CRYPTOGRAPHY AND NETWORK SECURITY LAB PROGRAMS

Under the guidance of

Prof. S.Thejaswini, B.E., M.Tech

Submitted by:

Name	**USN**	**Section**
Divya Kumari	1SI13CS029	A

DEPARTMENT OF COMPUTER SCIENCE

Siddaganga Institute of Technology, Tumakuru - 572 103

(An autonomous institution affiliated to Visvesvaraya Technological University)

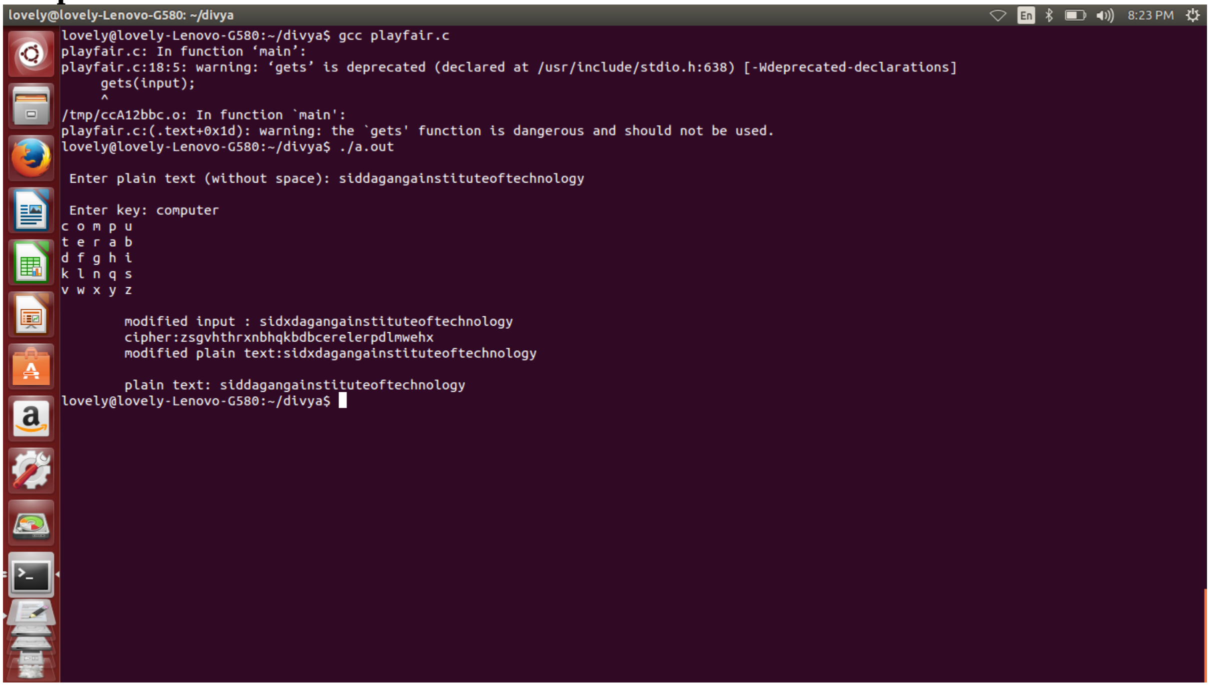
2016-17

1. **Write a program to perform the following using Playfair cipher technique**
    - **(i)** **Encrypt a given message M with different keys {k1,k2,…,kn}. Print key and cipher text pair**
    - **(ii)** **Decrypt the cipher texts obtained in (i) to get back M**

Program:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
void encrypt(char modify[20],char cipher[20],char mat[5][5]);
void decrypt(char cipher[20],char plain[20],char mat[5][5]);
void create_mat(char key[20],char mat[5][5]);
void modify_ip(char input[20],char modify[20]);
int main()
{
   char input[20],modify[20],cipher[20],plain[20];
   char key[20];
   char mat[5][5];
       printf("enter plain text without spaces\n");
        gets(input);
        printf("enter key\n");
        scanf("%s",key);
        modify_ip(input,modify);
        create_mat(key,mat);
        encrypt(modify,cipher,mat);
        decrypt(cipher,plain,mat);
        return 0;
}
void encrypt(char modify[20],char cipher[20],char mat[5][5])
{
        int clen=0,i,j,k,l;
        int c11,c12,c21,c22;
        char ch1,ch2;
        for(i=0;i<strlen(modify);i=i+2)
        {
                ch1=modify[i];
                ch2=modify[i+1];
                for(k=0;k<5;k++)
                        for(l=0;l<5;l++)
                        {
                                if(ch1==mat[k][l])
                                        c11=k,c12=l;
                                if(ch2==mat[k][l])
                                        c21=k,c22=l;
                        }
                if(c11==c21)
                {
                        cipher[clen++]=mat[c11][(c12+1)%5];
                        cipher[clen++]=mat[c21][(c22+1)%5];
                }
                else if(c12==c22)
                {
                        cipher[clen++]=mat[(c11+1)%5][c12];
                        cipher[clen++]=mat[(c21+1)%5][c22];
                }
                else
                {
                        cipher[clen++]=mat[c11][c22];
                        cipher[clen++]=mat[c21][c12];
                }
        }
        cipher[clen]='\0';
        printf("Cipher : %s\n",cipher);
}
void decrypt(char cipher[20],char plain[20],char mat[5][5])
{
        int plen=0,i,j,k,l;
        int c11,c12,c21,c22;
        char ch1,ch2;
        for(i=0;i<strlen(cipher);i=i+2)
        {
                ch1=cipher[i];
                ch2=cipher[i+1];
                for(k=0;k<5;k++)
                        for(l=0;l<5;l++)
                        {
                                if(ch1==mat[k][l])
                                        c11=k,c12=l;
                                if(ch2==mat[k][l])
                                        c21=k,c22=l;
                        }
                if(c11==c21)
                {
                        plain[plen++]=mat[c11][(c12-1)>0?(c12-1)%5:((c12+4)%5)];
                        plain[plen++]=mat[c21][(c22-1)>0?(c22-1)%5:((c22+4)%5)];
                }
                else if(c12==c22)
                {
                        plain[plen++]=mat[(c11-1)>0?(c11-1)%5:((c11+4)%5)][c12];
                        plain[plen++]=mat[(c21-1)>0?(c21-1)%5:((c21+4)%5)][c22];
                }
                else
                {
                        plain[plen++]=mat[c11][c22];
                        plain[plen++]=mat[c21][c12];
                }
        }
        plain[plen]='\0';
        printf("Plaintext : ");
        for(i=0;i<plen;i++)
        if(plain[i] != 'x')
        printf("%c",plain[i]);
        printf("\n");
}
void create_mat(char key[20],char mat[5][5])
{
        int i,j,k;
        int distinct[26]={0};
        for(i=0,k=0,j=0;i<strlen(key);i++)
        {
                if(!distinct[tolower(key[i])-'a'])
                {
                        mat[k][j++]=tolower(key[i]);
                        if(j==5)
                                k++,j=0;
        }
                if(key[i]=='j'||key[i]=='i')
                        distinct['j'-'a']=distinct['i'-'a']=1;
                else
                        distinct[tolower(key[i])-'a']=1;
        }
        for(i=0;i<26;i++)
        {
                if(!distinct[i])
                {
                        mat[k][j++]=i+'a';
                        if(j==5)
                                k++,j=0;
                        if(i+'a'=='i')
                                i++;
                }
        }
        for(i=0;i<5;i++)
        {
                for(j=0;j<5;j++)
                {
                        printf("%c",mat[i][j]);
                }
                printf("\n");
        }
}

void modify_ip(char input[20],char modify[20])
{
        int len=0,i=0;
        while(input[i]!='\0')
        {
                modify[len++]=input[i++];
                if(input[i]=='\0'||input[i]==input[i-1])
                {
                        modify[len++]='x';
                }
                else
                        modify[len++]=input[i++];
        }
        modify[len]='\0';
        printf("\n modified ip:%s\n",modify);
}
```

Output:



```
lovely@lovely-Lenovo-G580:~/divya$ gcc playfair.c
playfair.c: In function ‘main’:
playfair.c:18:5: warning: ‘gets’ is deprecated (declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]
     gets(input);
     ^
/tmp/ccA12bbc.o: In function `main':
playfair.c:(.text+0x1d): warning: the `gets' function is dangerous and should not be used.
lovely@lovely-Lenovo-G580:~/divya$ ./a.out

 Enter plain text (without space): siddagangainstituteoftechnology

 Enter key: computer
c o m p u
t e r a b
d f g h i
k l n q s
v w x y z

        modified input : sidxdagangainstituteoftechnology
        cipher:zsgvhthrxnbhqkbdbcerelerpdlmwehx
        modified plain text:sidxdagangainstituteoftechnology

