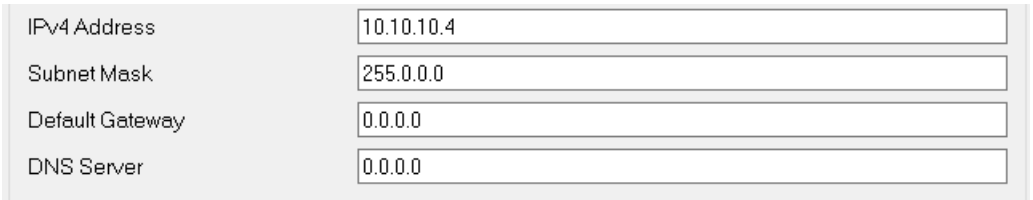
PC1



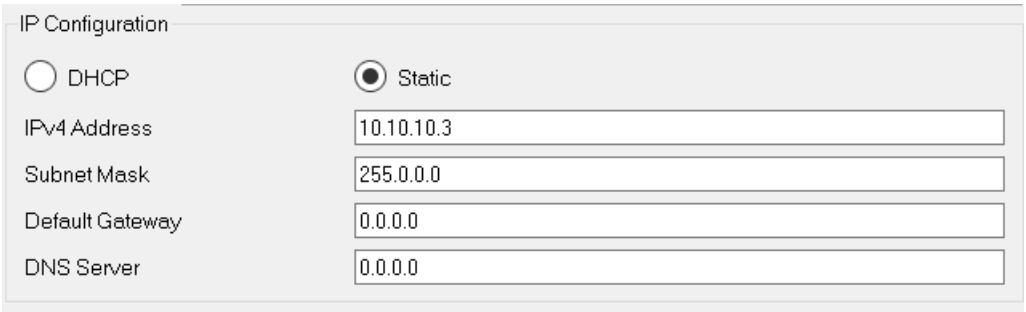
PC2



PC3



PC4



3.terminal commands

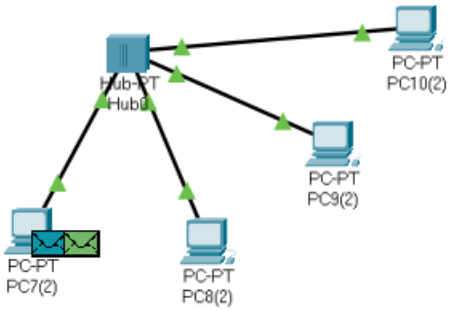
```
C:\>IPCONFIG

FastEthernet0 Connection:(default port)

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.........:
FE80::201:C9FF:FE52:C0B8
   IPv6 Address....................: ::
   IPv4 Address....................: 10.10.10.3
   Subnet Mask.....................: 255.0.0.0
   Default Gateway.................: ::
                                     0.0.0.0

Bluetooth Connection:

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.........: ::
   IPv6 Address....................: ::
   IPv4 Address....................: 0.0.0.0
   Subnet Mask.....................: 0.0.0.0
   Default Gateway.................: ::
```

Ping 10.10.10.2

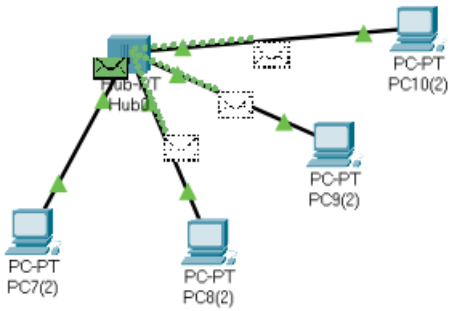
```
C:\>ping 10.10.10.2

Pinging 10.10.10.2 with 32 bytes of data:

Reply from 10.10.10.2: bytes=32 time<1ms TTL=128
Reply from 10.10.10.2: bytes=32 time<1ms TTL=128
Reply from 10.10.10.2: bytes=32 time<1ms TTL=128
Reply from 10.10.10.2: bytes=32 time=48ms TTL=128

Ping statistics for 10.10.10.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0%
loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 48ms, Average = 12ms
```

4. Simulation

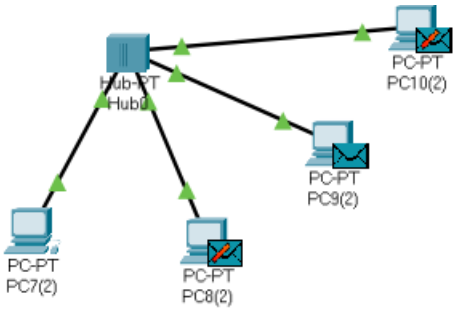
1



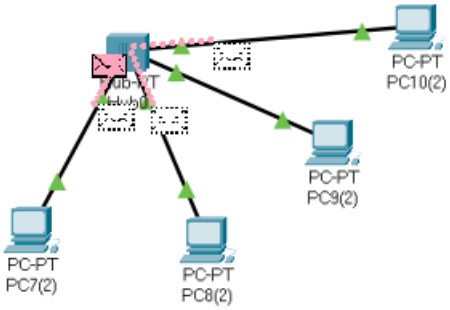
2 broadcast



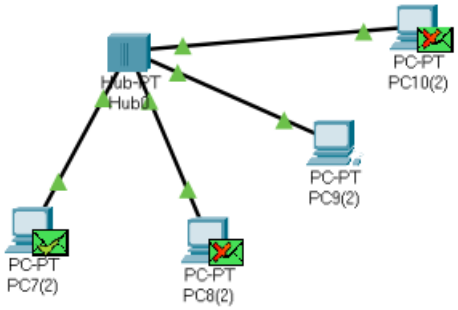
3



4 Acknowledgment

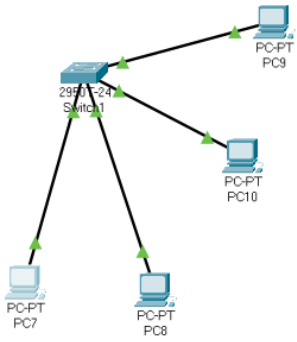


5



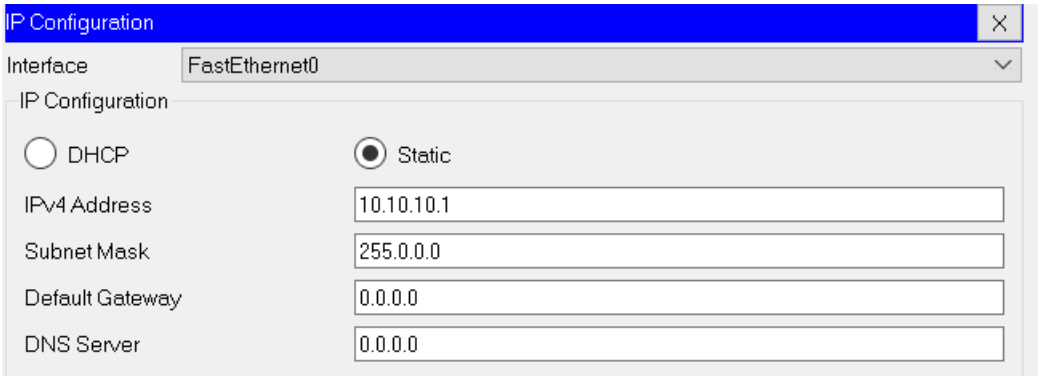
3.Switch

1. image

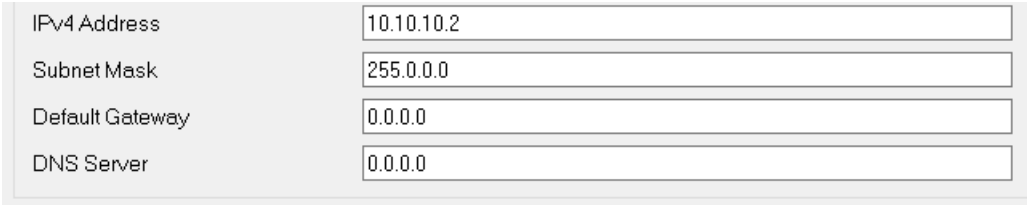


2.configuration

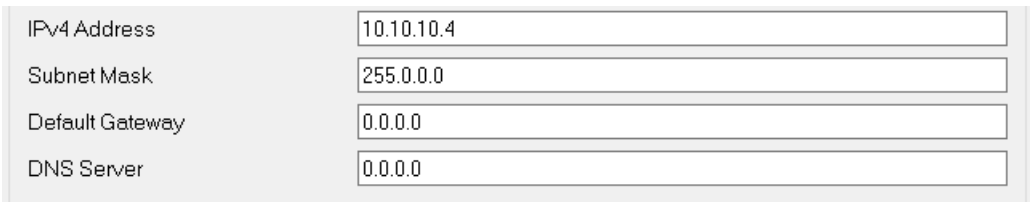
PC1



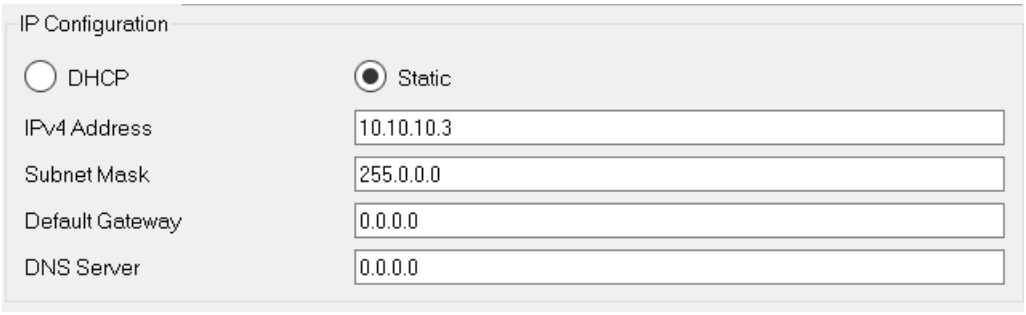
PC2



PC3



PC4



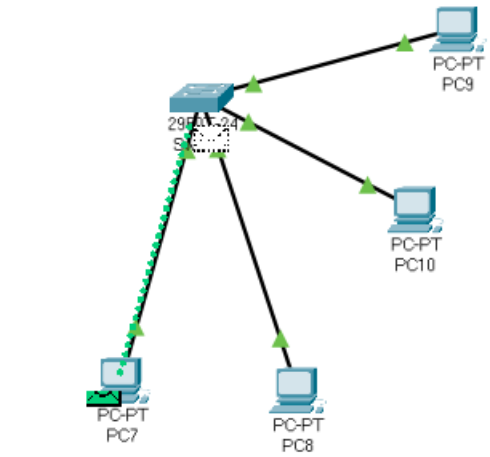
3.terminal commands

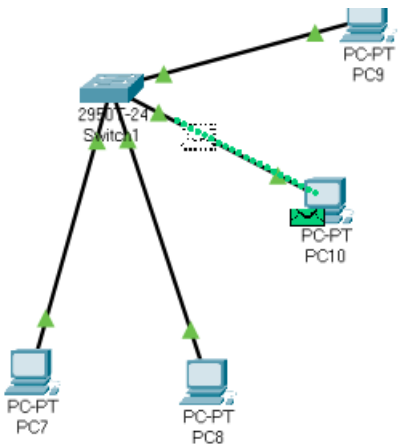
```
C:\>IPCONFIG

FastEthernet0 Connection:(default port)

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.....:
FE80::201:C9FF:FE52:C0B8
   IPv6 Address.....: ::
   IPv4 Address.....: 10.10.10.3
   Subnet Mask.....: 255.0.0.0
   Default Gateway.....: ::
                                  0.0.0.0

Bluetooth Connection:

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.....: ::
   IPv6 Address.....: ::
   IPv4 Address.....: 0.0.0.0
   Subnet Mask.....: 0.0.0.0
   Default Gateway.....: ::
```

