Name : Jagdish Milind Desai **Class: SY CSE**

Roll No. : 76 **Batch: S3**

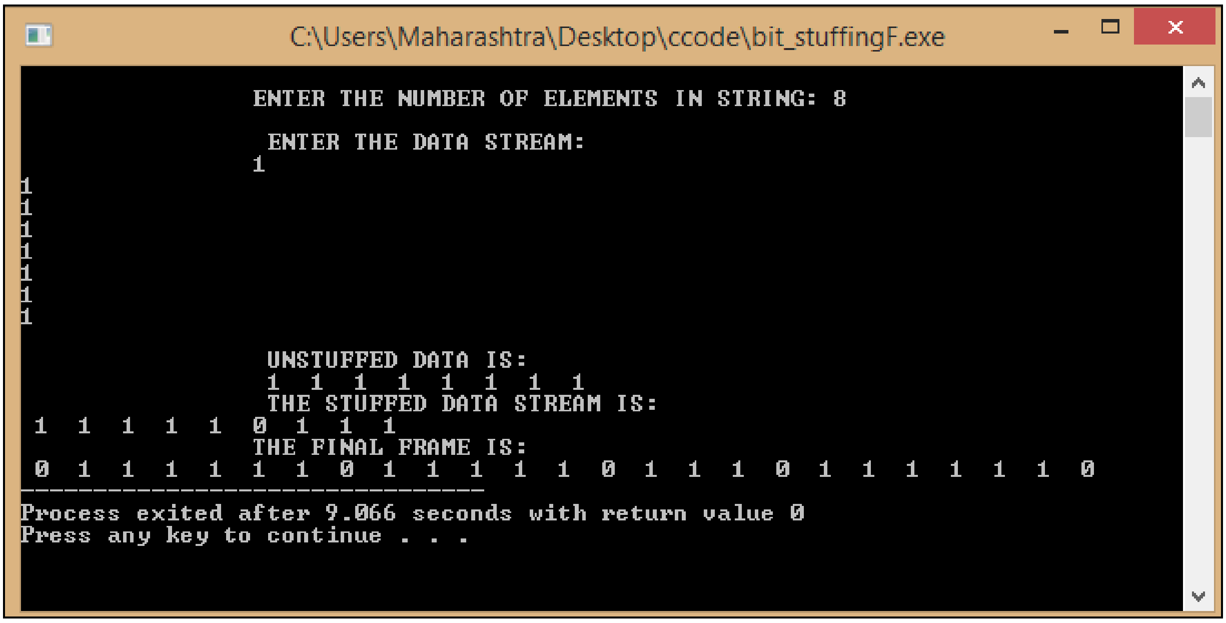
Title :Study and implementation of Bit Stuffing.

Program :

```
#include<stdio.h>
#include<string.h>
int main()
{
int unstuff[50],stuff[100],i,j,count=0,total=0,k=0,n,frame[100];
int byte[8]={0,1,1,1,1,1,1,0};
printf("\n\t\tENTER THE NUMBER OF ELEMENTS IN STRING: ");
scanf("%d",&n);
printf("\n\t\t ENTER THE DATA STREAM:\n\t\t");
for(i=0;i<n;i++)
{
scanf("%d",&unstuff[i]);
}
printf("\n\t\t UNSTUFFED DATA IS:\n\t\t");

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

i=0;
j=0;
while(j<25)
{
if(unstuff[i]==1)
{
        count++;
        if(count!=6)
        {
                stuff[j]=unstuff[i];
                i++;
                j++;
        }
        else
        {
                total++;
                count=0;
                stuff[j]=0;
                j++;
        }
}
else
{
        count=0;
        stuff[j]=unstuff[i];
        i++;
        j++;
}
}
stuff[n+total]='\0';
printf("\n\t\t THE STUFFED DATA STREAM IS:\n");

for(j=0;j<(n+total);j++)
{
printf(" %d ",stuff[j]);
}
printf("\n\t\tTHE FINAL FRAME IS:\n");
```

```
for(j=0;j<8;j++)
{
frame[k]=byte[j];
k++;
}
for(j=0;j<n+total;j++)
{
frame[k]=stuff[j];
k++;
}
for(j=0;j<8;j++)
{
frame[k]=byte[j];
k++;
}
for(j=0;j<k;j++)
{
printf(" %d ",frame[j]);
}

        return 0;

}
```

Output :-



```
                ENTER THE NUMBER OF ELEMENTS IN STRING: 8

                 ENTER THE DATA STREAM:
                1
1
1
1
1
1
1
1




                 UNSTUFFED DATA IS:
                 1  1  1  1  1  1  1  1
                 THE STUFFED DATA STREAM IS:
 1  1  1  1  1  0  1  1  1
                THE FINAL FRAME IS:
 0  1  1  1  1  1  1  0  1  1  1  1  1  0  1  1  1  0  1  1  1  1  1  1  0
--------------------------------
Process exited after 9.066 seconds with return value 0
Press any key to continue . . .
```