        plain text: siddagangainstituteoftechnology
lovely@lovely-Lenovo-G580:~/divya$ 
```

2. **Write a program to perform the following using Hill cipher:**
   - **(i) Encrypt a message M with a given key matrix of size 2X2 and 3X3**
   - **(ii) Decrypt the cipher text obtained in (i) by computing inverse of the respective key matrix.**

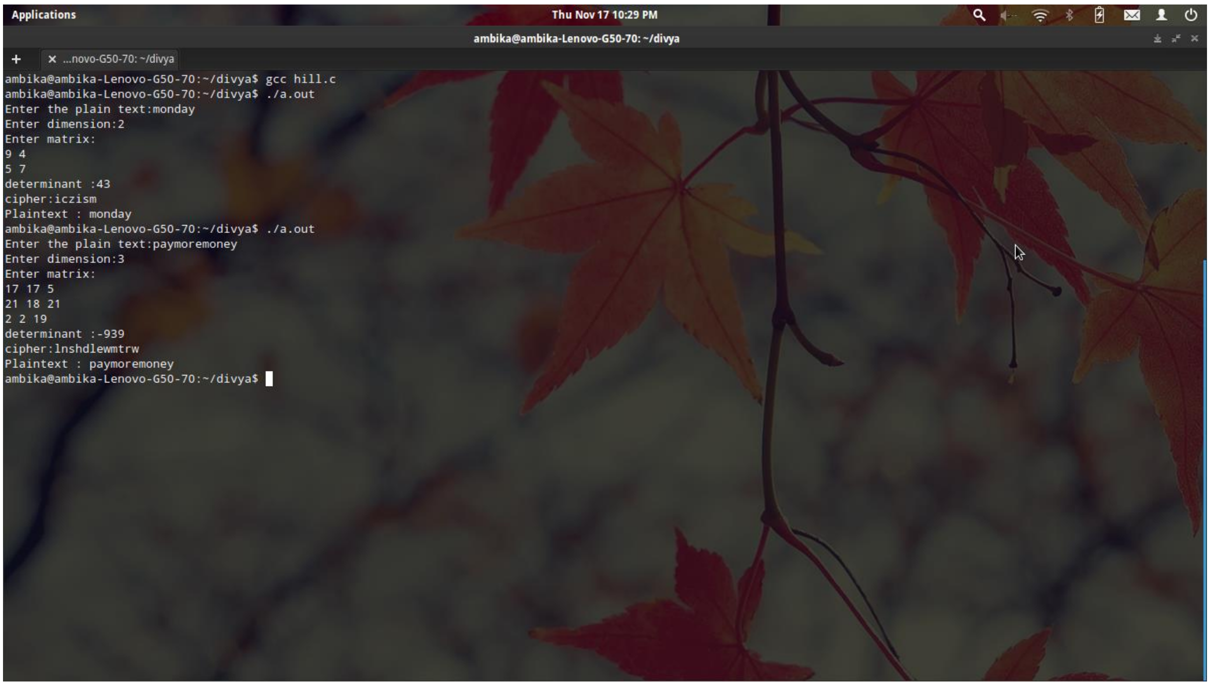
Program:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
void encrypt(char input[20],char cipher[20],int key[3][3],int dim)
{
        int i,j,k,clen, len, flag=0,l;
        int out[3][1], med[3][1];
        char inp[30];
        for(i=0,len=0;i<strlen(input);i++)
                if(input[i]!=' ' && isalpha(input[i]))
       inp[len++]=tolower(input[i]);
        inp[len]='\0';
        i=len=strlen(inp);
        while(len%dim!=0)
        {
                inp[i++]='x';
                len++;
                flag=1;
        }
        if(flag)
                inp[len]='\0';
        for(i=0,clen=0;i<strlen(inp);i=i+dim)
        {
                for(k=0;k<dim;k++)
                    med[k][0]=inp[i+k]-'a';
                for(k=0;k<dim;k++)
                {
                    out[k][0]=0;
                    for(l=0;l<dim;l++)
                        out[k][0]+=key[k][l]*med[l][0];
                }
                for(k=0;k<dim;k++)
                    cipher[clen++]=(out[k][0]%26)+'a';
        }
        cipher[clen]='\0';
        printf("cipher:%s\n",cipher);
}
void inverse(int key[3][3],int dim,int inv[3][3])
{
        int i,j,k,l,m,n;
        int cofact[3][3],med[2][2];
        int sign=1,det=0,temp;
        if(dim==2)
        {
                cofact[0][0]=key[1][1];
                cofact[0][1]=key[0][1]*-1;
                cofact[1][0]=key[1][0]*-1;
                cofact[1][1]=key[0][0];
                det=key[0][0]*key[1][1]- key[0][1]*key[1][0];
        }
        else
        {
                for(k=0;k<dim;k++)
                {
                    for(l=0;l<dim;l++)
                    {
                        m=0,n=0;
                        for(i=0;i<dim;i++)
                            if(i!=k)
                                for(j=0;j<dim;j++)
                                if(j!=l)
                                {
                                    med[m][n++]=key[i][j];
                                    if(n==2)
                                        m++,n=0;
                                }
                    cofact[k][l]=sign*(med[0][0]*med[1][1]-med[1][0]*med[0][1]);
```

```
                sign*=-1;
            }
        }
        for(i=0,det=0;i<dim;i++)
            det+=(cofact[0][i]*key[0][i]);
        for(i=0;i<dim-1;i++)
            for(j=i+1;j<dim;j++)
            {
                temp=cofact[i][j];
                cofact[i][j]=cofact[j][i];
                cofact[j][i]=temp;
            }
    }
    if(det==0)
    {
        printf("determinant is zero\n");
        exit(1);
    }
    printf("determinant :%d\n",det);
    if((det%2==0)||(det==13))
    {
        printf("(1/|%d|)mod26 cannot be found\n",det);
        exit(1);
    }
    n=1;
    while(n%det!=0)
        n+=26;
    n/=det;
    for(i=0;i<dim;i++)
        for(j=0;j<dim;j++)
        {
            inv[i][j]=(n*cofact[i][j])%26;
            while(inv[i][j]<0)
            {
                inv[i][j]+=26;
            }
        }
}
void decrypt(char cipher[20],char plain[20],int inv[3][3],int dim)
{
    int i,j,k,l,len,temp=0;
    int out[3][1],med[3][1];
    for(len=0,i=0;i<strlen(cipher);i+=dim)
{
        for(k=0;k<dim;k++)
            med[k][0]=cipher[i+k]-'a';
        for(k=0;k<dim;k++)
        {
            out[k][0]=0;
            for(l=0;l<dim;l++)
```

```
                        out[k][0]+=inv[k][l]*med[l][0];
                }
                for(k=0;k<dim;k++)
                        plain[len++]=(out[k][0]%26)+'a';
        }
        plain[len]='\0';
        printf("Plaintext : ");
        for(i=0;i<len;i++)
        if(plain[i] != 'x')
        printf("%c",plain[i]);
        printf("\n");
}
int main()
{
   char input[30], cipher[30], plain[30];
   int key[3][3],dim, inv[3][3];
        int i,j;
        printf("Enter the plain text:");
        gets(input);
        printf("Enter dimension:");
        scanf("%d",&dim);
        printf("Enter matrix:\n");
        for(i=0;i<dim;i++)
                for(j=0;j<dim;j++)
                        scanf("%d",&key[i][j]);
   inverse(key,dim,inv);
        encrypt(input,cipher,key,dim);
        decrypt(cipher,plain,inv,dim);
        return 0;
}
```

Output:



```
ambika@ambika-Lenovo-G50-70:~/divya$ gcc hill.c
ambika@ambika-Lenovo-G50-70:~/divya$ ./a.out
Enter the plain text:monday
Enter dimension:2
Enter matrix:
9 4
5 7
determinant :43
cipher:iczism
Plaintext : monday
ambika@ambika-Lenovo-G50-70:~/divya$ ./a.out
Enter the plain text:paymoremoney
Enter dimension:3
Enter matrix:
17 17 5
21 18 21
2 2 19
determinant :-939
cipher:lnshdlewmtrw
Plaintext : paymoremoney
ambika@ambika-Lenovo-G50-70:~/divya$ 
```

3. **Perform encryption and decryption using mono-alphabetic cipher. The program should support the following :**
    i. **Construct an input file named plaintext.txt (consisting of 1000 alphabets, without any space or special characters)**
    ii. **Encrypt the characters of plaintext.txt and store the corresponding ciphertext characters in ciphertext.txt**
    iii. **Compute the frequency of occurrence of each alphabet in both plaintext.txt and ciphertext.txt and tabulate the results as follows**

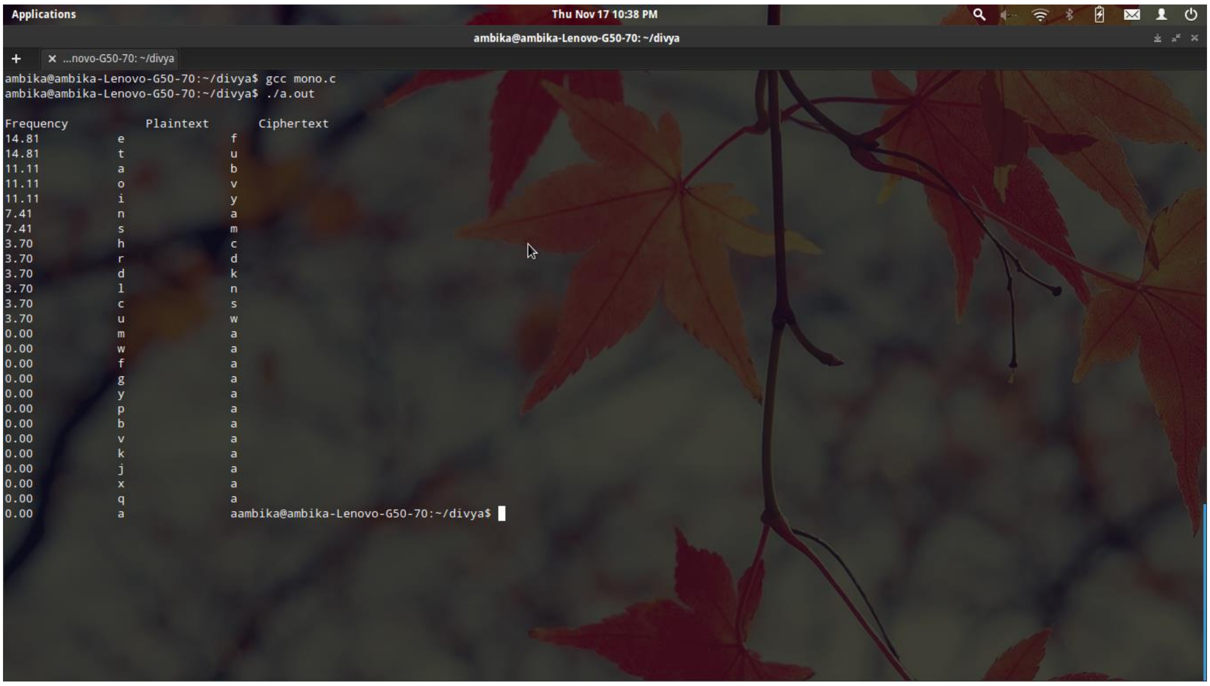
Frequency	**Plaintext character**	**Ciphertext character**
12.34	**A**	**X**
.	**.**	**.**
.	**.**	**.**

Program:

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
void analysis(int cf2[26]);
int max1();
int max2(int cf2[26]);
float
sf2[28]={8.167,1.492,2.782,4.253,12.702,2.228,2.015,6.094,6.996,0.153,0.773,4.025,2.406,6
.749,7.507,1.929,0.095,5.987,6.327,9.056,2.758,0.978,2.360,0.150,1.974,0,074};
int main()
{
   char key[26]={'g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','a','b','c','d','e','f'};
         FILE *fp1,*fp2;
         int temp,cf2[26]={0};
         char c1,c2;
         fp1=fopen("plain.txt","r");
         fp2=fopen("cipher.txt","w");
         while((c1=fgetc(fp1))!=EOF)
         {
                  if(isalpha(c1))
                  {
                           c2=key[tolower(c1)-'a'];
                  }
                  fputc(c2,fp2);
         }
   fclose(fp1);
   fclose(fp2);
   fp1=fopen("cipher.txt","r");
   fp2=fopen("result.txt","w");
   while((c1=fgetc(fp1))!=EOF)
         {
                if(isalpha(c1))
                {
                   temp=((c1-6)>=97)?(c1-6):(c1-20);
                        c2=temp;
                }
                fputc(c2,fp2);
        }
        fclose(fp1);
   fclose(fp2);
   analysis(cf2);
   return 0;
}
void analysis(int cf2[26])
{
        int i,index;
        char c;
        int ch1,ch2;
        FILE *fp;
        fp=fopen("cipher.txt","r");
        while((c=fgetc(fp))!=EOF)
        {
                if(isalpha(c))
                cf2[tolower(c)-'a']++;
        }
        int sum=0;
        float f;
        for(i=0;i<26;i++)
        sum+=cf2[i];
        printf("\nFrequency\t    Plaintext\t    Ciphertext");
        for(i=0;i<26;i++)
        {
                ch1=max1()+'a';
                ch2=max2(cf2)+'a';
                index=max2(cf2);
                f=(float)(cf2[index]*100)/sum;
                printf("\n%.2f\t\t%c\t\t%c",f,ch1,ch2);
                cf2[index]=0;
        }
        fclose(fp);
}
int max1()
{
        float max=0;
        int i,index=0;
        max=sf2[0];
        for(i=0;i<26;i++)
        {
                if(sf2[i]>max)
                {
                        index=i;
```

```
                                        max=sf2[i];
                                }
                        }
                        sf2[index]=0;
                        return index;
}
int max2(int cf2[26])
{
                        float max=0;
                        int i,index=0;
                        max=cf2[0];
                        for(i=0;i<26;i++)
                        {
                                if(cf2[i]>max)
                                {
                                        index=i;
                                        max=cf2[i];
                                }
                        }
                        return index;
}
```

Output:



```
ambika@ambika-Lenovo-G50-70:~/divya$ gcc mono.c
ambika@ambika-Lenovo-G50-70:~/divya$ ./a.out

Frequency          Plaintext        Ciphertext
14.81           e                f
14.81           t                u
11.11           a                b
11.11           o                v
11.11           i                y
7.41            n                a
7.41            s                m
3.70            h                c
3.70            r                d
3.70            d                k
3.70            l                n
3.70            c                s
3.70            u                w
0.00            m                a
0.00            w                a
0.00            f                a
0.00            g                a
0.00            y                a
0.00            p                a
0.00            b                a
0.00            v                a
0.00            k                a
0.00            j                a
0.00            x                a
0.00            q                a
0.00            a                aambika@ambika-Lenovo-G50-70:~/divya$
```

4. Write a program to perform encryption and decryption using transposition technique with column permutation given as key.

Program:

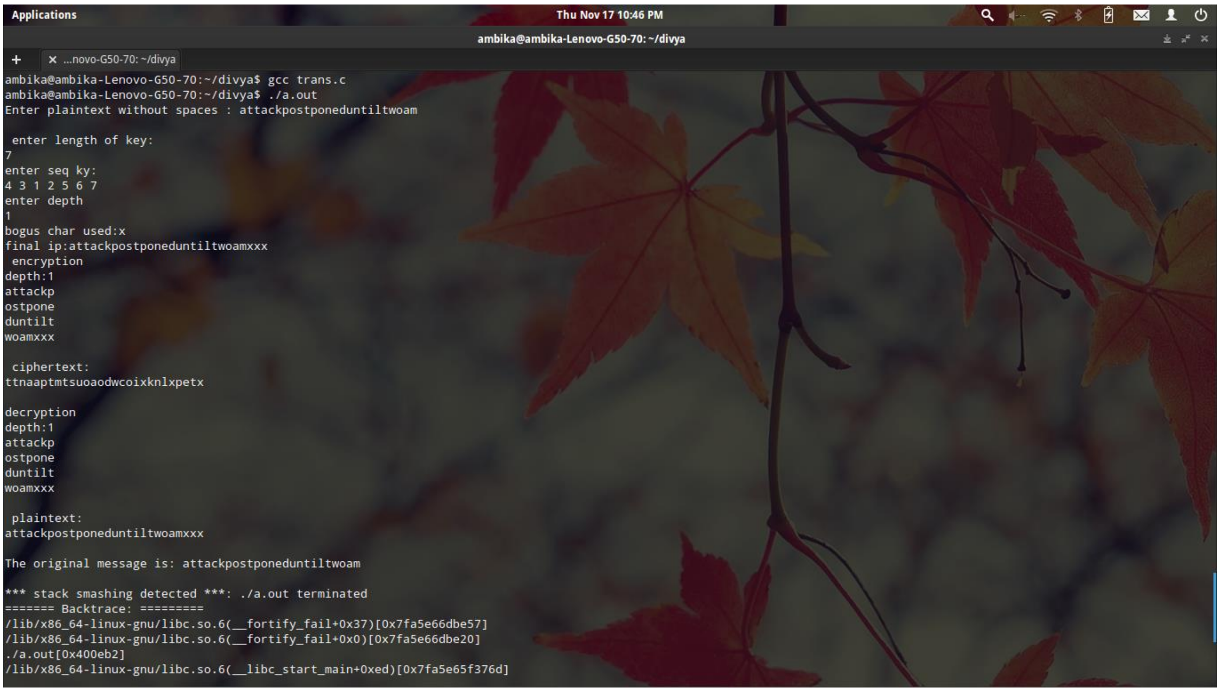
```
#include<stdio.h>
#include<string.h>
void encrypt(char input[20],char cipher[20],int d,int order[20],int l1)
{
```

```
int l2;
l2=strlen(input);
if(l2%l1!=0)
{
        while(l2%l1!=0)
                input[l2++]='x';
        input[l2]='\0';
        printf("bogus char used:%c\n",'x');
        printf("final ip:%s",input);
}
int r=l2/l1;
char p1[r][l1];
int count=0,k=1,i,j;
printf("\n encryption\n");
while(d>0)
{
        count=0;
        printf("depth:%d\n",k);
        k=k+1;
        for(i=0;i<r;i++)
        {
                for(j=0;j<l1;j++)
                {
                        p1[i][j]=input[count];
                        count=count+1;
                }
        }
        for(i=0;i<r;i++)
        {
                for(j=0;j<l1;j++)
                {
                        printf("%c",p1[i][j]);
                }
                printf("\n");
        }
        count=0;
        for(i=0;i<l1;++i)
        {
                for(j=0;j<r;++j)
                {
                        input[count]=p1[j][order[i]];
                        count=count+1;
                }
        }
        printf("\n ciphertext:\n");
        for(i=0;i<l2;i++)
                printf("%c",input[i]);
        printf("\n\n");
        d=d-1;
}
```

```
        for(i=0;i<l2;i++)
           cipher[i]=input[i];
         cipher[i]='\0';
}
void decrypt(char cipher[20],int d,int order[20],int l1)
{
        int l2=strlen(cipher);
        int r=l2/l1;
        char p1[r][l1];
        int count=0;
        int k1=1,i,j;
        printf("decryption\n");
        while(d>0)
        {
                count=0;
                printf("depth:%d\n",k1);
                k1=k1+1;
                for(i=0;i<l1;i++)
                {
                        for(j=0;j<r;j++)
                        {
                                p1[j][order[i]]=cipher[count];
                                count=count+1;
                        }
                }
                for(i=0;i<r;i++)
                {
                        for(j=0;j<l1;j++)
                        {
                                printf("%c",p1[i][j]);
                        }
                        printf("\n");
                }
                count=0;
                for(i=0;i<r;i++)
                {
                        for(j=0;j<l1;j++)
                        {
                                cipher[count]=p1[i][j];
                                count=count+1;
                        }
                }
                printf("\n plaintext:\n");
                for(i=0;i<l2;i++)
                        printf("%c",cipher[i]);
                printf("\n\n");
                d=d-1;
        }
        printf("The original message is: ");
        for(i=0;i<l2;i++)
```

```
    if(cipher[i] != 'x')
    printf("%c",cipher[i]);
    printf("\n");
}
int main()
{
    int l1,i,d,j;
    char input[20],cipher[20];
    printf("Enter plaintext without spaces : ");
    gets(input);
    printf("\n enter length of key:\n");
    scanf("%d",&l1);
    int sequence[l1];
    printf("enter seq ky:\n");
    for(i=0;i<l1;++i)
    {
        scanf("%d",&sequence[i]);
    }
    int order[l1];
    for(i=1;i<=l1;++i)
    {
        for(j=0;j<l1;++j)
        {
            if(sequence[j]==i)
                order[i-1]=j;
        }
    }
    printf("enter depth\n");
    scanf("%d",&d);
    int d1=d;
    encrypt(input,cipher,d,order,l1);
    decrypt(cipher,d1,order,l1);
    printf("\n");
  return 0;
}
```

Output:



```
ambika@ambika-Lenovo-G50-70:~/divya$ gcc trans.c
ambika@ambika-Lenovo-G50-70:~/divya$ ./a.out
Enter plaintext without spaces : attackpostponeduntiltwoam

 enter length of key:
7
enter seq ky:
4 3 1 2 5 6 7
enter depth
1
bogus char used:x
final ip:attackpostponeduntiltwoamxxx
 encryption
depth:1
attackp
ostpone
duntilt
woamxxx

 ciphertext:
ttnaaptmtsuoaodwcoixknlxpetx

decryption
depth:1
attackp
ostpone
duntilt
woamxxx

 plaintext:
attackpostponeduntiltwoamxxx

The original message is: attackpostponeduntiltwoam

*** stack smashing detected ***: ./a.out terminated
======= Backtrace: =========
/lib/x86_64-linux-gnu/libc.so.6(__fortify_fail+0x37)[0x7fa5e66dbe57]
/lib/x86_64-linux-gnu/libc.so.6(__fortify_fail+0x0)[0x7fa5e66dbe20]
./a.out[0x400eb2]
/lib/x86_64-linux-gnu/libc.so.6(__libc_start_main+0xed)[0x7fa5e65f376d]
```

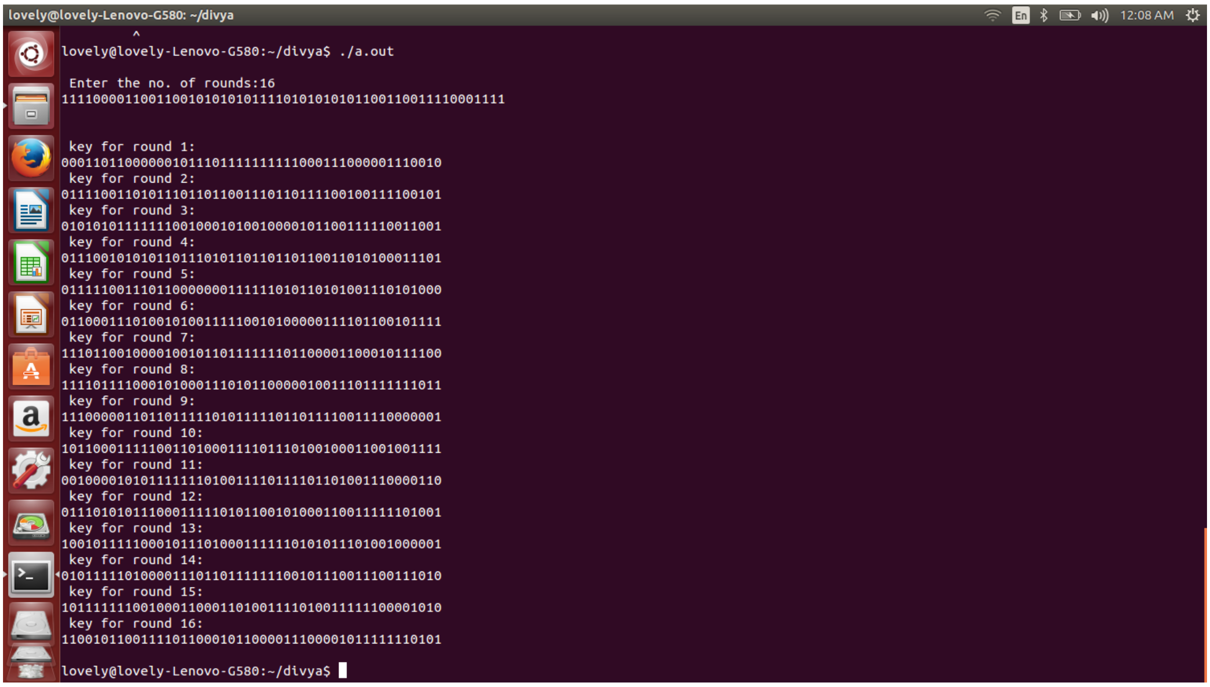
5. Generate and print 48-bit keys for all sixteen rounds of DES algorithm, given a 64-bit initial key.

Program:

```
#include<stdio.h>
#include<math.h>
int pc1_key[56],c[28],d[28],keyshift[56],pc2_key[48];
int key[64]={0,0,0,1,0,0,1,1,
             0,0,1,1,0,1,0,0,
             0,1,0,1,0,1,1,1,
             0,1,1,1,1,0,0,1,
             1,0,0,1,1,0,1,1,
             1,0,1,1,1,1,0,1,
             1,1,0,1,1,1,1,1,
             1,1,1,1,0,0,0,1};
int pc1[8][7]={57, 49, 41, 33,25,17,9,
                1,58,50,42 ,34,26,18,
                10,2,59,51,43 ,35,27,
                19,11,3,60,52,44,36,
                63,55,47,39,31,23,15,
                7,62,54,46, 38,30,22,
                14,6,61,53,45,37,29,
                 21,13,5, 28,20,12,4};
 int pc2[8][6]={14,17,11,24, 1, 5,
                 3, 28, 15, 6, 21, 10,
                 23,19,12,4,26,8,
                 16,7,27,20,13,2,
                 41,52,31,37,47,55,
                 30,40,51,45,33,48,
                   44,49,39,56,34,53,
                   46,42,50,36,29,32};
void PC1()
{
    int i,j,k=0;
    for(i=0;i<8;i++)
      for(j=0;j<7;j++)
         pc1_key[k++]=key[pc1[i][j]-1];
   for(i=0;i<56;i++)
      printf("%d",pc1_key[i]);
   printf("\n\n");
}
void leftshift()
{
   int i,j,k=0;
   k=c[0];
   for(i=1;i<28;i++)
      c[i-1]=c[i];
   c[i-1]=k;
   k=d[0];
   for(i=1;i<28;i++)
       d[i-1]=d[i];
   d[i-1]=k;
   for(i=0;i<28;i++)
      keyshift[i]=c[i];

  for(k=0,j=i;j<56;j++)
      keyshift[i++]=d[k++];
}
void PC2()
{
     int i,j,k=0;
     for(i=0;i<8;i++)
       for(j=0;j<6;j++)
          pc2_key[k++]=keyshift[pc2[i][j]-1];
}
void main()
{
   int i,j,k=0,round=1,n;
   printf("\n Enter the no. of rounds:");
   scanf("%d",&n);
   PC1();
   for(i=0;i<28;i++)
       c[i]=pc1_key[i];
   for(j=i;j<56;j++)
    d[k++]=pc1_key[j];
   while(round<=n)
   {
      if(round==1||round==2||round==9||round==16)
          leftshift();
```

```
        else
        {
            leftshift();
            leftshift();
        }
        PC2();
        printf("\n key for round %d:\n",round);
        for(i=0;i<48;i++)
            printf("%d",pc2_key[i]);
        round++;
    }
    printf("\n\n");
}
```

Output:



```
lovely@lovely-Lenovo-G580: ~/divya
          ^
lovely@lovely-Lenovo-G580:~/divya$ ./a.out

 Enter the no. of rounds:16
1111000011001100101010101111010101010110011001111000111 1


 key for round 1:
000110110000001011101111111111000111000001110010
 key for round 2:
011110011010111011011001110110111100100111100101
 key for round 3:
010101011111110010001010010000101100111110011001
 key for round 4:
011100101010110111010110110110110011010100011101
 key for round 5:
011111001110110000000111111010110101001110101000
 key for round 6:
011000111010010100111110010100000111101100101111
 key for round 7:
111011001000010010110111111101100001100010111100
 key for round 8:
111101111000101000111010110000001001110111111011
 key for round 9:
111000001101101111101011111011011110011110000001
 key for round 10:
101100011111001101000111101110100100011001001111
 key for round 11:
001000010101111111010011110111101101001110000110
 key for round 12:
011101010111000111110101100101000110011111101001
 key for round 13:
100101111100010111010001111110101011101001000001
 key for round 14:
010111110100001110110111111100101110011100111010
 key for round 15:
101111111001000110001101001111010011111100001010
 key for round 16:
110010110011110110001011000011100001011111110101

lovely@lovely-Lenovo-G580:~/divya$ █
```

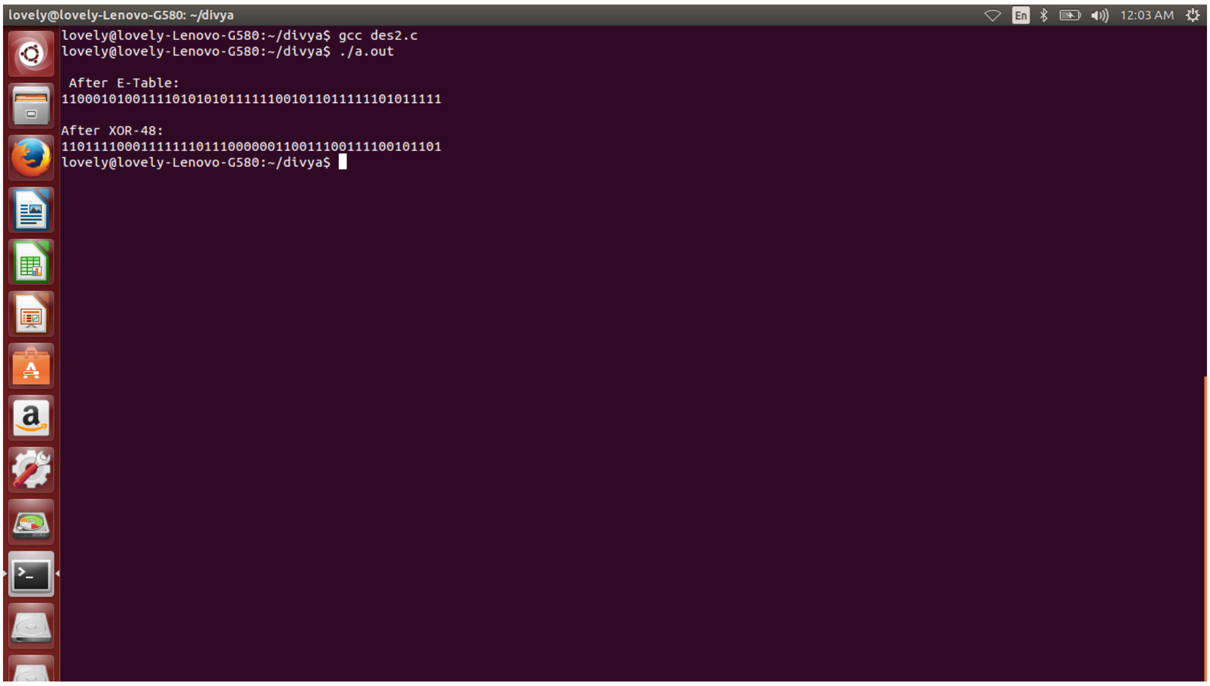
6. **Given 64-bit output of (i-1)th round of DES, 48-bit ith round key K_i and E table, find the 48-bit input for S-box.**

Program:

```
#include<stdio.h>
int l[32],r[32],er[48];
int pc2_key[48]=
            {
              0,0,0,1,1,0,1,
              1,0,0,0,0,0,0,
              1,0,1,1,1,0,1,
              1,1,1,1,1,1,1,
              1,1,0,0,0,1,1,
              1,0,0,0,0,0,1,
                   1,1,0,0,1,0
                };
int pt[64]=
                {
                   0,0,0,0,0,0,0,1,
                   0,0,1,0,0,0,1,1,
                   0,1,0,0,0,1,0,1,
                   0,1,1,0,0,1,1,1,
                   1,0,0,0,1,0,0,1,
                   1,0,1,0,1,0,1,1,
                   1,1,0,0,1,1,0,1,
                   1,1,1,0,1,1,1,1,
               };
int e_bit[8][6]=
                 {
                   32,1,2,3,4,5,
                   4,5,6,7,8,9,
                   8,9,10,11,12,13,
                   12,13,14,15,16,17,
                   16,17,18,19,20,21,
                   20,21,22,23,24,25,
                   24,25,26,27,28,29,
                   28,29,30,31,32,1
                 };
void etable()
{
  int i,j,k=0;
  for(i=0;i<8;i++)
  {
     for(j=0;j<6;j++)
        er[k++]=r[e_bit[i][j]-1];
  }
}
void xor48()
{
    int i;
     for(i=0;i<48;i++)
     {
       if(er[i]==pc2_key[i])
             er[i]=0;
       else
             er[i]=1;
     }
}
void main()
{
   int i,j,k=0;
   for(i=0;i<32;i++)
        l[i]=pt[i];
   for(j=i;j<64;j++)
```

```
        r[k++]=pt[j];
  etable();
  printf("\n After E-Table:\n");
  for(i=0;i<48;i++)
       printf("%d",er[i]);
  printf("\n");
  xor48();
  printf("\nAfter XOR-48:\n");
  for(i=0;i<48;i++)
       printf("%d",er[i]);
  printf("\n");
}
```

Output:



```
lovely@lovely-Lenovo-G580:~/divya$ gcc des2.c
lovely@lovely-Lenovo-G580:~/divya$ ./a.out

 After E-Table:
110001010011110101010111111001011011111101011111

After XOR-48:
110111100011111110111000000110011100111100101101
lovely@lovely-Lenovo-G580:~/divya$ 
```

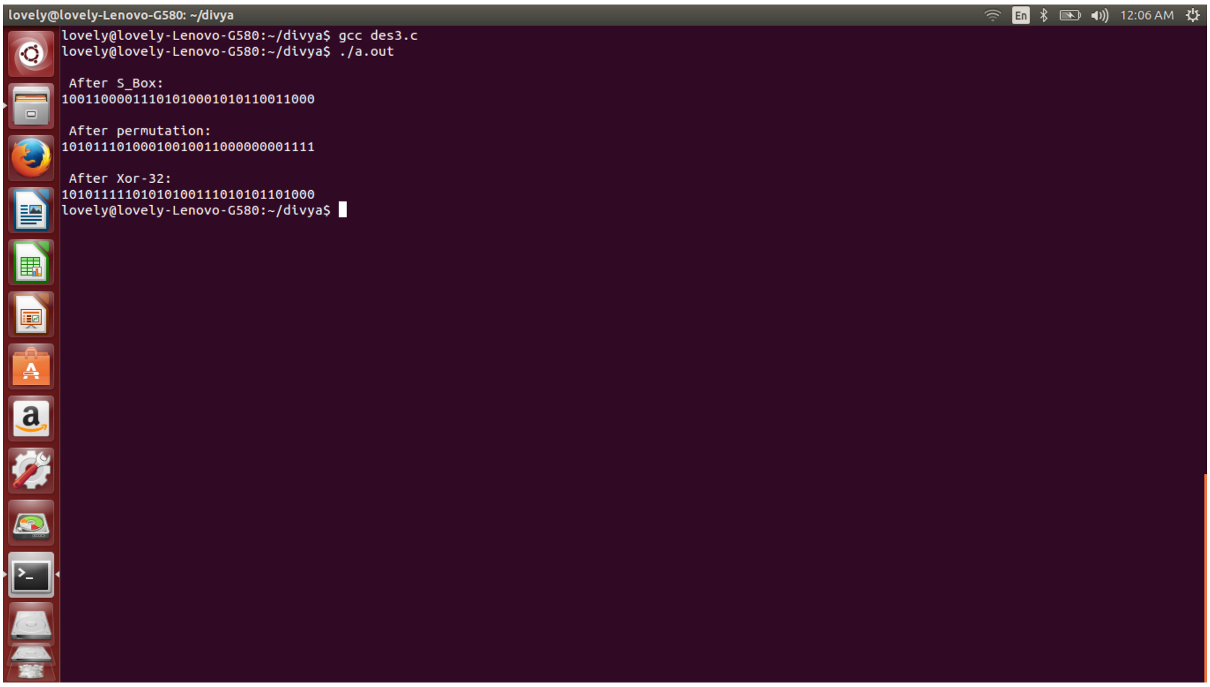
7. **Given 48-bit input to S-box and permutation table P, find the 32-bit output R_i of i^{th} round of DES algorithm.**

Program:

```
#include<stdio.h>
int sbox[32],sbp[32],s[8][6],si=0,r[32];
int
er[48]={0,1,1,0,0,1,1,0,0,0,1,1,1,1,1,1,1,0,1,1,1,0,0,0,0,0,0,1,1,0,0,1,1,1,0,0,1,1,1,1,0,0,1,0,1,
1,0,1};
int l[32]={0,0,0,0,0,0,0,1,
       0,0,1,0,0,0,1,1,
       0,1,0,0,0,1,0,1,
       0,1,1,0,0,1,1,1};

int sbox_table[8][4][16]={
    14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,
    0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8,
    4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0,
    15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13,

    15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,
    3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5,
    0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15,
    13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9,

    10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,
    13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1,
    13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7,
    1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12,

    7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,
    13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9,
    10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4,
    3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14,

    2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,
    14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6,
    4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14,
    11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3,

    12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,
    10,15,4,2,7,12,9,5,6,1,12,14,0,11,3,8,
    9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6,
    4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13,

    4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,
    13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6,
    1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2,
    6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12,

    13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,
    1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2,
    7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8,
    2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11};
int ptable[4][8]={
    16,7,20,21,29,12,28,17,
    1,15,23,26,5,18,31,10,
    2,8,24,14,32,27,3,9,
    19,13,30,6,22,11,4,25};
void sbpermute()
{
  int i,j,k=0;
  for(i=0;i<4;i++)
   for(j=0;j<8;j++)
      sbp[k++]=sbox[ptable[i][j]-1];
```

```
}
void xor32()
{
  int i;
  for(i=0;i<32;i++)
  {
    if(l[i]==sbp[i])
        r[i]=0;
    else
         r[i]=1;
  }
}
void dec_bin(int n)
{
  int a[4],i=3,j=0,rem=0;
  for(j=0;j<4;j++)
    a[j]=0;
  while(n!=0)
  {
    a[i]=n%2;
    i--;
    n/=2;
  }
  for(j=0;j<4;j++)
     sbox[si++]=a[j];
}
void sboxf()
{
  int i,j,k=0,row,col;
  for(i=0;i<8;i++)
    for(j=0;j<6;j++)
      s[i][j]=er[k++];
  k=0;
  while(k<8)
  {
   for(i=0;i<8;i++)
   {
     row=(s[i][0]<<1)+s[i][5];
     col=(s[i][1]<<3)+(s[i][2]<<2)+(s[i][3]<<1)+s[i][4];
     dec_bin(sbox_table[k++][row][col]);
   }
  }
}
void main()
{
    int i;
    sboxf();
    printf("\n After S_Box:\n");
    for(i=0;i<32;i++)
      printf("%d",sbox[i]);
    printf("\n");
    sbpermute();
    printf("\n After permutation:\n");
    for(i=0;i<32;i++)
      printf("%d",sbp[i]);
    printf("\n");
    xor32();
     printf("\n After Xor-32:\n");
    for(i=0;i<32;i++)
      printf("%d",r[i]);
    printf("\n");

}
```

Output:



```
lovely@lovely-Lenovo-G580:~/divya$ gcc des3.c
lovely@lovely-Lenovo-G580:~/divya$ ./a.out