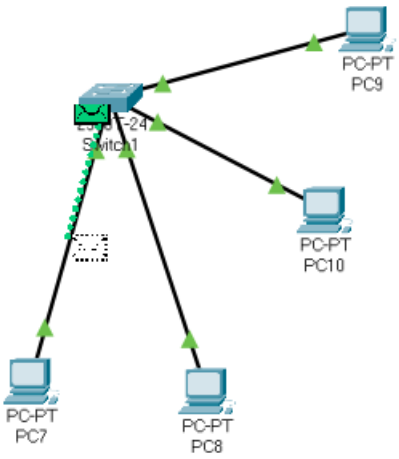
Ping 10.10.10.2

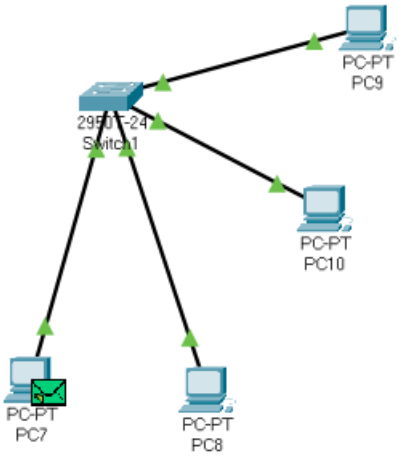
```
C:\>ping 10.10.10.2

Pinging 10.10.10.2 with 32 bytes of data:

Reply from 10.10.10.2: bytes=32 time<1ms TTL=128
Reply from 10.10.10.2: bytes=32 time<1ms TTL=128
Reply from 10.10.10.2: bytes=32 time<1ms TTL=128
Reply from 10.10.10.2: bytes=32 time=48ms TTL=128

Ping statistics for 10.10.10.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0%
loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 48ms, Average = 12ms
```

4.Simulation

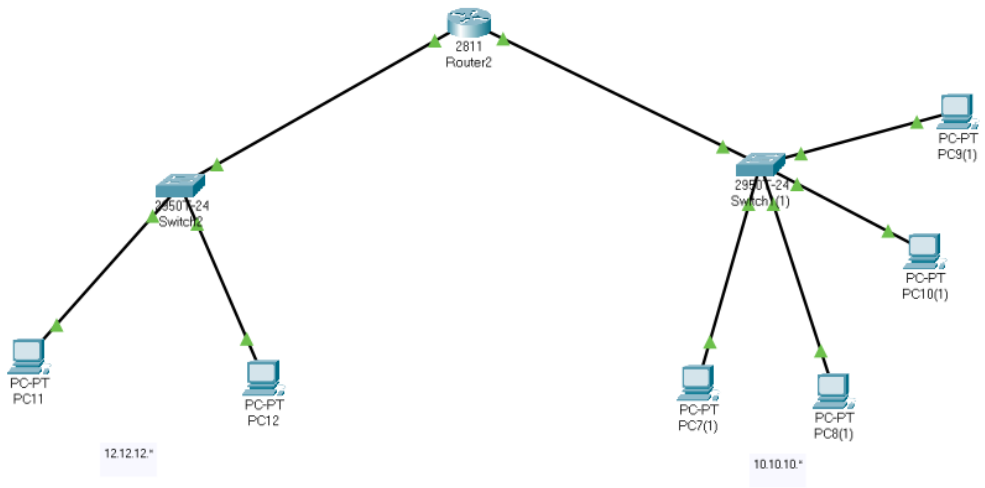




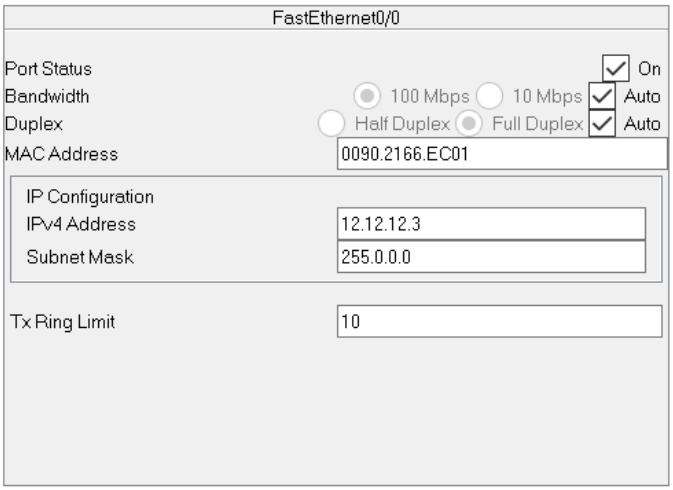

PC-PT
PC9
PC-PT
PC10
PC-PT
PC7
PC-PT
PC8

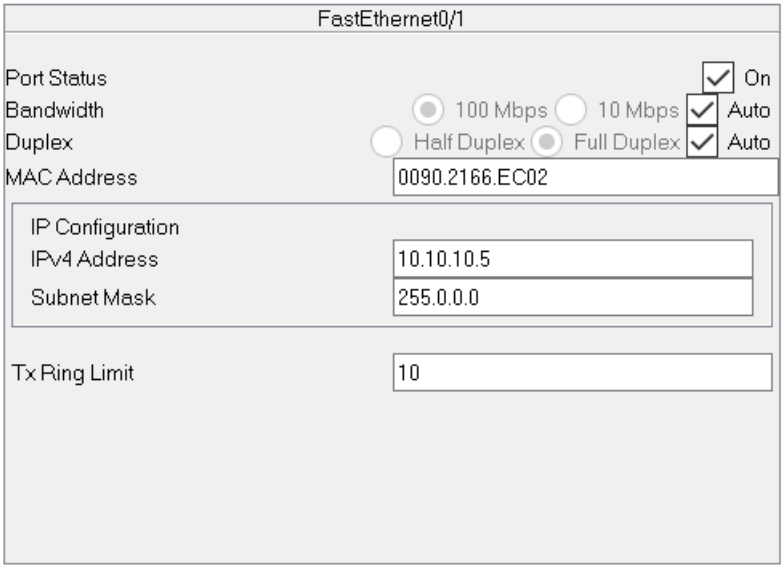

PC-PT
PC9
2950T-24
Switch1
PC-PT
PC10
PC-PT
PC7
PC-PT
PC8

4.Router



Router config

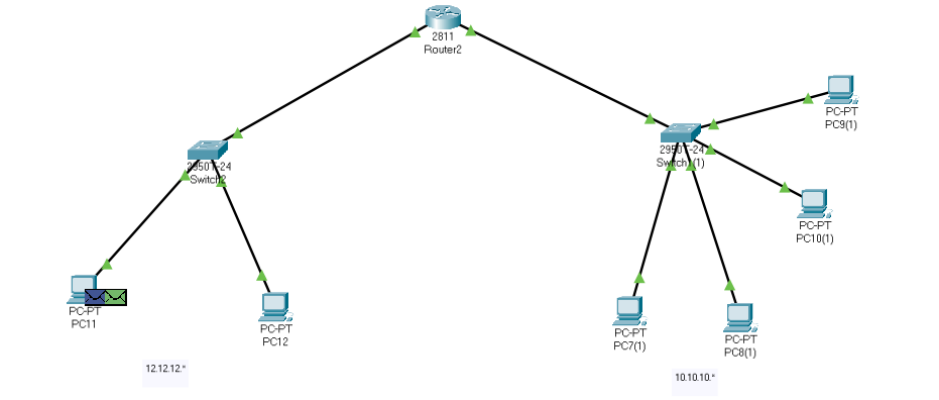


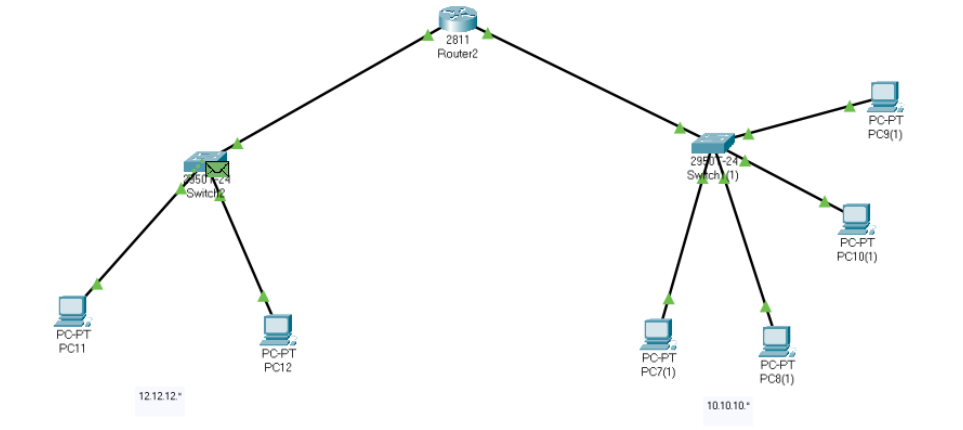


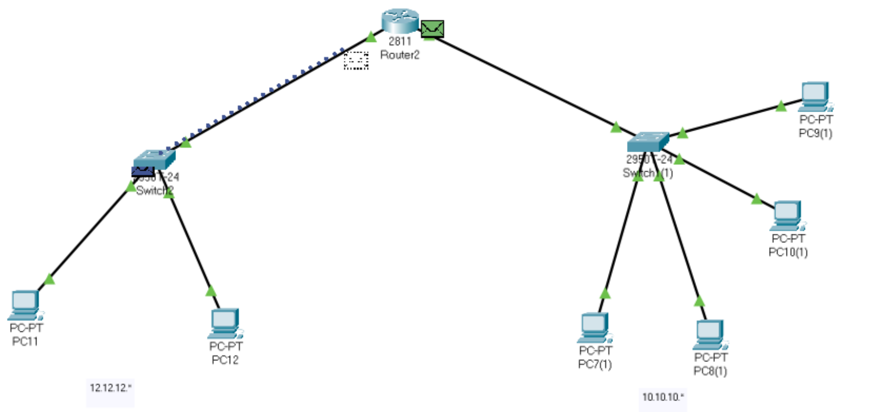
Ping command

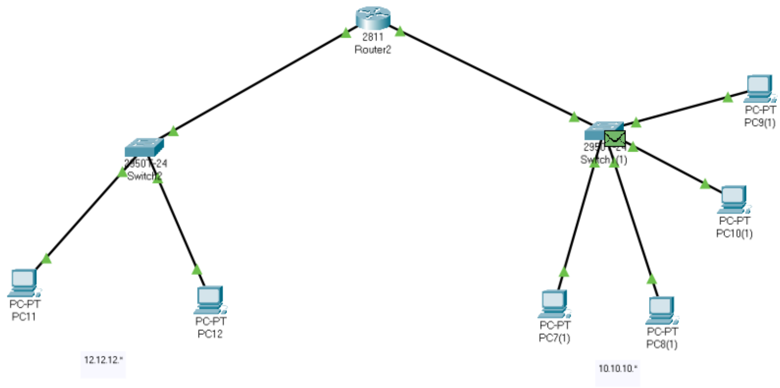
```
C:\>ping 10.10.10.2

Pinging 10.10.10.2 with 32 bytes of data:

Reply from 10.10.10.2: bytes=32 time<1ms TTL=128
Reply from 10.10.10.2: bytes=32 time<1ms TTL=128
Reply from 10.10.10.2: bytes=32 time<1ms TTL=128
Reply from 10.10.10.2: bytes=32 time=48ms TTL=128

Ping statistics for 10.10.10.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0%
loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 48ms, Average = 12ms
```

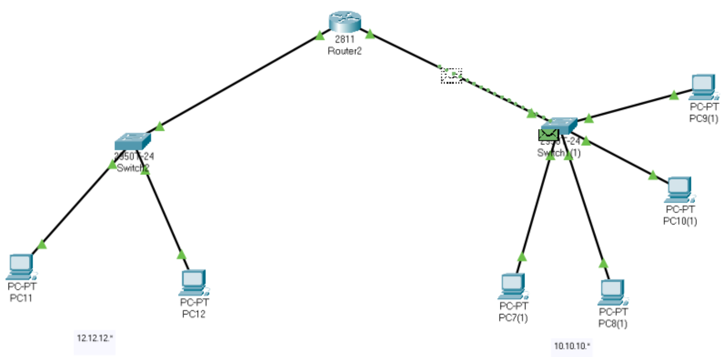
Simulation

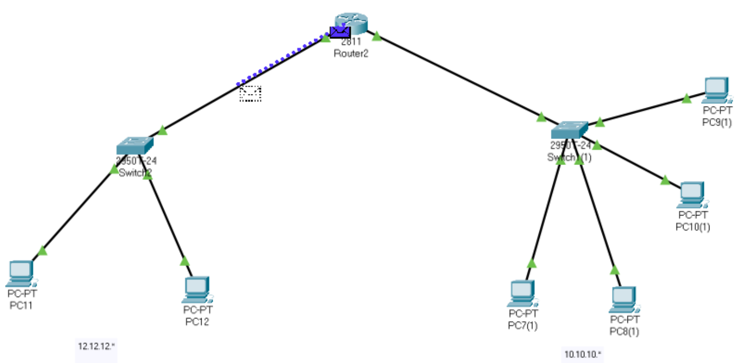


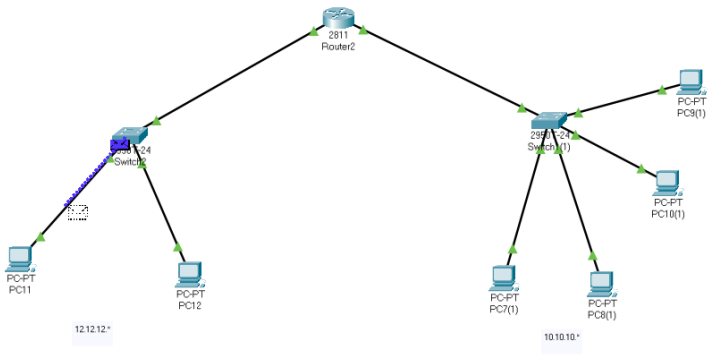



2811
Router2
PC-PT
PC11
PC-PT
PC12
PC-PT
PC9(1)
PC-PT
PC10(1)
PC-PT
PC7(1)
PC-PT
PC8(1)
12.12.12.*
10.10.10.*


2811
Router2
2950T-24
Switch2
PC-PT
PC11
PC-PT
PC12
PC-PT
PC9(1)
PC-PT
PC10(1)
PC-PT
PC7(1)
PC-PT
PC8(1)
12.12.12.*
10.10.10.*