Name : Jagdish Milind Desai **Class: SY CSE**

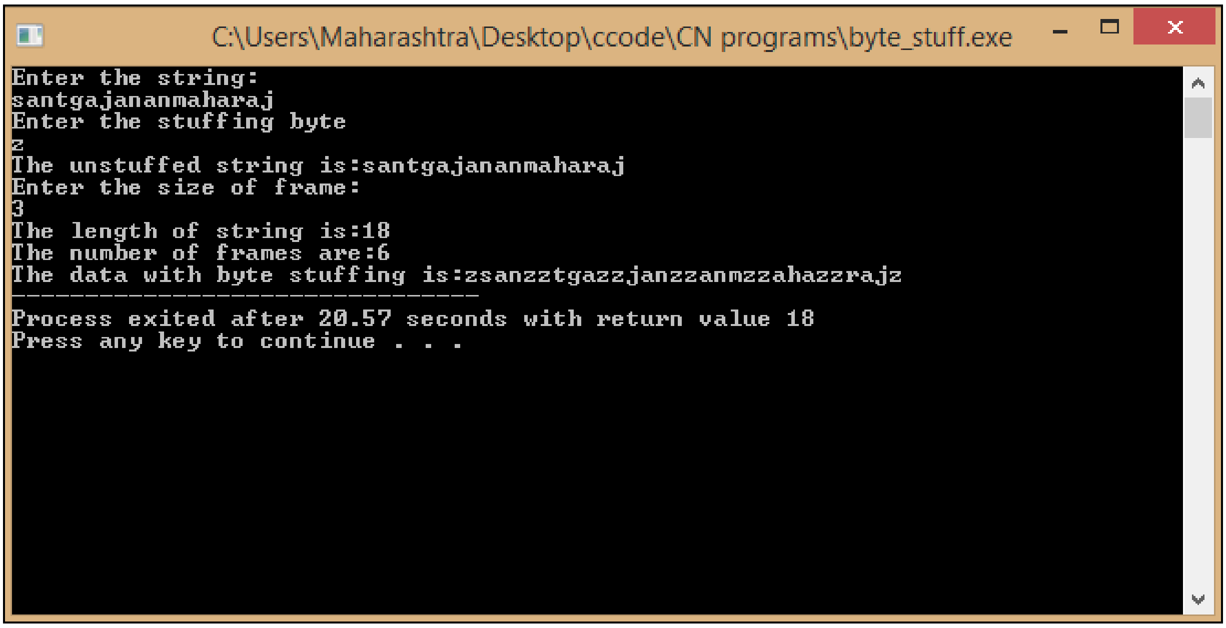
Roll No. : 76 **Batch: S3**

Title :Study and implementation of Byte Stuffing.

Program :

```
#include<stdio.h>
#include<string.h>
int  main()
{
            char data[50], stuff[50],ch;
            int i,j,len,fsize,frames;
            printf("Enter the string:\n");
            scanf("%s", data);
            printf("Enter the stuffing byte\n");
            scanf("%s",&ch);
            printf("The unstuffed string is:%s\n",data);
            printf("Enter the size of frame:\n");
            scanf("%d",&fsize);
            len=strlen(data);
            if((len%fsize)!=0)
            {
            printf("The length of string is:%d\n",len);
            frames=len/(fsize)+1;
            }
            else
            {
            printf("The length of string is:%d\n",len);
            frames=len/(fsize);
            }
            printf("The number of frames are:%d\n",frames);
            printf("The data with byte stuffing is:");
            for(i=0;i<len;)
                  {
                        j=0;
                        stuff[j]=ch;
                        j++;
                        while(j<(fsize+1))
                         {
                        if(data[i]=='\0')
                          {
                        stuff[j]=ch;
                        j++;
                        stuff[j]='\0';
                        goto ds;
                          }
                        else
                  {
                stuff[j]=data[i];
                j++;
                i++;
                 }
    }

                stuff[fsize+1]=ch;
                stuff[fsize+2]='\0';
                ds:
                printf("%s",stuff);

        }

}
```

Output :-


C:\Users\Maharashtra\Desktop\ccode\CN programs\byte_stuff.exe

```
Enter the string:
santgajananmaharaj
Enter the stuffing byte
z
The unstuffed string is:santgajananmaharaj
Enter the size of frame:
3
The length of string is:18
The number of frames are:6
The data with byte stuffing is:zsanzztgazzjanzzanmzzahazzrajz
--------------------------------
Process exited after 20.57 seconds with return value 18
Press any key to continue . . .
```

Name : Jagdish Milind Desai **Class: SY CSE**

Roll No. : 76 **Batch: S3**

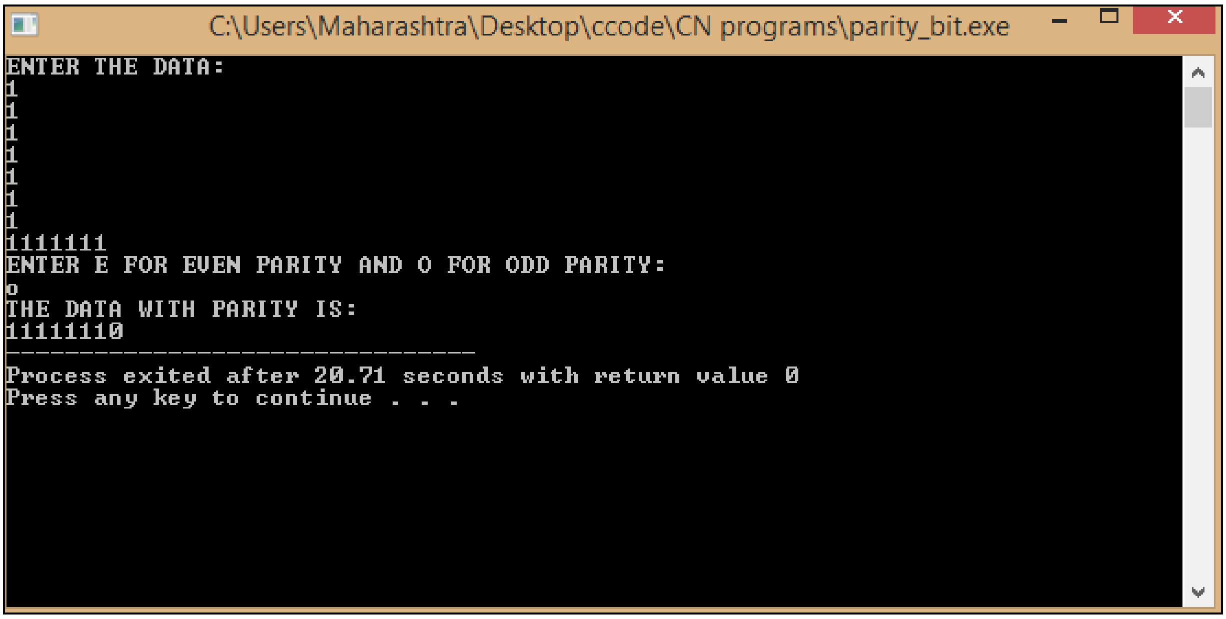
Title :Study and implementation of Parity Bit generation at sender side.