 After S_Box:
10011000011101010001010110011000

 After permutation:
10101110100010010011000000001111

 After Xor-32:
10101111101010100111010101101000
lovely@lovely-Lenovo-G580:~/divya$ 
```

8. **Implement the following with respect to RC4:**
   - **i. Print first *n* key bytes generated by key generation process.**
   - **ii. Illustrate encryption/decryption by accepting one byte data as input on the above generated keys.**

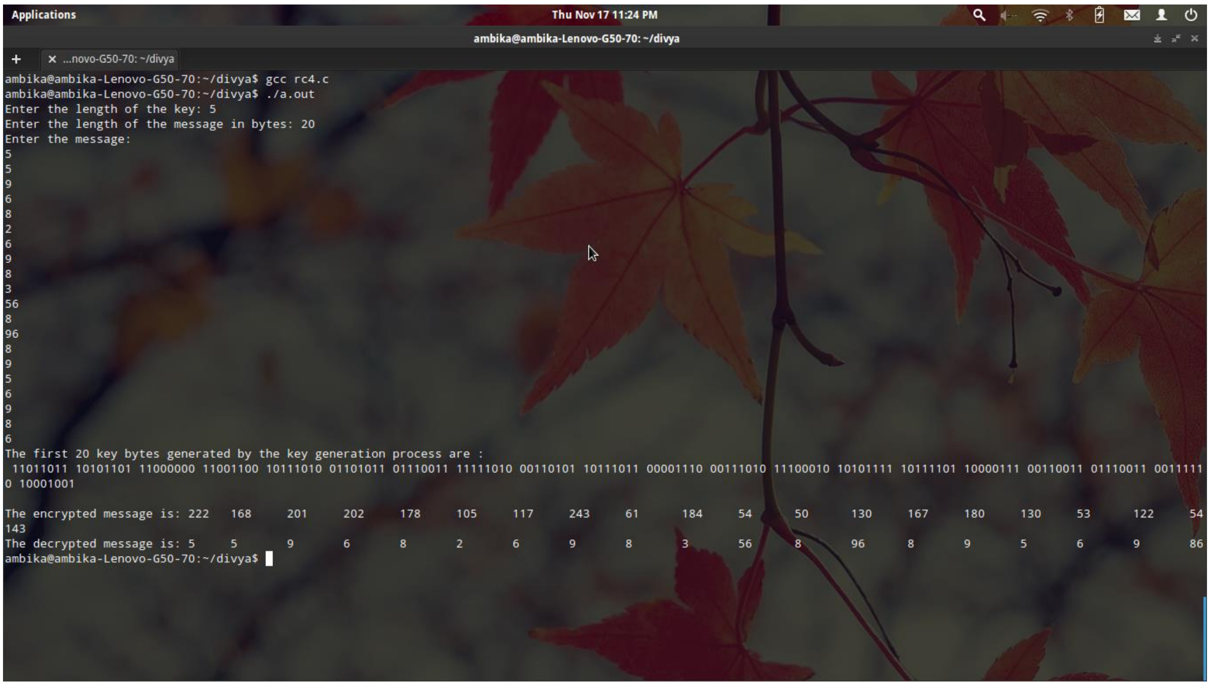
Program:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>
#include<ctype.h>
void swap(int *a,int *b)
{
        int t=*a;
```

```
      *a=*b;
      *b=t;
}
void init(int key[],int len,int s[])
{
      int i,j,t[256];
      for(i=0;i<256;i++)
      {
            s[i]=i;
            t[i]=key[i%len];
      }
      for(i=0,j=0;i<256;i++)
      {
            j=(j+s[i]+t[j])%256;
            swap(&s[i],&s[j]);
      }
}
void print_key(int key[],int len)
{
  int i,j,t,temp,k=0;
  int keystream[len*8];
  int bits[8];
      printf("The first %d key bytes generated by the key generation process are : \n",len);
  for(i=0;i<len;i++)
  {
    temp=key[i];
    for(j=7;j>=0;j--)
    {
      if(temp>0)
      {
        bits[j]=temp%2;
        temp/=2;
      }
      else
      bits[j]=0;
    }
    for(j=0;j<8;j++)
    keystream[k++]=bits[j];
  }
  for(i=0;i<(len*8);i++)
  {
    if((i%8) == 0)
    printf(" ");
    printf("%d",keystream[i]);
  }
  printf("\n");
}
void stream(int s[],int len,int key[])
{
      int i=0,j=0,k=0;
```

```
        while(k<len)
        {
                i = (i+1)%256;
                j=(j+s[i])%256;
                swap(&s[i],&s[j]);
                key[k++]=s[(s[i]+s[j])%256];
        }
        print_key(key,len);
}
void encrypt(int p[],int key[],int len,int c[])
{
        int i;
        for(i=0;i<len;i++)
                c[i]=p[i]^key[i];
        printf("\nThe encrypted message is: ");
        for(i=0;i<len;i++)
                printf("%d\t",c[i]);
}
void decrypt(int p[],int key[],int len,int c[])
{
        int i;
        for(i=0;i<len;i++)
                p[i]=c[i]^key[i];
        printf("\nThe decrypted message is: ");
        for(i=0;i<len;i++)
                printf("%d\t",p[i]);
}
void main()
{
                int s[256],k[256],len,key[500],p[500],c[500];
                int i;
                printf("Enter the length of the key: ");
                scanf("%d",&len);
                /*printf("Enter the key : ");
                for(i=0;i<len;i++)
                        scanf("%d",&k[i]);*/
     srand(time(NULL));
     for(i=0;i<len;i++)
        k[i]=rand()%256 + 1;
                init(k,len,s);
                printf("Enter the length of the message in bytes: ");
                scanf("%d",&len);
                printf("Enter the message: \n");
                for(i=0;i<len;i++)
                        scanf("%d",&p[i]);
                stream(s,len,key);
                encrypt(p,key,len,c);
                decrypt(p,key,len,c);
}
```

Output:



```
ambika@ambika-Lenovo-G50-70:~/divya$ gcc rc4.c
ambika@ambika-Lenovo-G50-70:~/divya$ ./a.out
Enter the length of the key: 5
Enter the length of the message in bytes: 20
Enter the message:
5
5
9
6
8
2
6
9
8
3
56
8
96
8
9
5
6
9
8
6
The first 20 key bytes generated by the key generation process are :
 11011011 10101101 11000000 11001100 10111010 01101011 01110011 11111010 00110101 10111011 00001110 00111010 11100010 10101111 10111101 10000111 00110011 01110011 0011111
0 10001001

The encrypted message is: 222   168    201    202    178    105    117    243    61     184    54     50     130    167    180    130    53     122    54
143
The decrypted message is: 5     5      9      6      8      2      6      9      8      3      56     8      96     8      9      5      6      9      86
ambika@ambika-Lenovo-G50-70:~/divya$ 
```

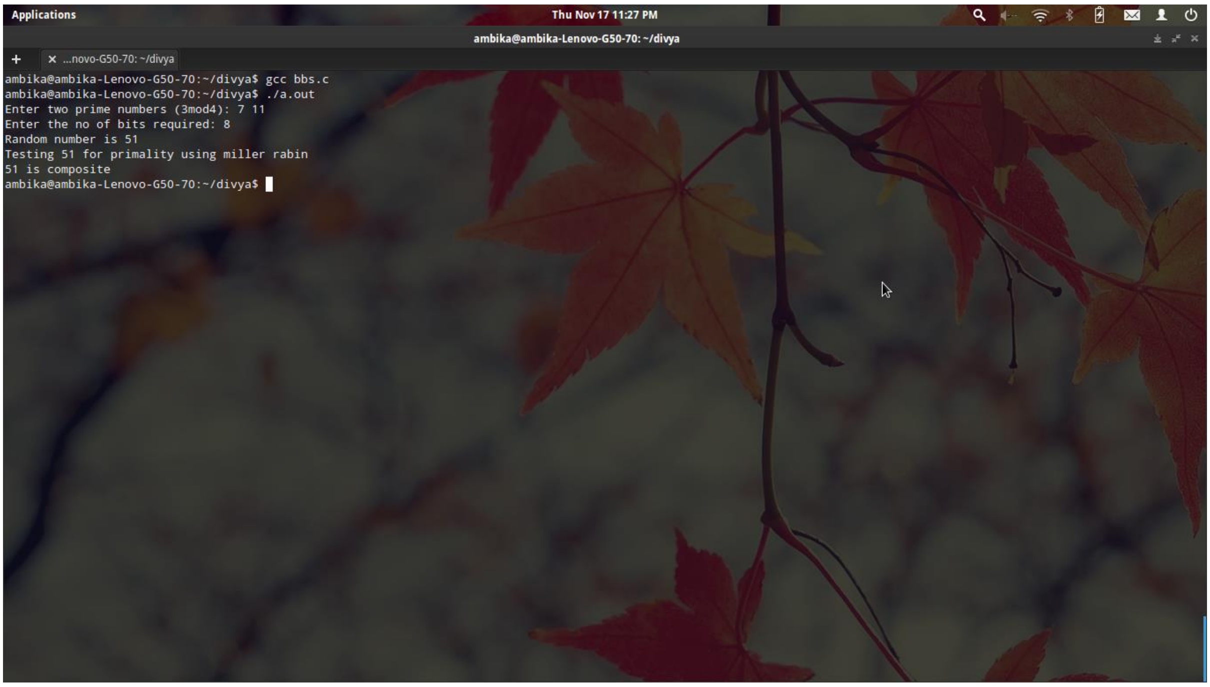
9. **Write a program to generate large random number using BBS random number generator algorithm and check whether the generated number is prime or not using RABIN-MILLER primality testing algorithm.**

Program:

```
#include<stdio.h>
#include<time.h>
int gcd(int a,int b)
{
            int n=1;
            while(n)
            {
                n=a%b;
                a=b;
                b=n;
            }
            return a;
}
void test(unsigned long long int n)
{
    unsigned long long int k=0,q=1,x,i,j,y,w;
    unsigned long long int z=n-1;
    unsigned long long int a;
    while(z%2==0)
    {
       z=z/2;
       k++;
    }
```

```
   q=z;
    a=rand()%(n-1);
    if(a==1) a++;
    x=1;
    for(i=1;i<=q;i++)
    {
        x*=a;
        x%=n;
    }
    if(x==1)
    {
        printf("%llu is Inconclusive\n",n);
        return;
    }
    x=1;
    for(i=0;i<k;i++)
    {
        x=1,y=1;
        for(j=1;j<=i;j++)
        {
            x*=2;
        }
        x*=q;
        for(w=1;w<=x;w++)                              // a^((2^i)*q)%n
        {
            y*=a;
            y%=n;
        }
        if(y==(n-1))
        {
            printf("%llu is Inconclusive\n",n);
            return;
        }
    }
    printf("%llu is composite\n",n);
}
void BlumBlumShub()
{
    int p,q,no,i;
    unsigned long long int n,s,res=0,x;
    srand(time(NULL));
    printf("Enter two prime numbers (3mod4): ");
    scanf("%d%d",&p,&q);
    n=p*q;
    printf("Enter the no of bits required: ");
    scanf("%d",&no);
    do{
        s= rand();
    }while((gcd(s,n)!=1)&&(s==1));
    x=(s*s)%n;
    for(i=1;i<=no;i++)
    {
        x = (x*x)%n;
        res = (res<<1)|(x&1);
    }
    printf("Random number is %llu\n",res);
    printf("Testing %llu for primality using miller rabin\n",res);
    test(res);
}
void main()
{
    BlumBlumShub();
}
```

Output:



```
ambika@ambika-Lenovo-G50-70:~/divya$ gcc bbs.c
ambika@ambika-Lenovo-G50-70:~/divya$ ./a.out
Enter two prime numbers (3mod4): 7 11
Enter the no of bits required: 8
Random number is 51
Testing 51 for primality using miller rabin
51 is composite
ambika@ambika-Lenovo-G50-70:~/divya$ 
```

10. **Implement RSA algorithm using client-server concept. The program should support the following :**
    - **i. Client generates {PU, PR} and distributes PU to Server.**
    - **ii. Sever encrypts message M using client's public key {PU}.**
    - **iii. Client decrypts the message sent by server using its private key {PR}.**