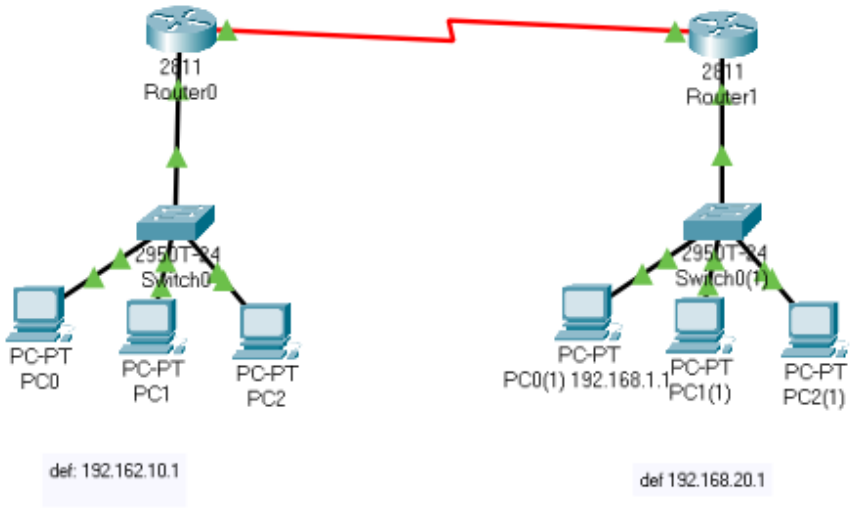

2811
Router2
PC-PT
PC11
PC-PT
PC12
PC-PT
PC9(1)
PC-PT
PC10(1)
PC-PT
PC7(1)
PC-PT
PC8(1)
12.12.12.*
10.10.10.*


Router2
2950T-24
PC-PT
PC11
PC-PT
PC12
PC-PT
PC9(1)
PC-PT
PC10(1)
PC-PT
PC7(1)
PC-PT
PC8(1)
12.12.12.*
10.10.10.*



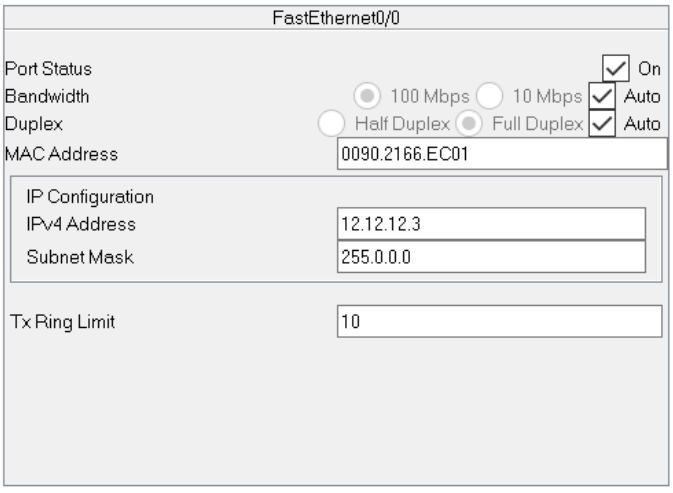
---

5 router-router

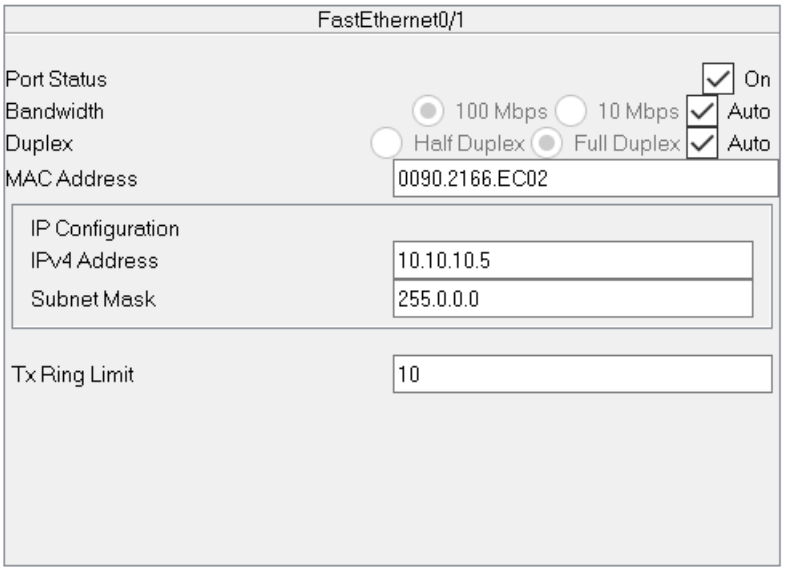


Additional module used WIC 1T

Router 1



Router 2



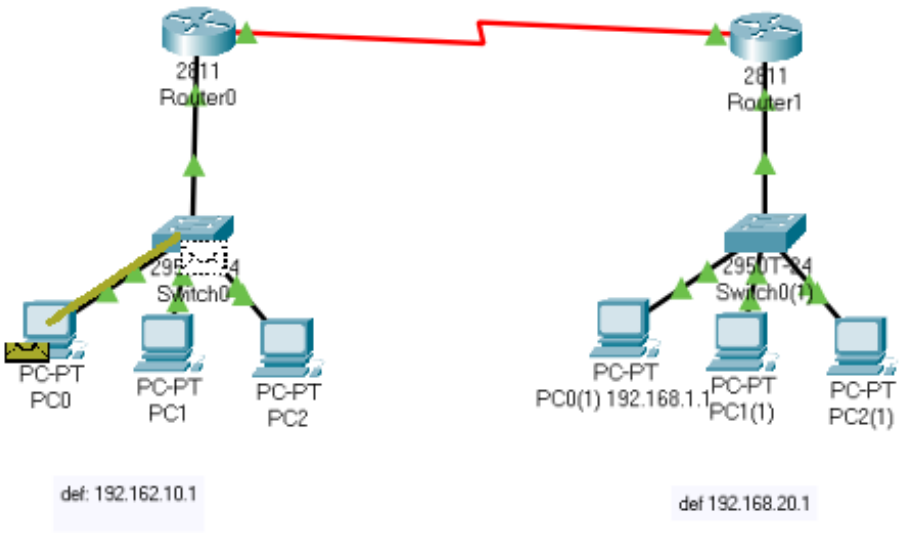
IOS commands to be run on both routers

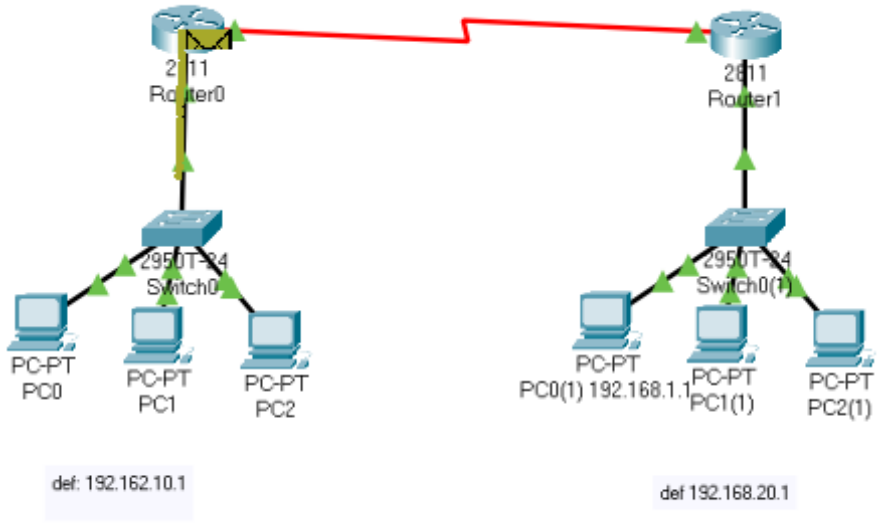
1. Switch off the router
2. Add module WIC-1T
3. Add connector serial DCE and set clock to 1280000
4. Run the commands in IOS CLI
5. #exit
6. Ip route 0.0.0.0 0.0.0.0 192.168.30.1
7. Router-router connection is set

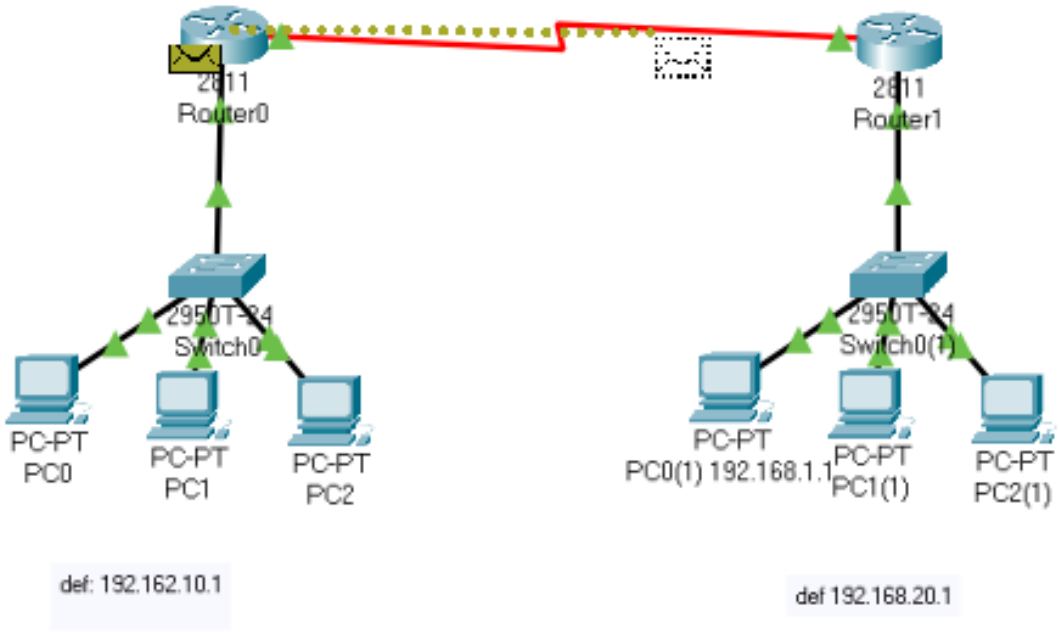
Ping

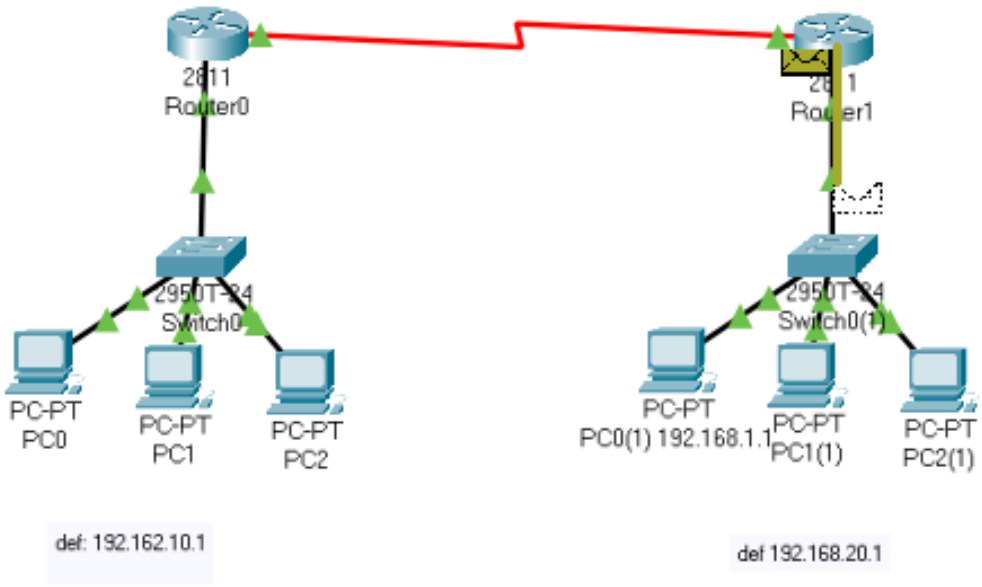
```
C:\>ping 10.10.10.2

Pinging 10.10.10.2 with 32 bytes of data:

Reply from 10.10.10.2: bytes=32 time<1ms TTL=128
Reply from 10.10.10.2: bytes=32 time<1ms TTL=128
Reply from 10.10.10.2: bytes=32 time<1ms TTL=128
Reply from 10.10.10.2: bytes=32 time=48ms TTL=128

Ping statistics for 10.10.10.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0%
loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 48ms, Average = 12ms
```