Program :

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

int main()
{
int data[7],pdata[8],i, j,count=0;
charch;
printf("ENTER THE DATA: \n");
for(i=0;i<7;i++)
 {
scanf("%d",&data[i]);
 }
for(i=0;i<7;i++)
 {
printf("%d",data[i]);
  }
printf("\nENTER E FOR EVEN PARITY AND O FOR ODD PARITY:\n");
scanf("%s",&ch);
i=0;
for(j=0;j<7;)
 {
pdata[j]=data[i];
if(data[i]==1)
 {   count++;
i++;
j++;
 }
else
{i++;
j++;
 }
  }
if(count%2==0 &&ch=='E' )
    {
pdata[j]=0;
 }
else if(count%2==0 &&ch=='O' )
    {
```

```
pdata[j]=1;
 }
else if(count%2!=0 &&ch=='E' )
    {
pdata[j]=1;
 }
else
    {
pdata[j]=0;
 }
printf("THE DATA WITH PARITY IS:\n");
for(j=0;j<8;j++)
  {
printf("%d",pdata[j]);

 }
return 0;

}
```

Output :-



C:\Users\Maharashtra\Desktop\ccode\CN programs\parity_bit.exe

```
ENTER THE DATA:
1
1
1
1
1
1
1
1111111
ENTER E FOR EVEN PARITY AND O FOR ODD PARITY:
o
THE DATA WITH PARITY IS:
11111110
--------------------------------
Process exited after 20.71 seconds with return value 0
Press any key to continue . . .
```

Name : Jagdish Milind Desai **Class: SY CSE**

Roll No. : 76 **Batch: S3**

Title :Study and implementation of Hamming code generation at sender side.

Program :

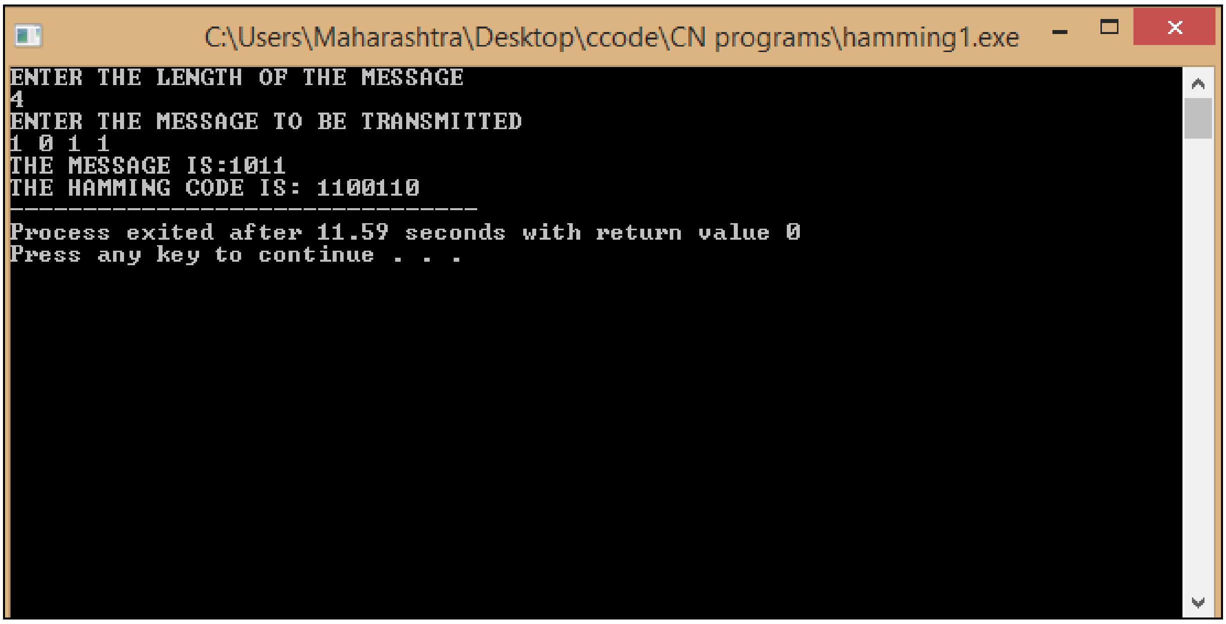
```
#include<stdio.h>

#include<math.h>

// Generation of Hamming Code


int main()

{


int message[4], Hamming[10],i,m,r,sum,count;

int m1[4],m2[4],m3[4];

charch;


printf("ENTER THE LENGTH OF THE MESSAGE\n");

scanf("%d",&m);

printf("ENTER THE MESSAGE TO BE TRANSMITTED\n");

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

{


scanf("%d",&message[i]);


}


printf("THE MESSAGE IS:");

for(i=0;i<m;i++)
```

```
{

printf("%d",message[i]);

}

m1[0]=message[0];
m1[1]=message[1];
m1[2]=message[3];

m2[0]=message[0];
m2[1]=message[2];
m2[2]=message[3];

m3[0]=message[1];
m3[1]=message[2];
m3[2]=message[3];

Hamming[2]=message[0];
Hamming[4]=message[1];
Hamming[5]=message[2];
Hamming[6]=message[3];

for(r=1;r<m;)
  {
sum=r+m+1;
if(pow(2,r)>=sum)
goto ds;
else
r++;
```

```
   }
ds:
for(i=0;i<3;)
{
if(m1[i]==1)
   {

count++;
i++;
 }
else
i++;
}
if(count%2==0)
Hamming[0]=0;
else
Hamming[0]=1;
count=0;

for(i=0;i<3;)
{
if(m2[i]==1)
   {
            count++;
            i++;
}

else
i++;
}
```

```
if(count%2==0)
Hamming[1]=0;
else
Hamming[1]=1;
count=0;
for(i=0;i<3;)
{
if(m3[i]==1)
  {
        count++;
        i++;
}
else
i++;
}
if(count%2==0)
Hamming[3]=0;
else
Hamming[3]=1;

printf("\nTHE HAMMING CODE IS: ");
for(i=sum-2;i>=0;i--)
printf("%d",Hamming[i]);
return 0;
}
```

Output :-



```
ENTER THE LENGTH OF THE MESSAGE
4
ENTER THE MESSAGE TO BE TRANSMITTED
1 0 1 1
THE MESSAGE IS:1011
THE HAMMING CODE IS: 1100110
--------------------------------
Process exited after 11.59 seconds with return value 0
Press any key to continue . . .
```

Name : Jagdish Milind Desai **Class: SY CSE**

Roll No. : 76 **Batch: S3**

Title :Study and implementation of error detection and correction by using Hamming Code at reciever.

Program :

```
#include<stdio.h>

#include<math.h>

// Error correction and detection by Hamming CODE


int main()

{


int message[4], Hamming[10],i,m,r,R1,R2,R3,R11,R22,R33,sum,count;

int m1[4],m2[4],m3[4];

printf("ENTER THE LENGTH OF HAMMING CODE\n");

scanf("%d",&m);

printf("ENTER THE HAMMING CODE RECEIVED AT THE RECEIVER\n");


for(i=m;i>0;i--)

{


    scanf("%d",&Hamming[i]);


}


printf("THE RECEIVED HAMMING CODE IS:");

for(i=m;i>0;i--)