Program:

rsac.c

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

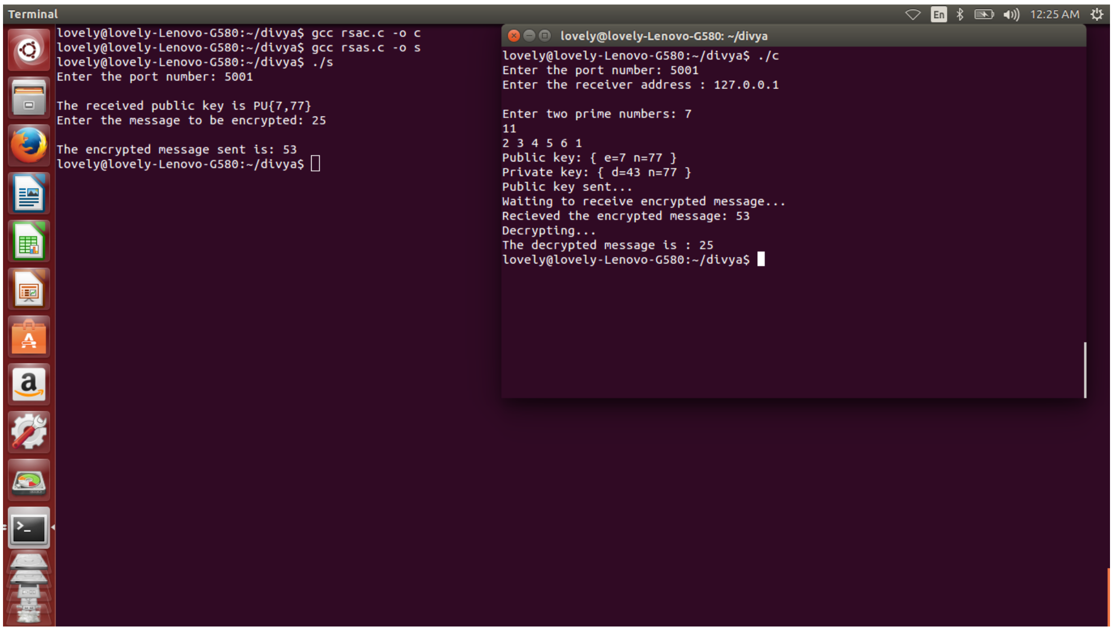
```
#include<netinet/in.h>
typedef long int int32;
int32 gcd(int32 a,int32 b)
{
        int32 n=1;
        while(n)
        {
                n= a%b;
                a=b;
                b=n;
        }
        return a;
}
void keygen(int32 *d,int32 *e,int32 *n)
{
   int32 phi,s;
   int32 t,p,q;
   printf("\nEnter two prime numbers: ");
   scanf("%ld%ld",&p,&q);

   *n = p*q;
   phi=(p-1)*(q-1);
   *e=1;
   do
   {
        (*e)++;
        t=gcd(phi,*e);
        printf("%ld ",t);
   }while(t!=1&& (*e)<phi);
   *d = 0;
   do
   {
      (*d)++;
      s = ((*d)*(*e))%phi;
   }while(s!=1);
   printf("\nPublic key: { e=%ld n=%ld }",*e,*n);
   printf("\nPrivate key: { d=%ld n=%ld }\n",*d,*n);
}
int32 decryptencrypt(int32 key,int32 msg,int32 n)
{       int32 k;
        int32 j;
   k=1;
   for(j=0;j<key;j++)
   {
                k=k*msg;
                k=k%n;
   }
   return k;
}
void main()
```

```
{
        int32 d,e,n;
        int sockfd,clen;
        int32 msg,en,m;
        int32 pu[2];
        struct sockaddr_in server;
        int port;
        char host[15];
        bzero((char*)&server,sizeof(server));
        printf("Enter the port number: ");
        scanf("%d",&port);
        printf("Enter the receiver address : ");
        scanf("%s",host);
        sockfd = socket(AF_INET,SOCK_STREAM,0);
        server.sin_family = AF_INET;
        server.sin_port = htons(port);
        server.sin_addr.s_addr = inet_addr(host);
        connect(sockfd,(struct sockaddr*)&server,sizeof(server));
        keygen(&d,&e,&n);
        pu[0] = e;
        // PU{e,n}
        pu[1] = n;
        send(sockfd,(char*)pu,sizeof(pu),0);
        printf("Public key sent...\n");
        printf("Waiting to receive encrypted message...\n");
        recv(sockfd,&en,sizeof(en),0);
        printf("Recieved the encrypted message: %ld \n",en);
        printf("Decrypting...\n");
        m = decryptencrypt(d,en,n);
        printf("The decrypted message is : %ld \n",m);
        close(sockfd);
}
```

rsas.c

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<netinet/in.h>
typedef long int int32;
int32 decryptencrypt(int32 key,int32 msg,int32 n)
{
        int32 k;
        int32 i=0,j;
   k=1;
   for(j=0;j<key;j++)
   {
                k=k*msg;
                k=k%n;
   }
   return k;
}
void main()
{
        int32 msg,en;
        int32 e,n;
        int sockfd,newsockfd,port,clen,y=1;
        int32 pu[2];
        struct sockaddr_in server,client;
        clen = sizeof(client);
        printf("Enter the port number: ");
        scanf("%d",&port);
        sockfd = socket(AF_INET,SOCK_STREAM,0);
        server.sin_family = AF_INET;
        server.sin_port = htons(port);
        server.sin_addr.s_addr = INADDR_ANY;
        setsockopt(sockfd,SOL_SOCKET,SO_REUSEADDR,&y,sizeof(int));
        bind(sockfd,(struct sockaddr * )&server, sizeof(server) );
        listen(sockfd,1);
        bzero((char*)&client,clen);
        newsockfd = accept(sockfd, (struct sockaddr *)&client, &clen);
        close(sockfd);
        recv(newsockfd,(char*)pu,sizeof(pu),0);
        printf("\nThe received public key is PU{%ld,%ld}",pu[0],pu[1]);
        e=pu[0],n=pu[1];
        printf("\nEnter the message to be encrypted: ");
        scanf("%ld",&msg);
        en = decryptencrypt(e,msg,n);
        printf("\nThe encrypted message sent is: %ld\n",en);
        send(newsockfd,(char*)&en,sizeof(en),0);
        close(sockfd);
}
```

Output:



```
lovely@lovely-Lenovo-G580:~/divya$ gcc rsac.c -o c
lovely@lovely-Lenovo-G580:~/divya$ gcc rsas.c -o s
lovely@lovely-Lenovo-G580:~/divya$ ./s
Enter the port number: 5001

The received public key is PU{7,77}
Enter the message to be encrypted: 25

The encrypted message sent is: 53
lovely@lovely-Lenovo-G580:~/divya$ 
```

```
lovely@lovely-Lenovo-G580:~/divya$ ./c
Enter the port number: 5001
Enter the receiver address : 127.0.0.1