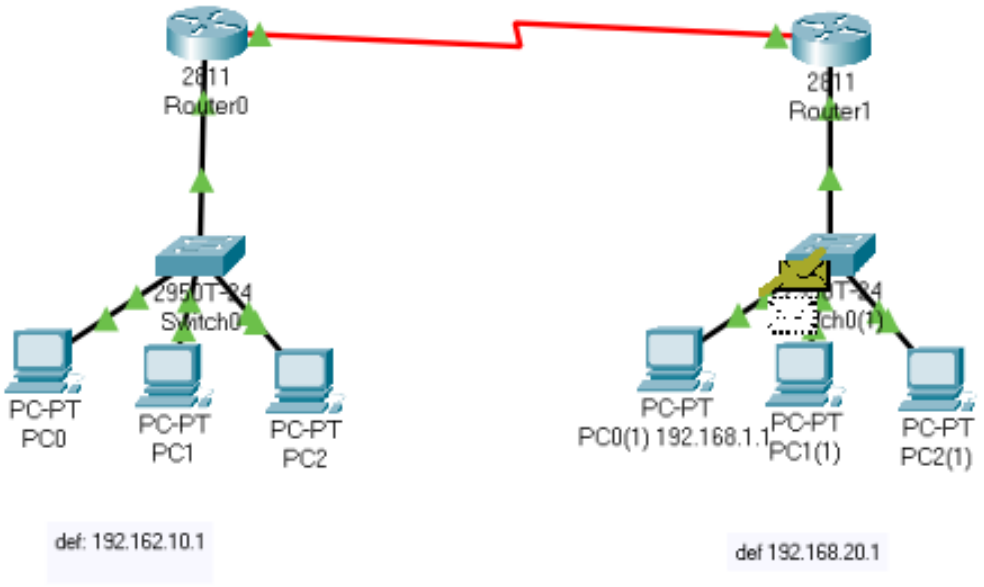
Simulation

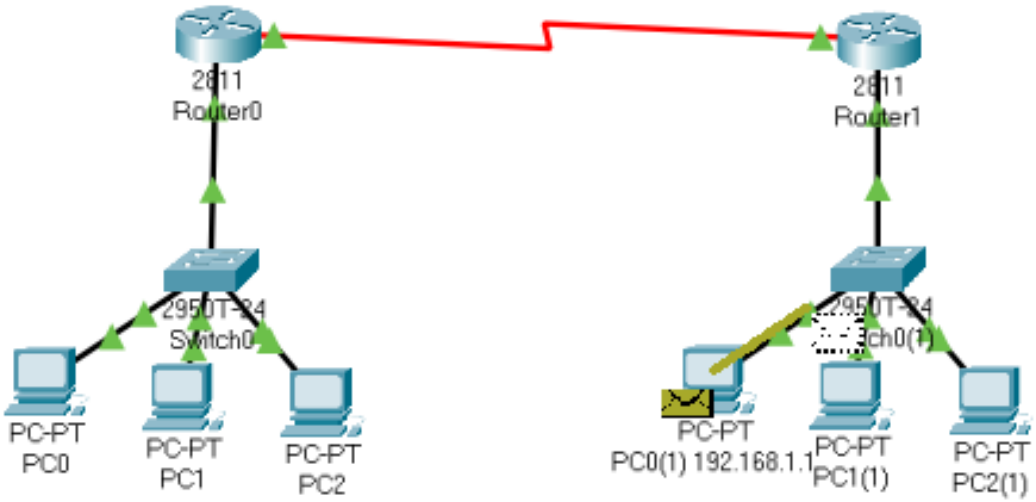



2811
Router0
2811
Router1
2950T-24
Switch0
2950T-24
Switch0(1)
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC0(1) 192.168.1.1
PC-PT
PC1(1)
PC-PT
PC2(1)
def: 192.162.10.1
def 192.168.20.1


2811
Router0
2811
Router1
2950T-24
Switch0
2950T-24
Switch0(1)
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC0(1) 192.168.1.1
PC-PT
PC1(1)
PC-PT
PC2(1)
def: 192.162.10.1
def 192.168.20.1

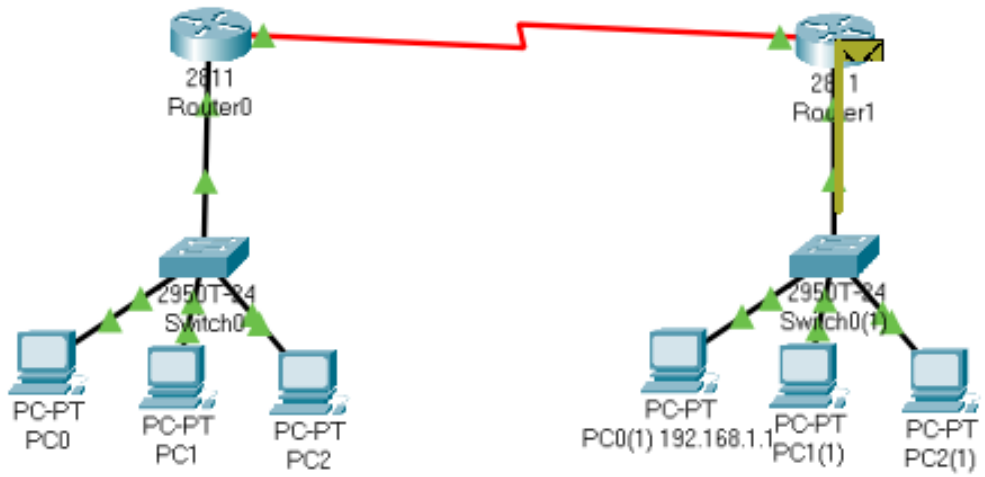

2811
Router0
2811
Router1
2950T-24
Switch0
2950T-24
Switch0(1)
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC0(1) 192.168.1.1
PC-PT
PC1(1)
PC-PT
PC2(1)
def: 192.162.10.1
def 192.168.20.1


2811
Router0
2811
Router1
2950T-24
Switch0
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC0(1) 192.168.1.1
PC-PT
PC1(1)
PC-PT
PC2(1)
def: 192.162.10.1
def 192.168.20.1