{
```

```
printf("%d",Hamming[i]);

}




for(r=1;r<m;)
  {

if(pow(2,r)>=m)
goto ds;
else
r++;
  }

ds:
printf("\nTHE NUMBER OF REDUNTANT BITS ARE:%d\n",r);
printf("THE NUMBER OF DATA  BITS ARE:%d\n",m-r);

printf("THE REDUNTANT BITS ARE:\n");

  R1=Hamming[1];
  R2=Hamming[2];
  R3=Hamming[4];
printf("R1=%d\n",R1);
printf("R2=%d\n",R2);
printf("R3=%d\n",R3);

printf("THE MESSAGE   BITS ARE:\n");
```

```
 }
message[0]=Hamming[3];
message[1]=Hamming[5];
message[2]=Hamming[6];
message[3]=Hamming[7];
printf("D1=%d\n",message[0]);
printf("D2=%d\n",message[1]);
printf("D3=%d\n",message[2]);
printf("D4=%d\n",message[3]);

m1[0]=message[0];
m1[1]=message[1];
m1[2]=message[3];

m2[0]=message[0];
m2[1]=message[2];
m2[2]=message[3];

m3[0]=message[1];
m3[1]=message[2];
m3[2]=message[3];

        for(i=0;i<3;)
{
if(m1[i]==1)
   {

count++;
i++;
 }
else
```

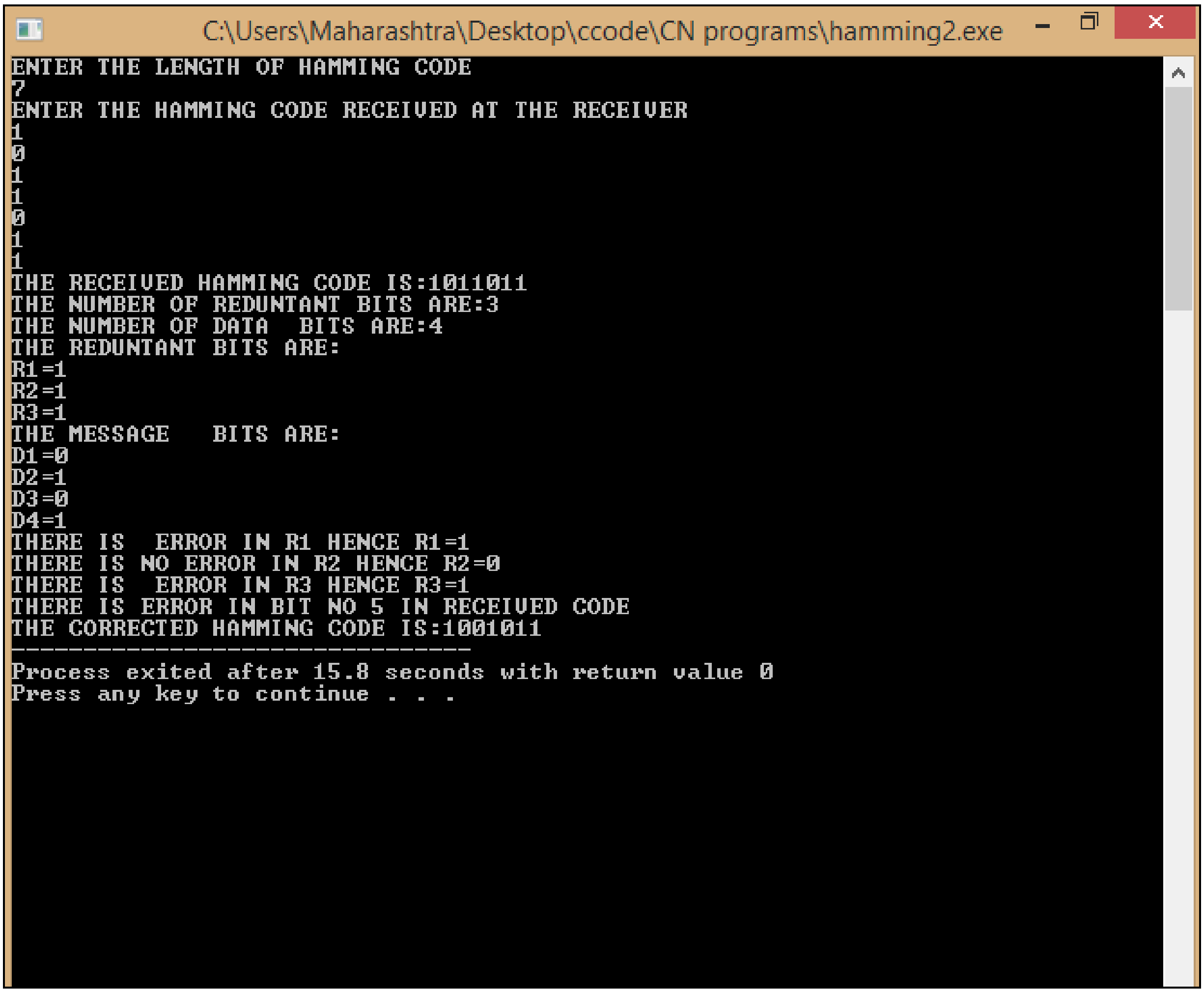
```
i++;
}
if(count%2==0)
  R11=0;
else
  R11=1;
count=0;

for(i=0;i<3;)
{
if(m2[i]==1)
   {
             count++;
             i++;
}

else
i++;
}
if(count%2==0)
  R22=0;
else
  R22=1;
count=0;
for(i=0;i<3;)
{
if(m3[i]==1)
   {
             count++;
             i++;
```

```
}
else
i++;
}
if(count%2==0)
  R33=0;
else
  R33=1;



if((R1==R11)&&(R2==R22)&&(R3==R33))
printf("THERE IS NO ERRROR IN RECEIVED CODE");
else
   {


if(R1==R11)
  {
                          R1=0;
                        printf("THERE IS NO ERROR IN R1 HENCE R1=%d\n",R1);
}else
{
                          R1=1;
                        printf("THERE IS  ERROR IN R1 HENCE R1=%d\n",R1);


}
                        if(R2==R22)
  {


                          R2=0;
                        printf("THERE IS NO ERROR IN R2 HENCE R2=%d\n",R2);
```

```
}else
{


              R2=1;
             printf("THERE IS  ERROR IN R2 HENCE R2=%d\n",R2);


}


      if(R3==R33)
 {


              R3=0;
             printf("THERE IS NO ERROR IN R3 HENCE R3=%d\n",R3);
}else
{


              R3=1;
             printf("THERE IS  ERROR IN R3 HENCE R3=%d",R3);


}




sum=R3*4+R2*2+R1*1;
printf("\nTHERE IS ERROR IN BIT NO %d IN RECEIVED CODE\n", sum);


if(Hamming[sum]==1)
Hamming[sum]=0;
else
Hamming[sum]=1;
```

```
printf("THE CORRECTED HAMMING CODE IS:");

for(i=m;i>0;i--)

printf("%d",Hamming[i]);


}

return 0;

}
```

Output :-



```
ENTER THE LENGTH OF HAMMING CODE
7
ENTER THE HAMMING CODE RECEIVED AT THE RECEIVER
1
0
1
1
0
1
1
THE RECEIVED HAMMING CODE IS:1011011
THE NUMBER OF REDUNTANT BITS ARE:3
THE NUMBER OF DATA  BITS ARE:4
THE REDUNTANT BITS ARE:
R1=1
R2=1
R3=1
THE MESSAGE   BITS ARE:
D1=0
D2=1
D3=0
D4=1
THERE IS  ERROR IN R1 HENCE R1=1
THERE IS NO ERROR IN R2 HENCE R2=0
THERE IS  ERROR IN R3 HENCE R3=1
THERE IS ERROR IN BIT NO 5 IN RECEIVED CODE
THE CORRECTED HAMMING CODE IS:1001011
--------------------------------
Process exited after 15.8 seconds with return value 0
Press any key to continue . . .
```

Name : Jagdish Milind Desai **Class: SY CSE**

Roll No. : 76 **Batch: S3**

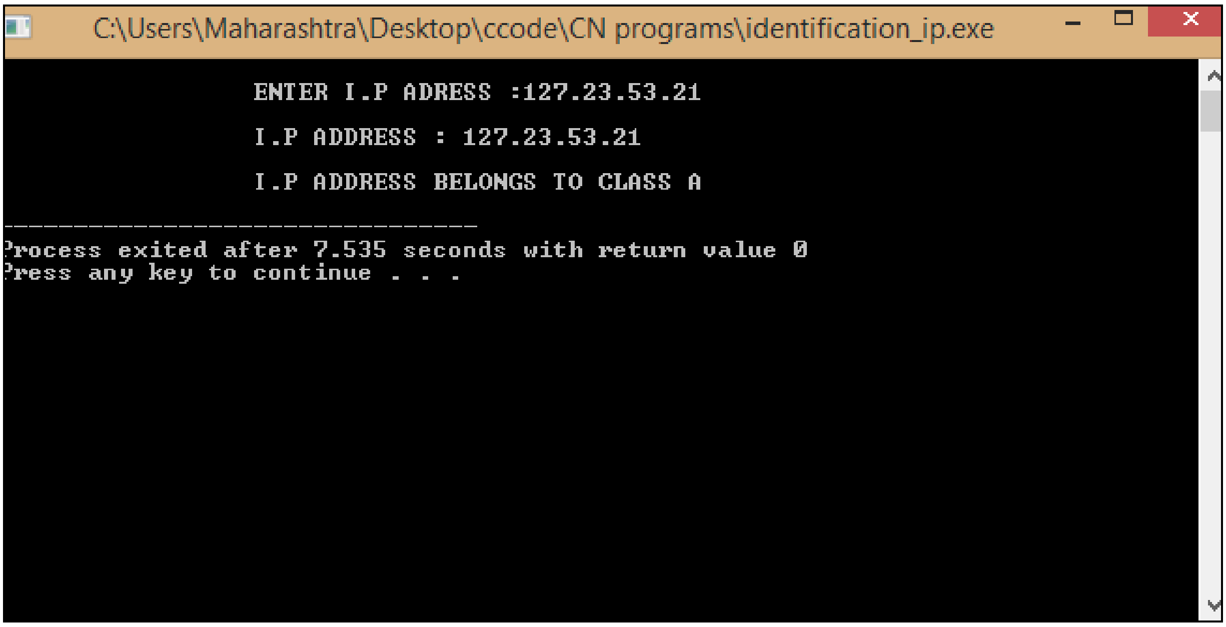
Title :Study and implementation of Identification of class of given IP address.

Program :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char input[20],s1[5],s2[5],s3[5],s4[5];
    int n1,n2,n3,n4,i=0,j=0,k,count=0;

    printf("\n\t\t ENTER I.P ADRESS :");      //127.23.63.19
    gets(input);

    for ( i=0 ; input[i]!='\0' ; i++)
    {
        if (count==0)
        {
            k=0;
            for ( j = 0; input[j]!='.'; j++)
            {
                s1[k]=input[j];
                k++;
            }
            s1[k]='\0';
        }
        if (input[i]=='.')
```