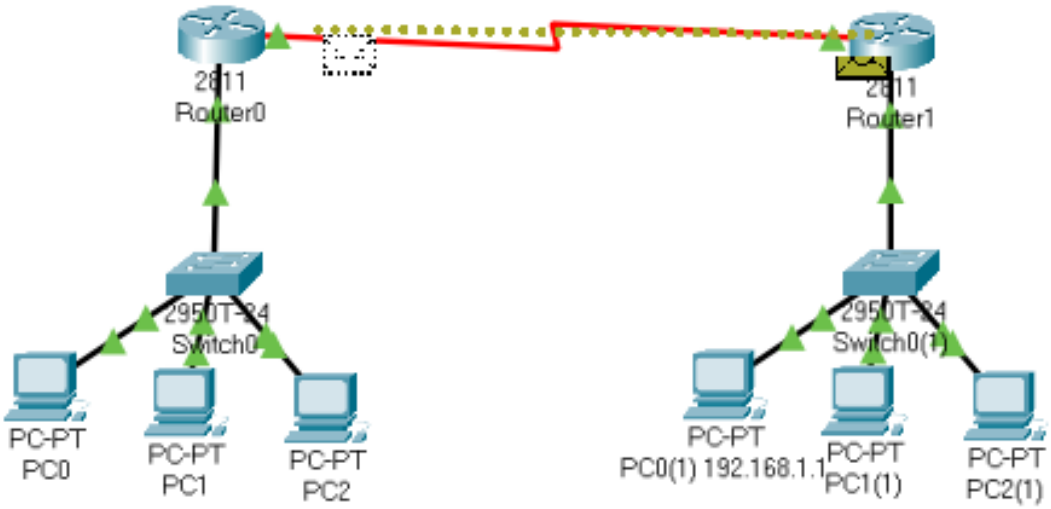
def: 192.162.10.1

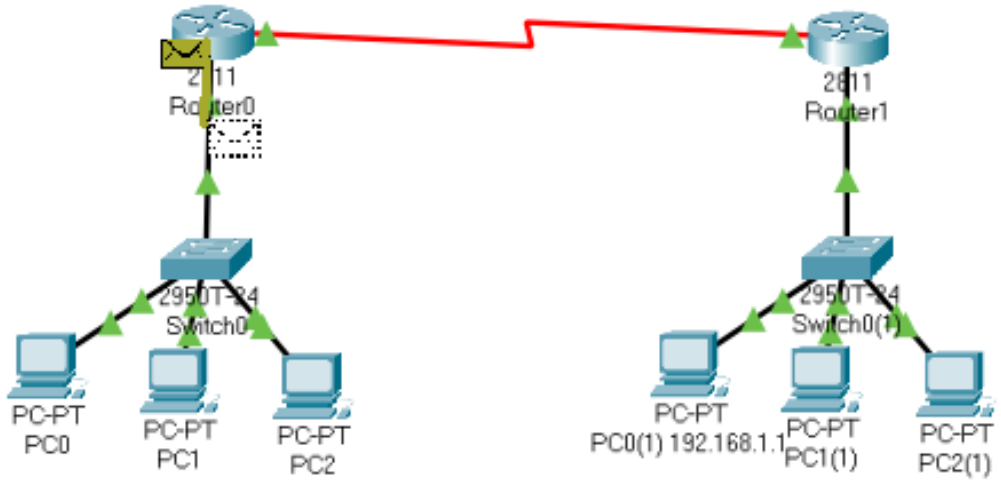
def 192.168.20.1

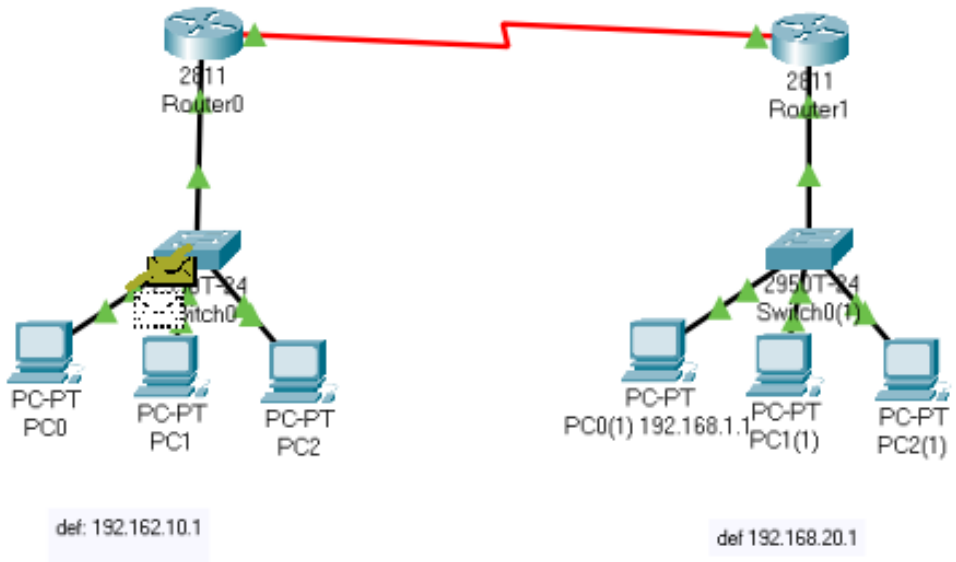


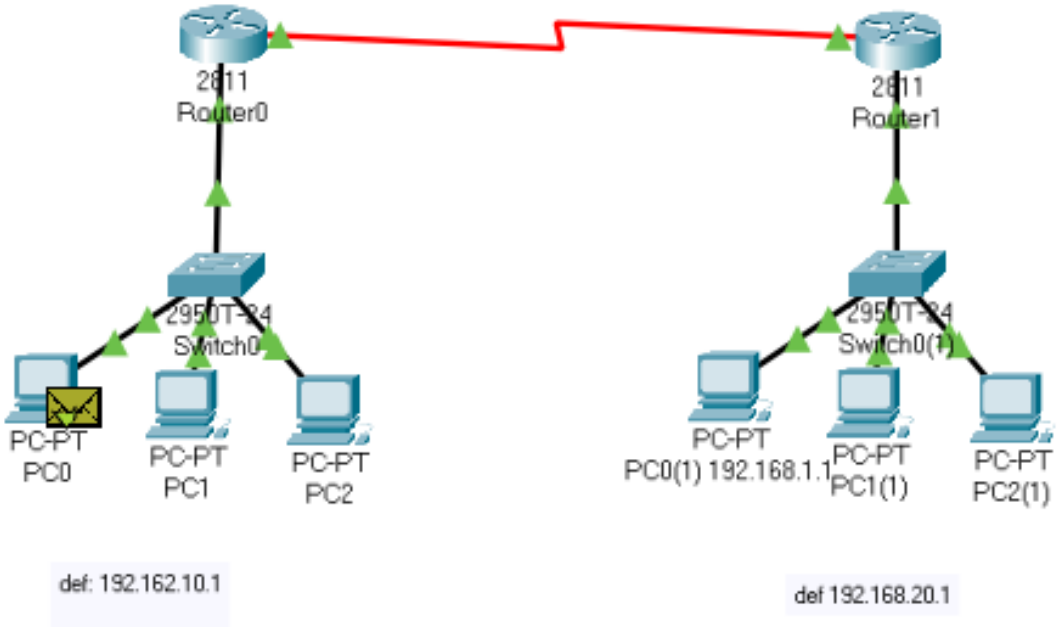
def: 192.162.10.1

def 192.168.20.1


2811
Router0
2811
Router1
2950T-24
Switch0
2950T-24
Switch0(1)
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC0(1) 192.168.1.1
PC-PT
PC1(1)
PC-PT
PC2(1)
def: 192.162.10.1
def 192.168.20.1


2811
Router0
2811
Router1
2950T-24
Switch0
2950T-24
Switch0(1)
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC0(1) 192.168.1.1
PC-PT
PC1(1)
PC-PT
PC2(1)
def: 192.162.10.1
def 192.168.20.1


2811
Router0
2811
Router1
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC0(1) 192.168.1.1
PC-PT
PC1(1)
PC-PT
PC2(1)
def: 192.162.10.1
def 192.168.20.1


2811
Router0
2811
Router1
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC0(1) 192.168.1.1
PC-PT
PC1(1)
PC-PT
PC2(1)
def: 192.162.10.1
def 192.168.20.1

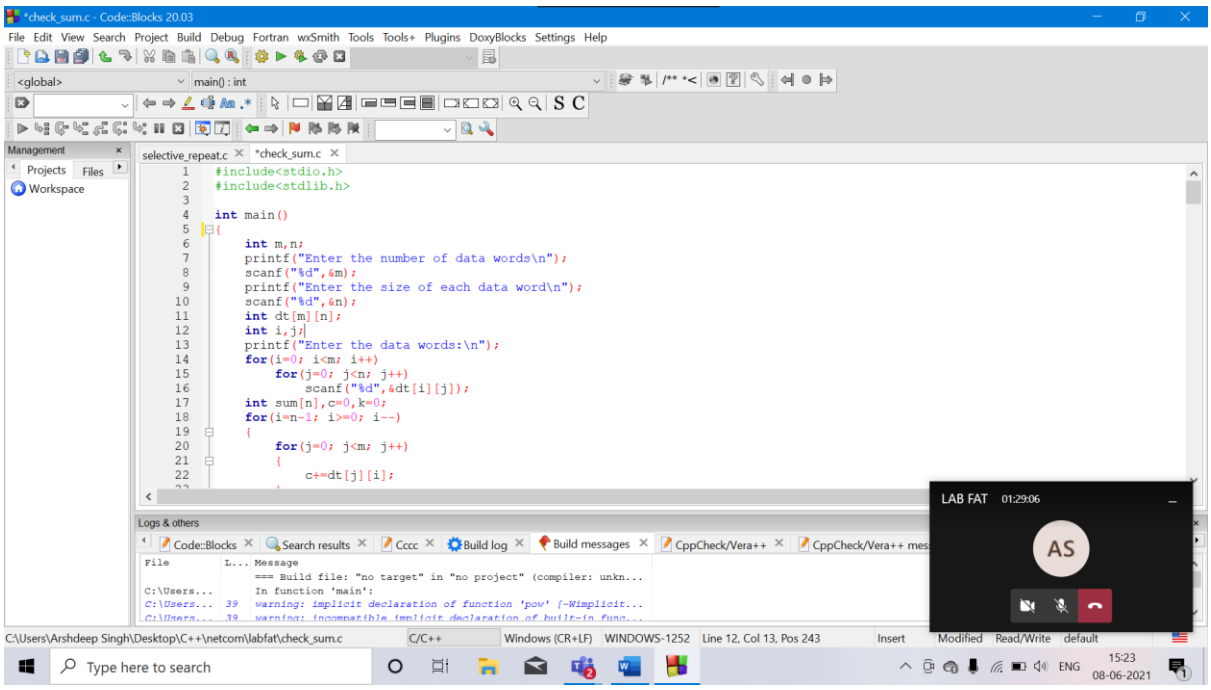
Arshdeep Singh
19BCB0086

Network and Communications LAB FAT

Questions

1. Calculate the checksum for 8 bit word, for given data 1 0 0 1 1 0 0 1 1 1 1 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0.Calculate the checksum if alternate bit is changed due to error in receiver side

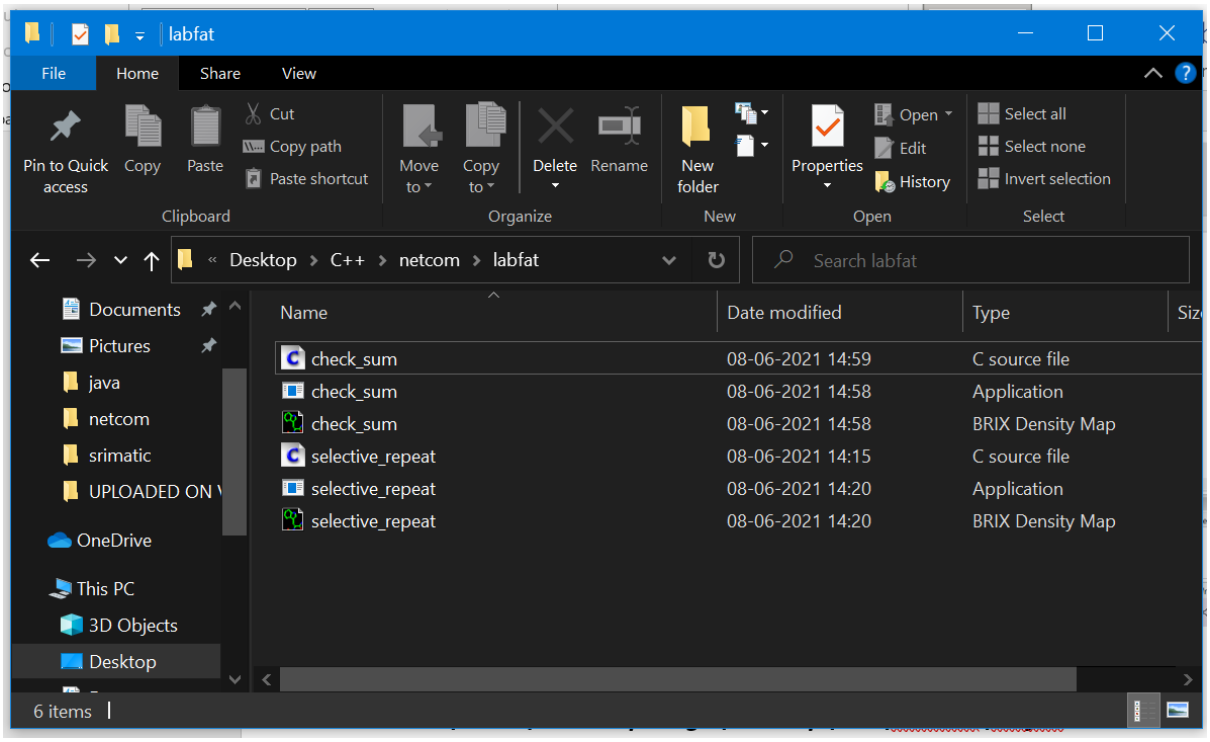
Environment Codeblocks IDE



File path in bottom left

C:\Users\Arshdeep Singh\Desktop\C++\netcom\labfat\checksum.c

Folder



CODE

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

int main()
{
   int m,n;
   printf("Enter the number of data words\n");
   scanf("%d",&m);
   printf("Enter the size of each data word\n");
   scanf("%d",&n);
   int dt[m][n];
   int i,j;
   printf("Enter the data words:\n");
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
         scanf("%d",&dt[i][j]);
   int sum[n],c=0,k=0;
   for(i=n-1; i>=0; i--)
   {
      for(j=0; j<m; j++)
      {
         c+=dt[j][i];
      }
      if(c%2==1)
      {
         sum[i]=1;
```

```
        c--;
    }
    else
        sum[i]=0;
    c=c/2;
}
i=0;
m=0;

while(i<c)
{
    if(pow(2,i)<=c)
        i++;
}
int ojsum[100],ocsum[100];
j=0;
k=0;
printf("\nActual Sum Is:\n");
for(i; i>=0; i--)
{
    if(fmod(c,pow(2,i))!=c)
    {
        ojsum[k]=1;
        printf("1")
        ;
        c-=pow(2,i);
    }
    else
    {
        ojsum[k]=0;
        printf("0");
    }
    k++;
}
j=0;
for(i=0; i<n; i++)
{
    ocsum[j];
    if(sum[i]==0)
    {
        printf("0");
        ocsum[j]=0;
    }
    else
    {
        printf("1");
        ocsum[j]=1;
    }
    j++;
}

int sumi[100];
j--;
```

```
k--;
int z=j;
c=0;
while(k>=0)
{

   c+=ojsum[k]+ocsum[j];
   if(c%2==0)
      sumi[j]=0;
   else
   {
      sumi[j]=1;
      c-=1;
   }
   c=c/2;
   j--;
   k--;
}
while(j>=0)
{
   if(c==1)
   {
      if(ocsum[j]==1)
      {
         sumi[j]=0;
         j--;
      }
      else
      {
         sumi[j]=1;
         j--;
         c--;
      }
   }
   else
   {
      sumi[j]=ocsum[j];
      j--;
   }
}
printf("\n\nSUM IS (IN %d BITS):\n",n);
for(i=0; i<=z; i++)
{
   printf("%d",sumi[i]);
}
printf("\n");
printf("\nCHECKSUM IS:\n");
for(i=0; i<=z; i++)
{
   if(sumi[i]==0)
      printf("1");
   else
      printf("0");
```

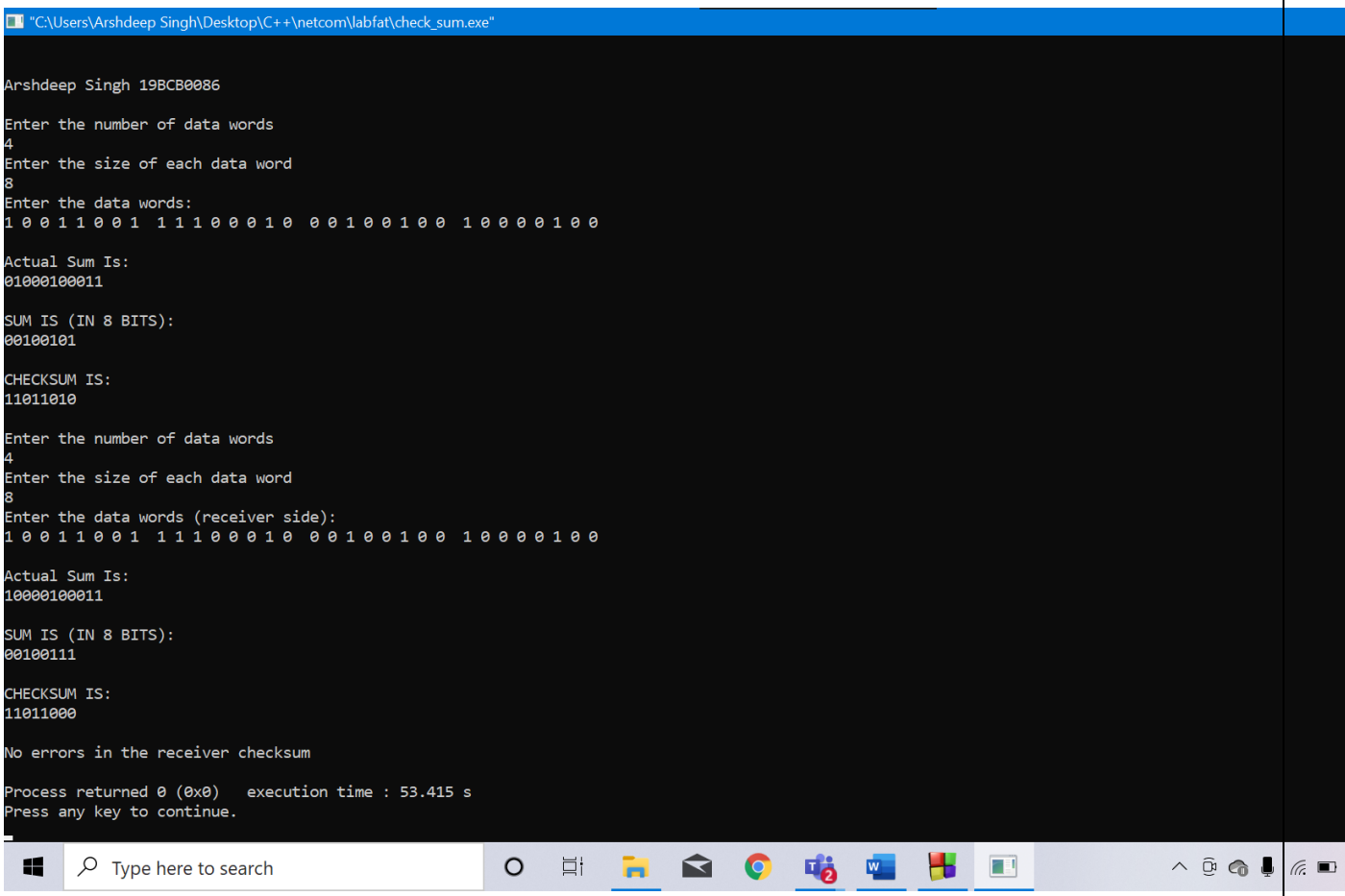
```
}

printf("\n\nEnter the number of data words\n");
scanf("%d",&m);
printf("Enter the size of each data word\n");
scanf("%d",&n);
int dt_[m][n];
printf("Enter the data words (receiver side):\n");
for(i=0; i<m; i++)
   for(j=0; j<n; j++)
      scanf("%d",&dt_[i][j]);
int sum_[n],c_=0;
for(i=n-1; i>=0; i--)
{
   for(j=0; j<m; j++)
   {
      c_+=dt_[j][i];
   }
   if(c_%2==1)
   {
      sum_[i]=1;
      c_--;
   }
   else
      sum_[i]=0;
   c_=c_/2;
}
i=0;
m=0;

while(i<c_)
{
   if(pow(2,i)<=c_)
      i++;
}
int ojsum_[100],ocsum_[100];
j=0;
k=0;
printf("\nActual Sum Is:\n");
for(i; i>=0; i--)
{
   if(fmod(c_,pow(2,i))!=c)
   {
      ojsum_[k]=1;
      printf("1")
      ;
      c_-=pow(2,i);
   }
   else
   {
      ojsum_[k]=0;
      printf("0");
   }
```

```
    k++;
}
j=0;
for(i=0; i<n; i++)
{
    ocsum_[j];
    if(sum_[i]==0)
    {
        printf("0");
        ocsum_[j]=0;
    }
    else
    {
        printf("1");
        ocsum_[j]=1;
    }
    j++;
}

int sumi_[100];
j--;
k--;
int z_=j;
c_=0;
while(k>=0)
{

    c_+=ojsum_[k]+ocsum_[j];
    if(c_%2==0)
        sumi_[j]=0;
    else
    {
        sumi_[j]=1;
        c_-=1;
    }
    c_=c_/2;
    j--;
    k--;
}
while(j>=0)
{
    if(c_==1)
    {
        if(ocsum_[j]==1)
        {
            sumi_[j]=0;
            j--;
        }
        else
        {
            sumi_[j]=1;
            j--;
            c_--;
```

```
            }
        }
        else
        {
            sumi_[j]=ocsum_[j];
            j--;
        }
    }
    printf("\n\nSUM IS (IN %d BITS):\n",n);
    for(i=0; i<=z_; i++)
    {
        printf("%d",sumi_[i]);
    }
    printf("\n");
    printf("\nCHECKSUM IS:\n");
    for(i=0; i<=z_; i++)
    {
        if(sumi_[i]==0)
            printf("1");
        else
            printf("0");
    }


    int qp=-1;
    for(i=0; i<4; i++)
        for(j=0; j<8; j++)
            {
                if(dt[i][j]!=dt_[i][j]){
                      qp=(i+1)*(j+1);
                   printf("\n\nerror in bit %d",qp);
                   break;
                }
            }
    if(qp==-1){
        printf("\n\nNo errors in the receiver checksum\n");
    }
}
```

After Compilation Output

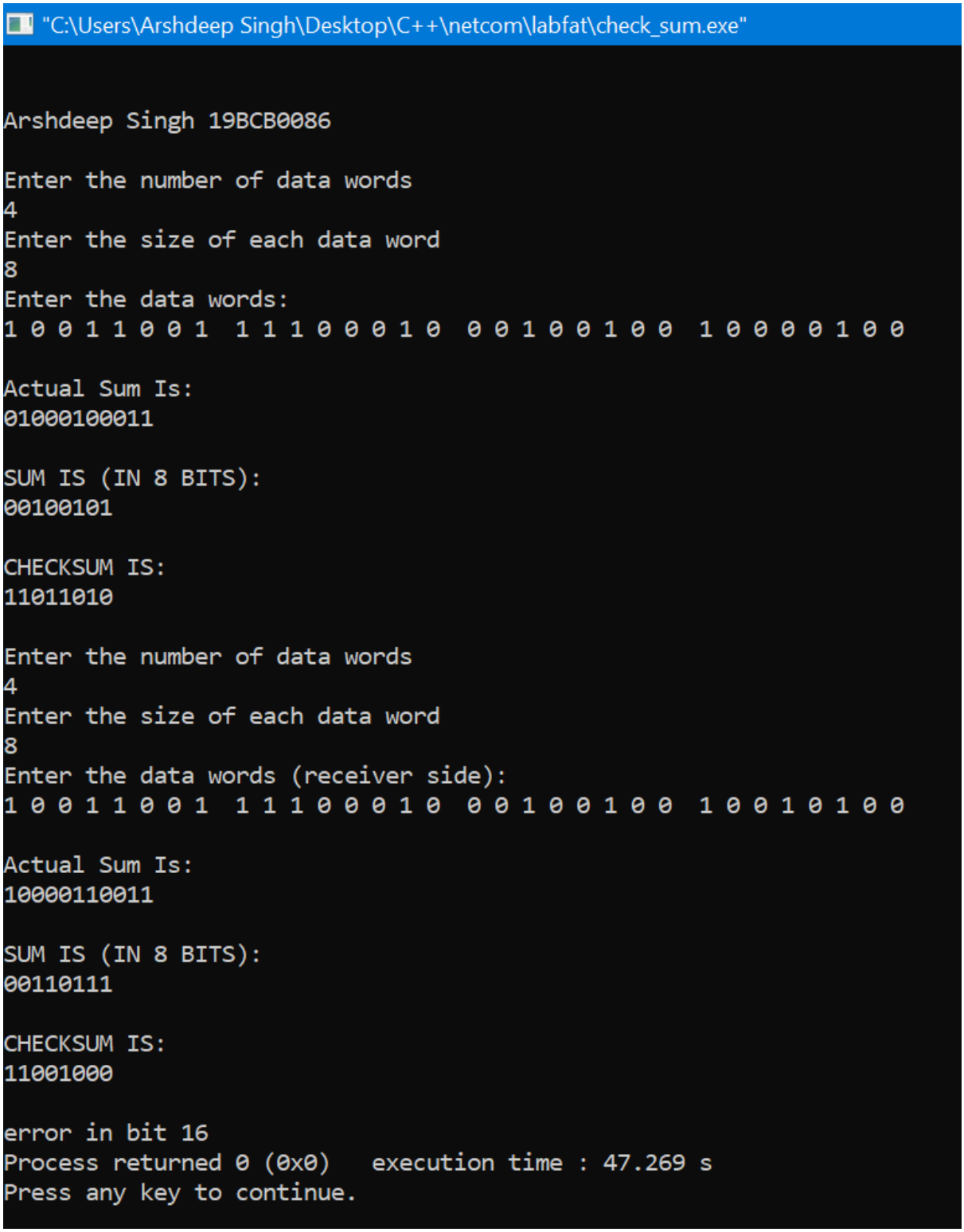
Full screen



Zoom in (No errors see last line)

```
Arshdeep Singh 19BCB0086

Enter the number of data words
4
Enter the size of each data word
8
Enter the data words:
1 0 0 1 1 0 0 1  1 1 1 0 0 0 1 0  0 0 1 0 0 1 0 0  1 0 0 0 0 1 0 0

Actual Sum Is:
01000100011

SUM IS (IN 8 BITS):
00100101

CHECKSUM IS:
11011010

Enter the number of data words
4
Enter the size of each data word
8
Enter the data words (receiver side):
1 0 0 1 1 0 0 1  1 1 1 0 0 0 1 0  0 0 1 0 0 1 0 0  1 0 0 0 0 1 0 0