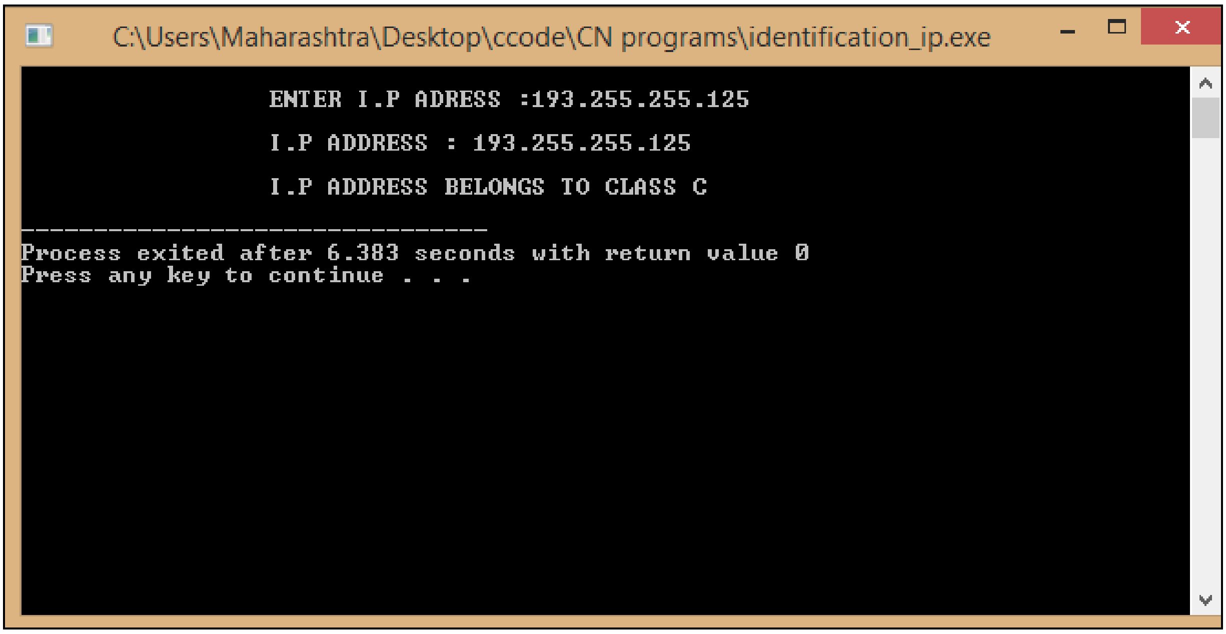
```
{
    count++;
    if (count==1)
    {
        k=0;
        for (j=i+1; input[j]!='.'; j++)
        {
            s2[k]=input[j];
            k++;
        }
        s2[k]='\0';
    }
    else if (count==2)
    {
        k=0;
        for ( j=i+1; input[j]!='.'; j++)
        {
            s3[k]=input[j];
            k++;
        }
        s3[k]='\0';
    }
    else if (count==3)
    {
        k=0;
        for ( j=i+1; input[j]!='\0'; j++)
        {
            s4[k]=input[j];
            k++;
        }
        s4[k]='\0';
    }
    }
}

n1=atoi(s1);
n2=atoi(s2);
n3=atoi(s3);
n4=atoi(s4);

printf("\n\t\t I.P ADDRESS : %d.%d.%d.%d\n",n1,n2,n3,n4);
if (n1>255 || n2>255 || n3>255 || n4>255)
{
    printf("\n\t\t WRONG I.P ADDRESS\n");
    exit(1);
}
if (n1>=0 && n1<=127)
{
    printf("\n\t\t I.P ADDRESS BELONGS TO CLASS A\n");
}
else if (n1>=128 && n1<=191)
{
    printf("\n\t\t I.P ADDRESS BELONGS TO CLASS B\n");
}
else if (n1>=192 && n1<=223)
{
    printf("\n\t\t I.P ADDRESS BELONGS TO CLASS C\n");
}
else if (n1>=224 && n1<=239)
{
    printf("\n\t\t I.P ADDRESS BELONGS TO CLASS D\n");
}
else if (n1>=240 && n1<=255)
{
        printf("\n\t\t I.P ADDRESS BELONGS TO CLASS E\n");
    }
    return 0;
}
```

Output :-



C:\Users\Maharashtra\Desktop\ccode\CN programs\identification_ip.exe

```
                 ENTER I.P ADRESS :127.23.53.21
                 I.P ADDRESS : 127.23.53.21
                 I.P ADDRESS BELONGS TO CLASS A

--------------------------------
Process exited after 7.535 seconds with return value 0
Press any key to continue . . .
```



C:\Users\Maharashtra\Desktop\ccode\CN programs\identification_ip.exe

```
                 ENTER I.P ADRESS :193.255.255.125
                 I.P ADDRESS : 193.255.255.125
                 I.P ADDRESS BELONGS TO CLASS C

--------------------------------
Process exited after 6.383 seconds with return value 0
Press any key to continue . . .
```

Name : Jagdish Milind Desai **Class: SY CSE**

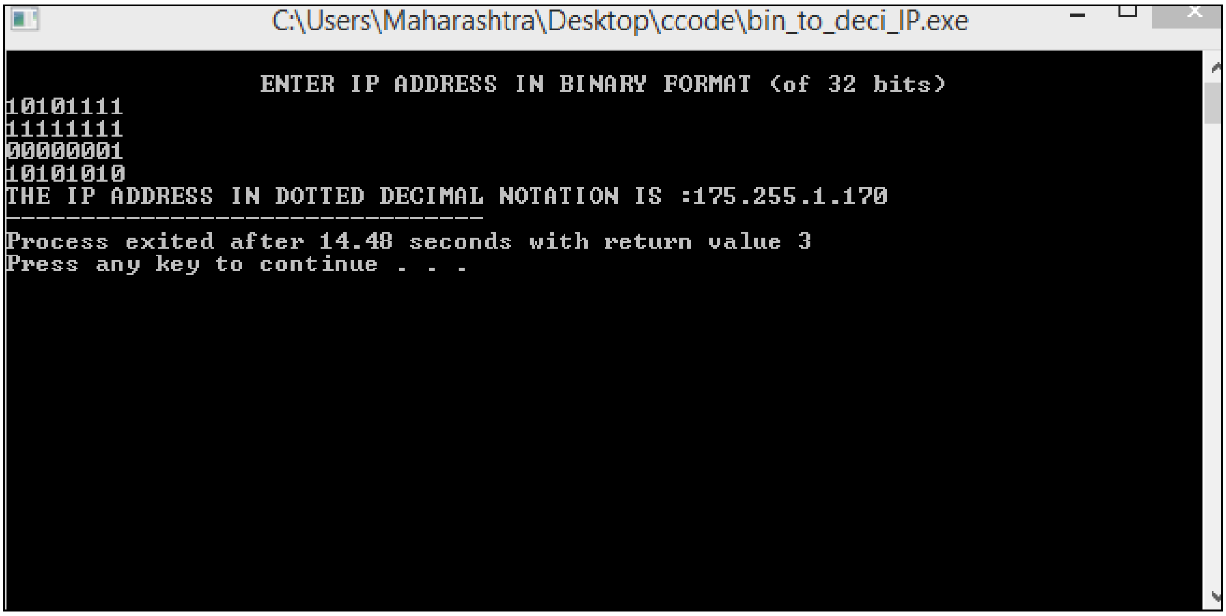
Roll No. : 76 **Batch: S3**

Title :Study and implementation of converting binary IP address to dotted decimal notation.