Actual Sum Is:
10000100011

SUM IS (IN 8 BITS):
00100111

CHECKSUM IS:
11011000

No errors in the receiver checksum

Process returned 0 (0x0)   execution time : 53.415 s
Press any key to continue.
```

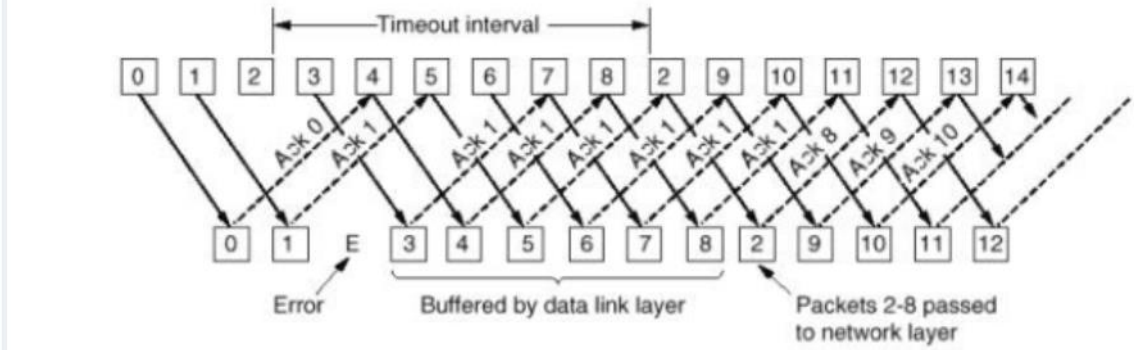
With errors(file path in window heading)



2

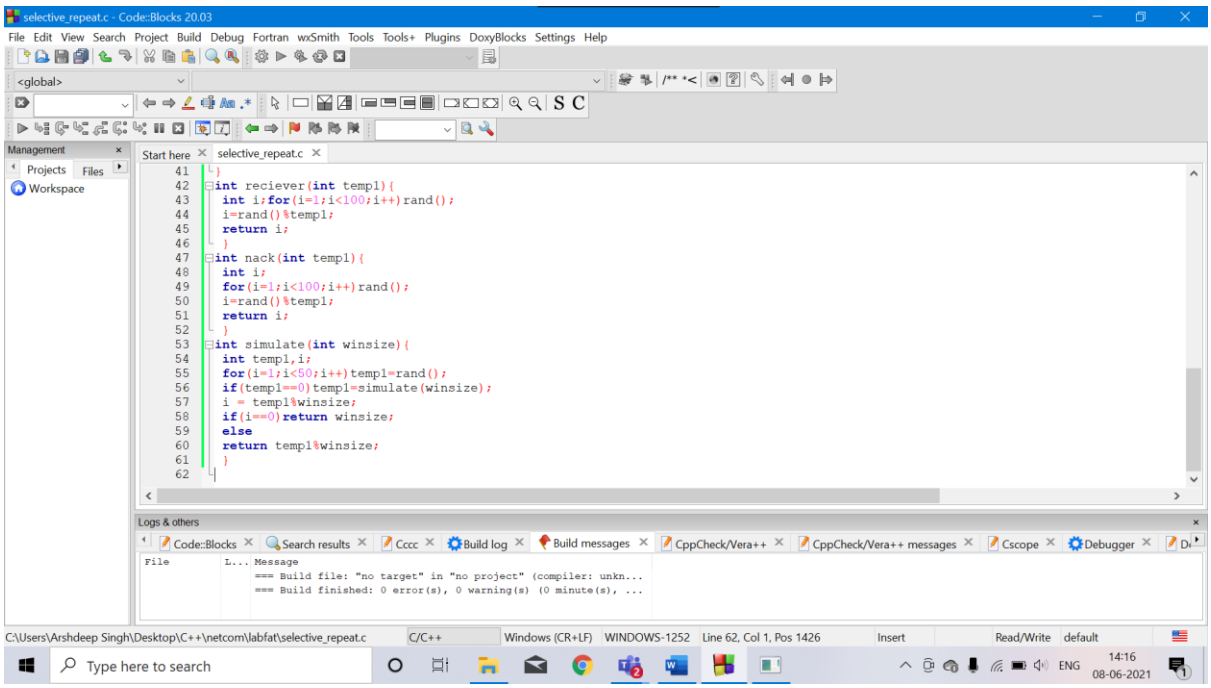
2

Write a C program for the flow control mechanism to implement the given scenario

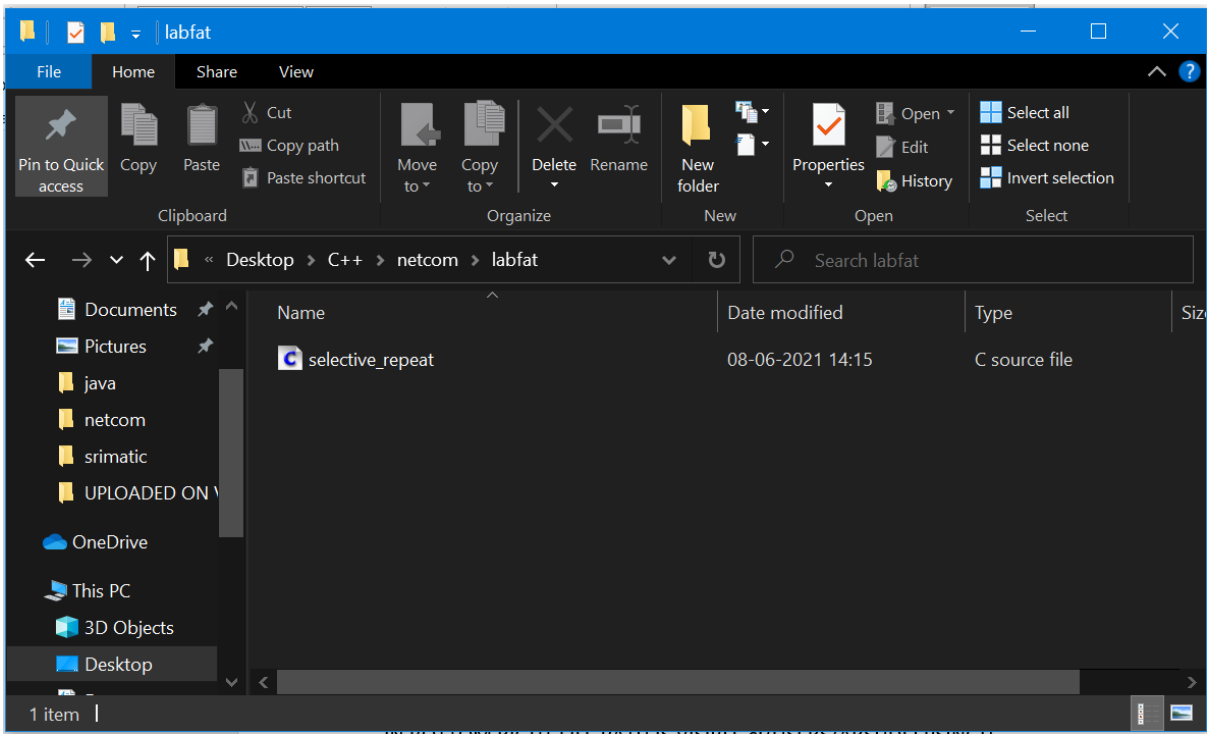


In this we see that after time out only frame 2 is resent ie this is a **selective repeat protocol**

Environment Codeblocks IDE



IN BOTTOM RIGHT FILE PATH IS VISIBLE C:\Users\Arshdeep Singh\Desktop\C++\netcom\labfat\selective_repeat.c



CODE

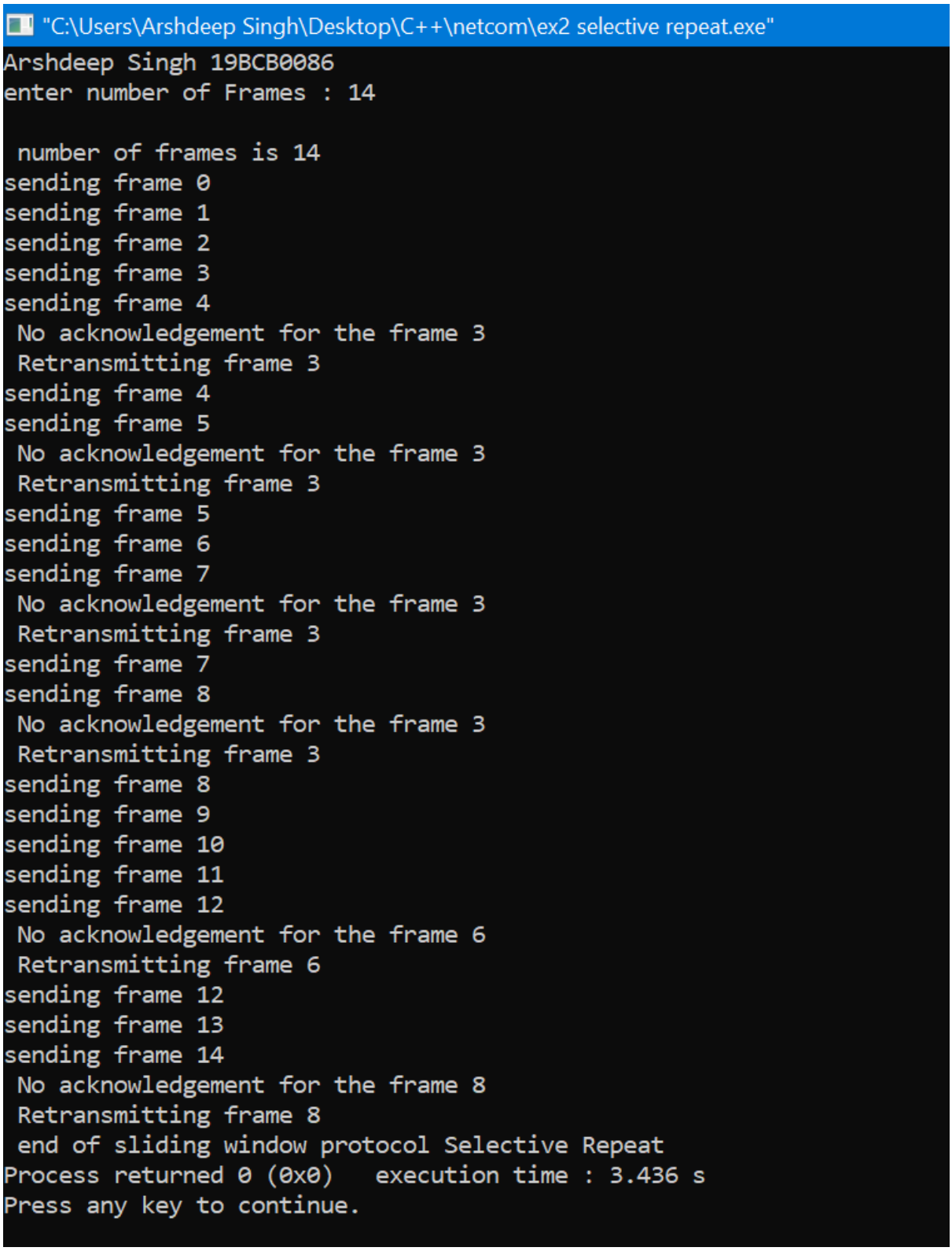
IN C PROGRAMMING LANGUAGE

```
#include<stdio.h>

#include<stdlib.h>

int main(){

 int temp1,temp2,temp3,temp4,temp5,i,winsize=8,noframes,moreframes;

 char c;

 int reciever(int);

 int simulate(int);

 int nack(int);

 temp4=0,temp1=0,temp2=0,temp3=0,temp5 = 0;

 for(i=0;i<200;i++)

 rand();

 printf("Arshdeep Singh 19BCB0086 \n");

 printf("enter number of Frames : ");

 scanf("%d",&noframes);
```

```
printf("\n number of frames is %d",noframes);
moreframes=noframes;
while(moreframes>=0){
temp1=simulate(winsize);
winsize-=temp1;
temp4+=temp1;
if(temp4 >noframes)
temp4 = noframes;
for(i=noframes - moreframes;i<=temp4;i++)
printf("\nsending frame %d",i);
temp2=reciever(temp1);
temp3+=temp2;
if(temp3 > noframes)
temp3 = noframes;
temp2 = nack(temp1);
temp5+=temp2;
if (temp5 !=0){
printf("\n No acknowledgement for the frame %d",temp5);
for(i=1;i<temp5;i++);
printf("\n Retransmitting frame %d",temp5);
}
moreframes-=temp1;
if(winsize<=0)
winsize=8;
}
printf("\n end of sliding window protocol Selective Repeat");
}
int reciever(int temp1){
int i;for(i=1;i<100;i++)rand();
i=rand()%temp1;
return i;
```

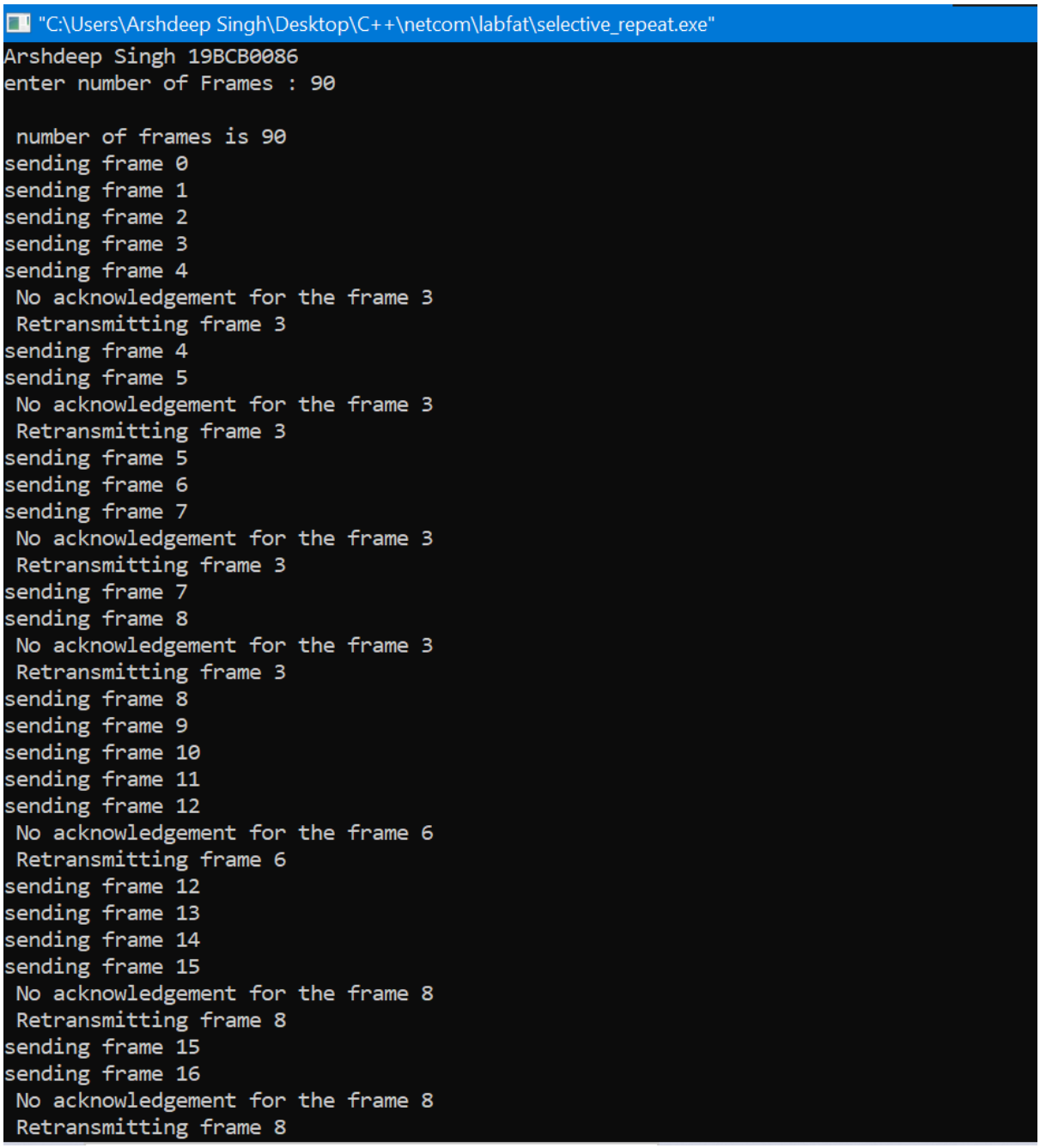
```
 }
int nack(int temp1){
 int i;
 for(i=1;i<100;i++)rand();
 i=rand()%temp1;
 return i;
 }
int simulate(int winsize){
 int temp1,i;
 for(i=1;i<50;i++)temp1=rand();
 if(temp1==0)temp1=simulate(winsize);
 i = temp1%winsize;
 if(i==0)return winsize;
 else
 return temp1%winsize;
 }
```

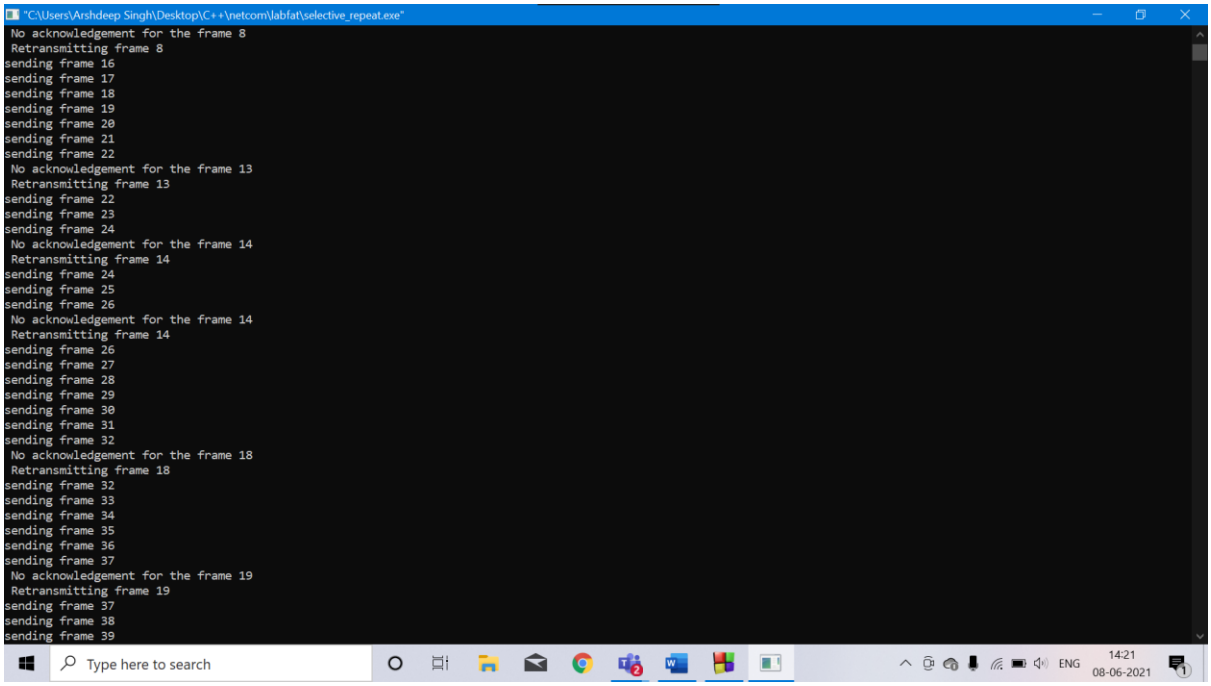
OUTPUT

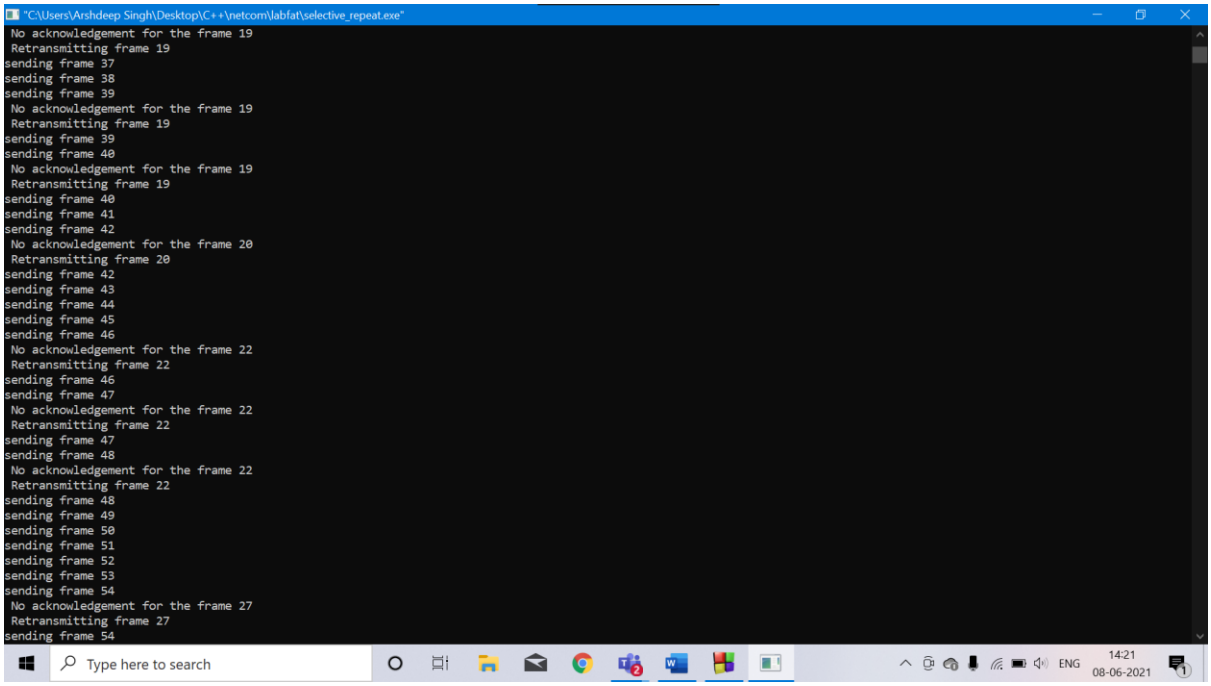

"C:\Users\Arshdeep Singh\Desktop\C++\netcom\ex2 selective repeat.exe"
Arshdeep Singh 19BCB0086
enter number of Frames : 14
number of frames is 14
sending frame 0
sending frame 1
sending frame 2
sending frame 3
sending frame 4
No acknowledgement for the frame 3
Retransmitting frame 3
sending frame 4
sending frame 5
No acknowledgement for the frame 3
Retransmitting frame 3
sending frame 5
sending frame 6
sending frame 7
No acknowledgement for the frame 3
Retransmitting frame 3
sending frame 7
sending frame 8
No acknowledgement for the frame 3
Retransmitting frame 3
sending frame 8
sending frame 9
sending frame 10
sending frame 11
sending frame 12
No acknowledgement for the frame 6
Retransmitting frame 6
sending frame 12
sending frame 13
sending frame 14
No acknowledgement for the frame 8
Retransmitting frame 8
end of sliding window protocol Selective Repeat
Process returned 0 (0x0) execution time : 3.436 s
Press any key to continue.

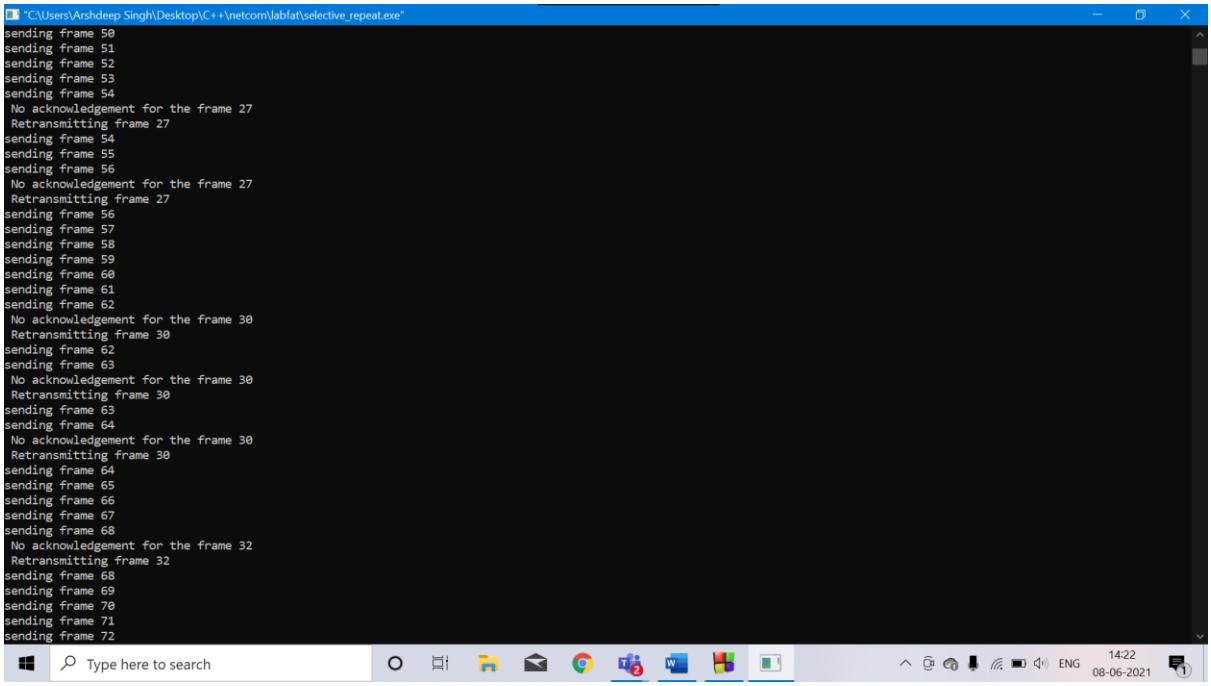
SAME OUTPUT FOR DIFFERENT NUMBER OF FRAMES

Number = 90




"C:\Users\Arshdeep Singh\Desktop\C++\netcom\labfat\selective_repeat.exe"
No acknowledgement for the frame 8
Retransmitting frame 8
sending frame 16
sending frame 17
sending frame 18
sending frame 19
sending frame 20
sending frame 21
sending frame 22
No acknowledgement for the frame 13
Retransmitting frame 13
sending frame 22
sending frame 23
sending frame 24
No acknowledgement for the frame 14
Retransmitting frame 14
sending frame 24
sending frame 25
sending frame 26
No acknowledgement for the frame 14
Retransmitting frame 14
sending frame 26
sending frame 27
sending frame 28
sending frame 29
sending frame 30
sending frame 31
sending frame 32
No acknowledgement for the frame 18
Retransmitting frame 18
sending frame 32
sending frame 33
sending frame 34
sending frame 35
sending frame 36
sending frame 37
No acknowledgement for the frame 19
Retransmitting frame 19
sending frame 37
sending frame 38
sending frame 39
Type here to search
ENG
14:21
08-06-2021


"C:\Users\Arshdeep Singh\Desktop\C++\netcom\labfat\selective_repeat.exe"
No acknowledgement for the frame 19
Retransmitting frame 19
sending frame 37
sending frame 38
sending frame 39
No acknowledgement for the frame 19
Retransmitting frame 19
sending frame 39
sending frame 40
No acknowledgement for the frame 19
Retransmitting frame 19
sending frame 40
sending frame 41
sending frame 42
No acknowledgement for the frame 20
Retransmitting frame 20
sending frame 42
sending frame 43
sending frame 44
sending frame 45
sending frame 46
No acknowledgement for the frame 22
Retransmitting frame 22
sending frame 46
sending frame 47
No acknowledgement for the frame 22
Retransmitting frame 22
sending frame 47
sending frame 48
No acknowledgement for the frame 22
Retransmitting frame 22
sending frame 48
sending frame 49
sending frame 50
sending frame 51
sending frame 52
sending frame 53
sending frame 54
No acknowledgement for the frame 27
Retransmitting frame 27
sending frame 54
Type here to search
ENG
14:21
08-06-2021



Since 90 frames output is very big