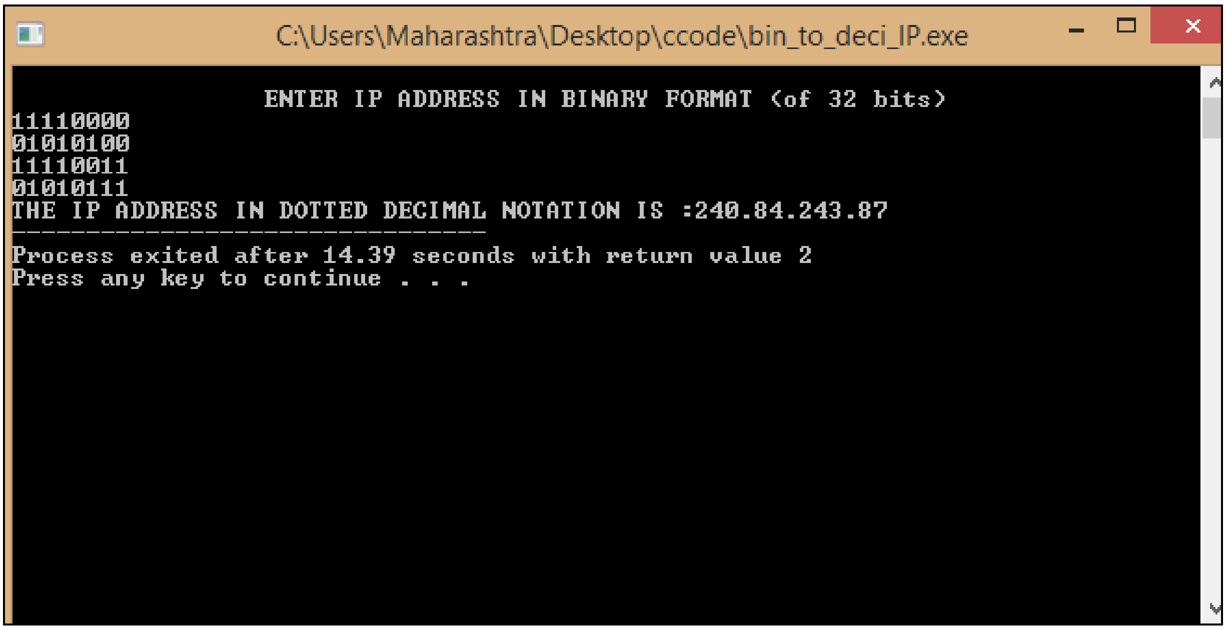
Program :

```
#include<stdio.h>
#include<math.h>
void ipInDeci(int *a[]);
int main()
{
int a[4],i;
printf("\n\t\t ENTER IP ADDRESS IN BINARY FORMAT (of 32 bits)\n");
for(i=0; i<4; i++)
{
scanf("%d",&a[i]);
}
printf("THE IP ADDRESS IN DOTTED DECIMAL NOTATION IS :");
ipInDeci(a);

}
void ipInDeci(int *a[])
{
int k,i,n,add=0,p=0;
for(k=0; k<4; k++)
{
int n= a[k];
        int i=0;
        while(n>0)
        {

            int a = n%10;
            int power = pow(2,i);
            add=add + (a*power);
            i++;
            n=n/10;
        }
        printf("%d",add);
        if(p != 3)
        {
            printf(".");
}


        p++;
        add=0;
}
}
```

Output :-



```
                 ENTER IP ADDRESS IN BINARY FORMAT (of 32 bits)
10101111
11111111
00000001
10101010
THE IP ADDRESS IN DOTTED DECIMAL NOTATION IS :175.255.1.170
--------------------------------
Process exited after 14.48 seconds with return value 3
Press any key to continue . . .
```



```
                 ENTER IP ADDRESS IN BINARY FORMAT (of 32 bits)
11110000
01010100
11110011
01010111
THE IP ADDRESS IN DOTTED DECIMAL NOTATION IS :240.84.243.87
--------------------------------
Process exited after 14.39 seconds with return value 2
Press any key to continue . . .
```

Name : Jagdish Milind Desai **Class: SY CSE**

Roll No. : 76 **Batch: S3**

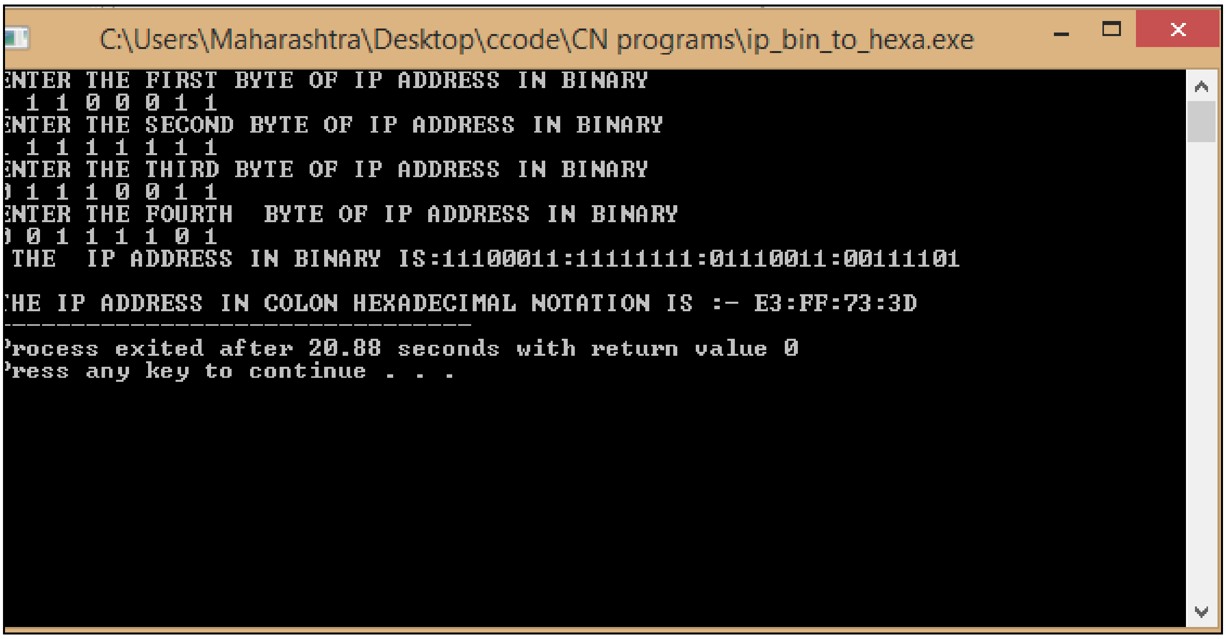
Title :Study and implementation of converting binary IP address to colon hexadecimal notation.

Program :

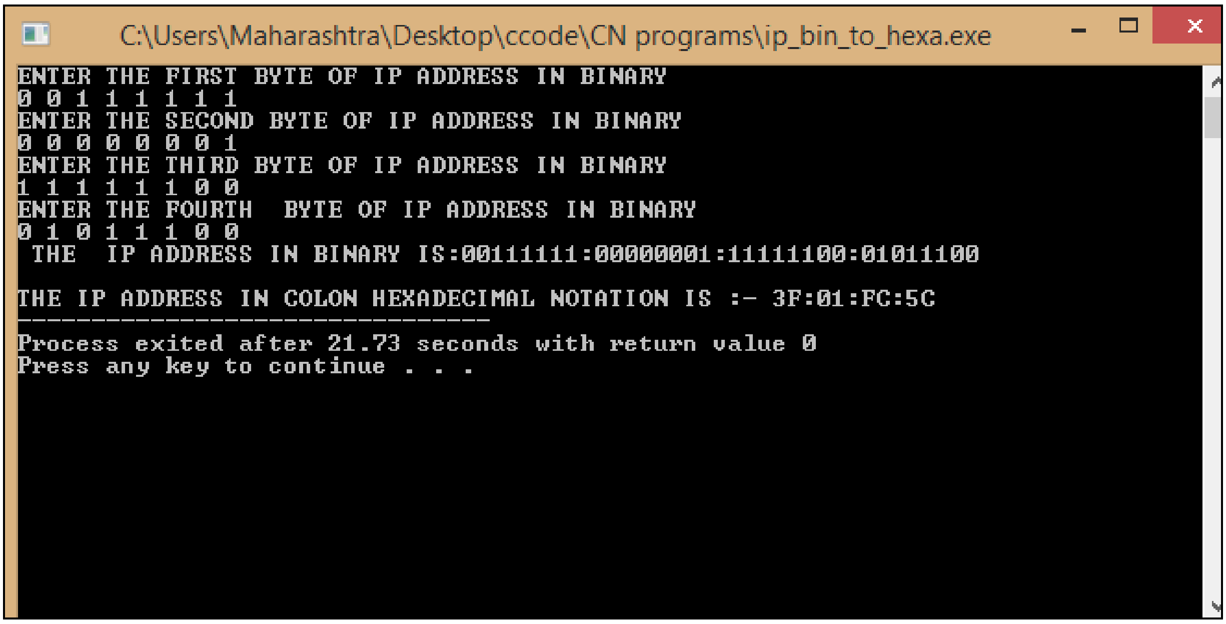
```
#include<stdio.h>
#include<math.h>

int main()
{          int
byte1[8],byte2[8],byte3[8],byte4[8],i,j=0,hd1=0,hd2=0,hd3=0,hd4=0,hd5=0,hd6=0,hd7=0,hd8=0;
char ch=':';
printf("ENTER THE FIRST BYTE OF IP ADDRESS IN BINARY \n");
for(i=7;i>=0;i--)
  {
scanf("%d",&byte1[i]);
}
printf("ENTER THE SECOND BYTE OF IP ADDRESS IN BINARY \n");
for(i=7;i>=0;i--)
  {
scanf("%d",&byte2[i]);
  }
printf("ENTER THE THIRD BYTE OF IP ADDRESS IN BINARY \n");
for(i=7;i>=0;i--)
  {
scanf("%d",&byte3[i]);
  }
printf("ENTER THE FOURTH  BYTE OF IP ADDRESS IN BINARY \n");
for(i=7;i>=0;i--)
  {
scanf("%d",&byte4[i]);
  }
printf(" THE  IP ADDRESS IN BINARY IS:");

while(j<4)
{
for(i=7;i>=0;i--)
```

```
  {
if(j==0)
printf("%d",byte1[i]);
else if(j==1)
printf("%d",byte2[i]);
else if(j==2)
printf("%d",byte3[i]);
else if (j==3)
printf("%d",byte4[i]);
else
break;
}
if(j!=3)
printf(":");
j++;
  }
for(i=0;i<4;i++)
{
hd1+=pow(2,i)*byte1[i];
hd2+=pow(2,i)*byte2[i];
hd3+=pow(2,i)*byte3[i];
hd4+=pow(2,i)*byte4[i];
}
for(i=4,j=0;i<8,j<4;i++,j++)
{
hd5+=pow(2,j)*byte1[i];
hd6+=pow(2,j)*byte2[i];
hd7+=pow(2,j)*byte3[i];
hd8+=pow(2,j)*byte4[i];
}
printf("\n \nTHE IP ADDRESS IN COLON HEXADECIMAL NOTATION IS :- ");
  printf("%X%X%c%X%X%c%X%X%c%X%X",hd5,hd1,ch,hd6,hd2,ch,hd7,hd3,ch,hd8,hd4);
return 0;

}
```

Output :-


C:\Users\Maharashtra\Desktop\ccode\CN programs\ip_bin_to_hexa.exe

```
ENTER THE FIRST BYTE OF IP ADDRESS IN BINARY
1 1 1 0 0 0 1 1
ENTER THE SECOND BYTE OF IP ADDRESS IN BINARY
1 1 1 1 1 1 1 1
ENTER THE THIRD BYTE OF IP ADDRESS IN BINARY
0 1 1 1 0 0 1 1
ENTER THE FOURTH  BYTE OF IP ADDRESS IN BINARY
0 0 1 1 1 1 0 1
 THE  IP ADDRESS IN BINARY IS:11100011:11111111:01110011:00111101

THE IP ADDRESS IN COLON HEXADECIMAL NOTATION IS :- E3:FF:73:3D
--------------------------------
Process exited after 20.88 seconds with return value 0
Press any key to continue . . .
```


C:\Users\Maharashtra\Desktop\ccode\CN programs\ip_bin_to_hexa.exe

```
ENTER THE FIRST BYTE OF IP ADDRESS IN BINARY
0 0 1 1 1 1 1 1
ENTER THE SECOND BYTE OF IP ADDRESS IN BINARY
0 0 0 0 0 0 0 1
ENTER THE THIRD BYTE OF IP ADDRESS IN BINARY
1 1 1 1 1 1 0 0
ENTER THE FOURTH  BYTE OF IP ADDRESS IN BINARY
0 1 0 1 1 1 0 0
 THE  IP ADDRESS IN BINARY IS:00111111:00000001:11111100:01011100

THE IP ADDRESS IN COLON HEXADECIMAL NOTATION IS :- 3F:01:FC:5C
--------------------------------
Process exited after 21.73 seconds with return value 0
Press any key to continue . . .
```