Enter two prime numbers: 7
11
2 3 4 5 6 1
Public key: { e=7 n=77 }
Private key: { d=43 n=77 }
Public key sent...
Waiting to receive encrypted message...
Recieved the encrypted message: 53
Decrypting...
The decrypted message is : 25
lovely@lovely-Lenovo-G580:~/divya$ 
```

11. **Implement RSA algorithm to process blocks of plaintext (refer Figure 9.7 of the text book), where plaintext is a string of characters and let the block size be two characters. (Note : assign a unique code to each plain text character i.e., a=00, A=26). The program should support the following.**
    1. **Accept string of characters as plaintext.**
    2. **Encryption takes plaintext and produces ciphertext characters**
    3. **Decryption takes ciphertext characters obtained in step ii and produces corresponding plaintext characters**
    4. **Display the result after each step.**

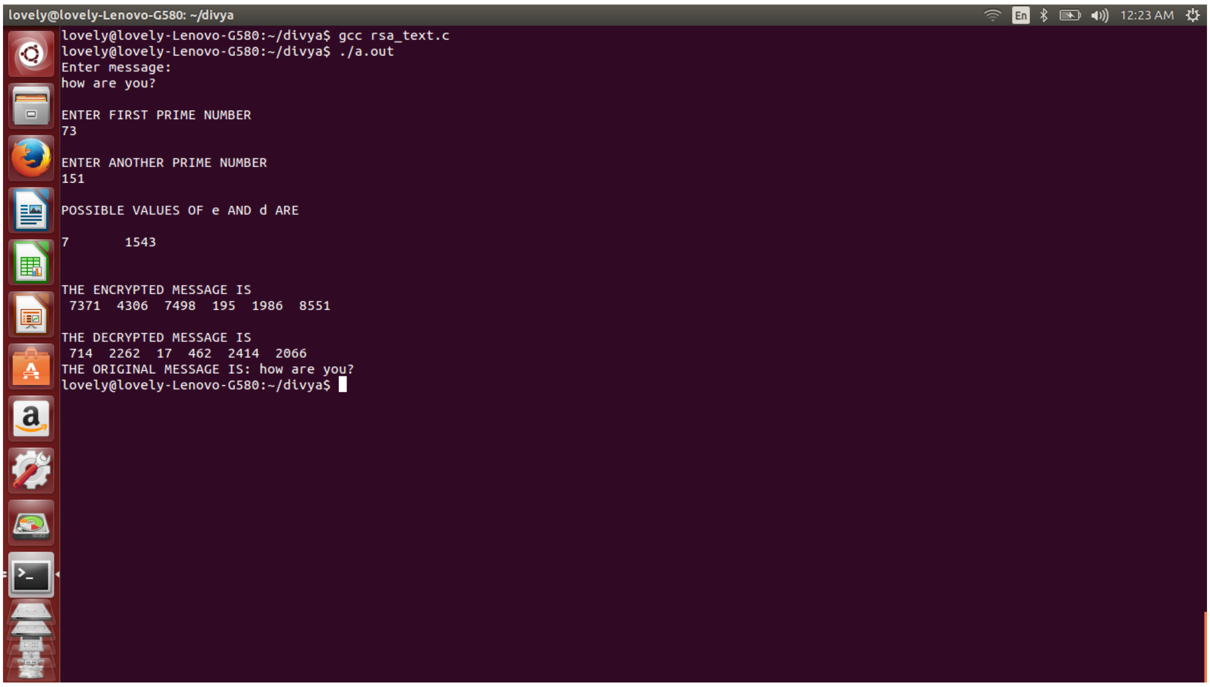
Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<ctype.h>
#include<string.h>
typedef long int int32;
int prime(int32);
void ced(int32[],int32[],int32,int32);
void encrypt(int32, char[],int32,int32[]);
void decrypt(int32,int32[],int32);
int32 decryptencrypt(int32 key,int32 msg,int32 n);
int32 gcd(int32 a,int32 b);
int lower[26]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25};
int
upper[26]={26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51
};
int num[10]={52,53,54,55,56,57,58,59,60,61};
int main()
{
   int32 p,q,n,flag,e,d,en[100];
   char msg[100];
   int i=0;
   printf("Enter message:\n");
   while((msg[i++]=getchar()) != '\n');
   msg[i-1]='\0';
   //puts(msg);
   printf("\nENTER FIRST PRIME NUMBER\n");
   scanf("%ld",&p);
   flag=prime(p);
   if(flag==0)
   {
      printf("\nWRONG INPUT\n");
      exit(1);
   }
   printf("\nENTER ANOTHER PRIME NUMBER\n");
```

```
    scanf("%ld",&q);
    flag=prime(q);
    if(flag==0||p==q)
    {
        printf("\nWRONG INPUT\n");
        exit(1);
    }
    n=p*q;
    ced(&e,&d,p,q);
    printf("\nPOSSIBLE VALUES OF e AND d ARE\n");
    printf("\n%ld\t%ld",e,d);
    printf("\n");
    encrypt(e,msg,n,en);
    decrypt(d,en,n);
    return 0;
}
int prime(int32 pr)
{
    int i;
    for(i=2;i<=pr/2;i++)
    {
        if(pr%i==0)
        return 0;
    }
    return 1;
}
void ced(int32 *e,int32 *d,int32 p,int32 q)
{
    int32 phi,s,t;
    int32 n;
    n = p*q;
    phi=(p-1)*(q-1);
    *e=1;
    do
    {
            (*e)++;
            t=gcd(phi,*e);
    }while(t!=1&& (*e)<phi);
    *d = 0;
    do
    {
        (*d)++;
        s = ((*d)*(*e))%phi;
    }while(s!=1);
}
int32 gcd(int32 a,int32 b)
{
            int32 n=1;
            while(n)
            {
                n= a%b;
                a=b;
                b=n;
        }
        return a;
}
int32 decryptencrypt(int32 key,int32 msg,int32 n)
{
        int32 k;
        int32 j;
   k=1;
   for(j=0;j<key;j++)
   {
                k=k*msg;
                k=k%n;
   }
   return k;
}
void encrypt(int32 key,char msg[100],int32 n,int32 en[100])
{
   int32 k,len;
   int32 temp[100];
   int i=0,j=0;
   int ch;
   int m[100];
   len=strlen(msg);
   if(len%2 != 0)
   {
      printf("\nCharacter used for padding: %c\n",'x');
      msg[len++]='x';
   }
   msg[len]='\0';
   for(i=0;i<len;i++)
   {
      if(isupper(msg[i]))
      ch=upper[msg[i]-65];
      else if(islower(msg[i]))
      ch=lower[msg[i]-97];
      else if(msg[i]>=48 && msg[i]<=57)
      ch=num[msg[i]-48];
      else if(msg[i] == ' ')
      ch=62;
      else if(msg[i] == '.')
      ch=63;
      else if(msg[i] == ',')
      ch=64;
      else if(msg[i] == ';')
      ch=65;
      else if(msg[i] == '?')
      ch=66;
```

```
        else
        {
            printf("Invalid character in message\n");
            exit(1);
        }
        m[i]=ch;
    }
    for(i=0,j=0;i<len;i+=2,j++)
    {
        temp[j]=(m[i]*100+m[i+1]);
        k=decryptencrypt(key,temp[j],n);
        en[j]=k;
    }
    en[j]=-1;
    printf("\n\nTHE ENCRYPTED MESSAGE IS\n");
    for(i=0;en[i]!=-1;i++)
    printf(" %ld ",en[i]);
}
void decrypt(int32 key,int32 en[100],int32 n)
{
    int32 k;
    int32 de[100];
    int m[100],len=0;
    int j;
    char msg[100];
    int ch1,ch2;
    int i;
    i=0;
    while(en[i]!=-1)
    {
        k=decryptencrypt(key,en[i],n);
        de[i]=k;
        i++;
    }
    de[i]=-1;
    printf("\n\nTHE DECRYPTED MESSAGE IS\n");
    for(i=0;de[i]!=-1;i++)
    printf(" %ld ",de[i]);
    i=0;
    while(de[i] != -1)
    {
        ch2=de[i]%100;
        ch1=de[i]/100;
        m[len++]=ch1;
        m[len++]=ch2;
        i++;
    }
    for(i=0;i<len;i++)
    {
        if(m[i]>=0 && m[i]<=25)
        msg[i]=m[i]+'a';
        else if(m[i]>=26 && m[i]<=51)
        msg[i]=m[i]-26+'A';
        else if(m[i]>=52 && m[i]<=61)
        msg[i]=m[i]-52+'0';
        else if(m[i]==62)
        msg[i]=' ';
        else if(m[i]==63)
        msg[i]='.';
        else if(m[i]==64)
        msg[i]=',';
        else if(m[i]==65)
        msg[i]=';';
        else if(m[i]==66)
        msg[i]='?';
    }
    msg[i]='\0';
    printf("\nTHE ORIGINAL MESSAGE IS: ");
    for(j=0;j<i;j++)
    if(msg[j]!='x')
    printf("%c",msg[j]);
    printf("\n");
}
```

Output:



```
lovely@lovely-Lenovo-G580:~/divya$ gcc rsa_text.c
lovely@lovely-Lenovo-G580:~/divya$ ./a.out
Enter message:
how are you?

ENTER FIRST PRIME NUMBER
73

ENTER ANOTHER PRIME NUMBER
151

POSSIBLE VALUES OF e AND d ARE

7       1543


THE ENCRYPTED MESSAGE IS
 7371  4306  7498  195  1986  8551

THE DECRYPTED MESSAGE IS
 714  2262  17  462  2414  2066
THE ORIGINAL MESSAGE IS: how are you?
lovely@lovely-Lenovo-G580:~/divya$ 
```

12. **Implement RSA algorithm using client-server concept. Using this illustrate secret key distribution scenario with confidentiality and authentication. The program should support the following :**

i. **Both client and server generates{PU, PR} and distributes PU to each other.**

ii. **Establish a secret key K between client and server by exchanging the messages as shown in below figure.**

Program:

usera.c

```
#include <stdio.h>
#include <stdlib.h>
#include<netinet/in.h>
typedef long int int32;
int32 gcd(int32 a,int32 b)
{
        int32 n=1;
        while(n)
        {
                n= a%b;
                a=b;
                b=n;
        }
        return a;
}
void keygen(int32 *d,int32 *e,int32 *n)
{
   int32 phi,s;
   int32 t,p,q;
   printf("\nEnter two prime numbers: ");
   scanf("%ld%ld",&p,&q);
   *n = p*q;
   phi=(p-1)*(q-1);
   *e=1;
   do
   {
        (*e)++;
        t=gcd(phi,*e);
        printf("%ld ",t);
   }while(t!=1&& (*e)<phi);
   *d = 0;
   do
   {
      (*d)++;
      s = ((*d)*(*e))%phi;
   }while(s!=1);

   printf("\nPublic key: { e=%ld n=%ld }",*e,*n);
   printf("\nPrivate key: { d=%ld n=%ld }\n",*d,*n);
}
int32 decryptencrypt(int32 key,int32 msg,int32 n)
{
        int32 k;
        int32 i=0,j;
    k=1;
    for(j=0;j<key;j++)
    {
                k=k*msg;
                k=k%n;
    }
    return k;
}
int main()
{
    int32 d,e,n;
        int sockfd,k;
        int32 idA,N1,N2;
        int32 eN2,ek;
        int32 snd[2];
        int32 pu_client[2],pu_server[2];
        struct sockaddr_in server;
        int port;
        char host[15];
        sockfd = socket(AF_INET,SOCK_STREAM,0);
        printf("Enter the port number: ");
        scanf("%d",&port);
        printf("Enter the address of B : ");
        scanf("%s",host);
        server.sin_family = AF_INET;
        server.sin_port = htons(port);
        server.sin_addr.s_addr = inet_addr(host);
        connect(sockfd,(struct sockaddr*)&server,sizeof(server));
        keygen(&d,&e,&n);
        pu_client[0] = e;
        pu_client[1] = n;
        printf("\nPublic key of A: {%ld,%ld}\n",e,n);
        send(sockfd,&pu_client,sizeof(pu_client),0);
        printf("Public key of A sent to B...\n");
        recv(sockfd,&pu_server,sizeof(pu_server),0);
        printf("Public key of B is : {%ld,%ld}\n ",pu_server[0],pu_server[1]);
        printf("Enter the Identity of A:\n");
        scanf("%ld",&idA);
        printf("Enter nonce value N1:\n");
        scanf("%ld",&N1);
        snd[0]=decryptencrypt(pu_server[0],idA,pu_server[1]);
        snd[1]=decryptencrypt(pu_server[0],N1,pu_server[1]);
        printf("Sending encrypted idA and nonce N1 to B...\n");
        send(sockfd,&snd,sizeof(snd),0);
        recv(sockfd,&snd,sizeof(snd),0);
        N2=decryptencrypt(d,snd[1],n);
        eN2=decryptencrypt(pu_server[0],N2,pu_server[1]);
        printf("Sending encrypted nonce N2 to B...\n");
        send(sockfd,&eN2,sizeof(eN2),0);
        printf("Enter the secret key : ");
        scanf("%d",&k);
        ek=decryptencrypt(d,k,n);
        ek=decryptencrypt(pu_server[0],ek,pu_server[1]);
        printf("Sending encrypted secret key to B...\n");
        send(sockfd,&ek,sizeof(ek),0);
        close(sockfd);
   return 0;
}
```

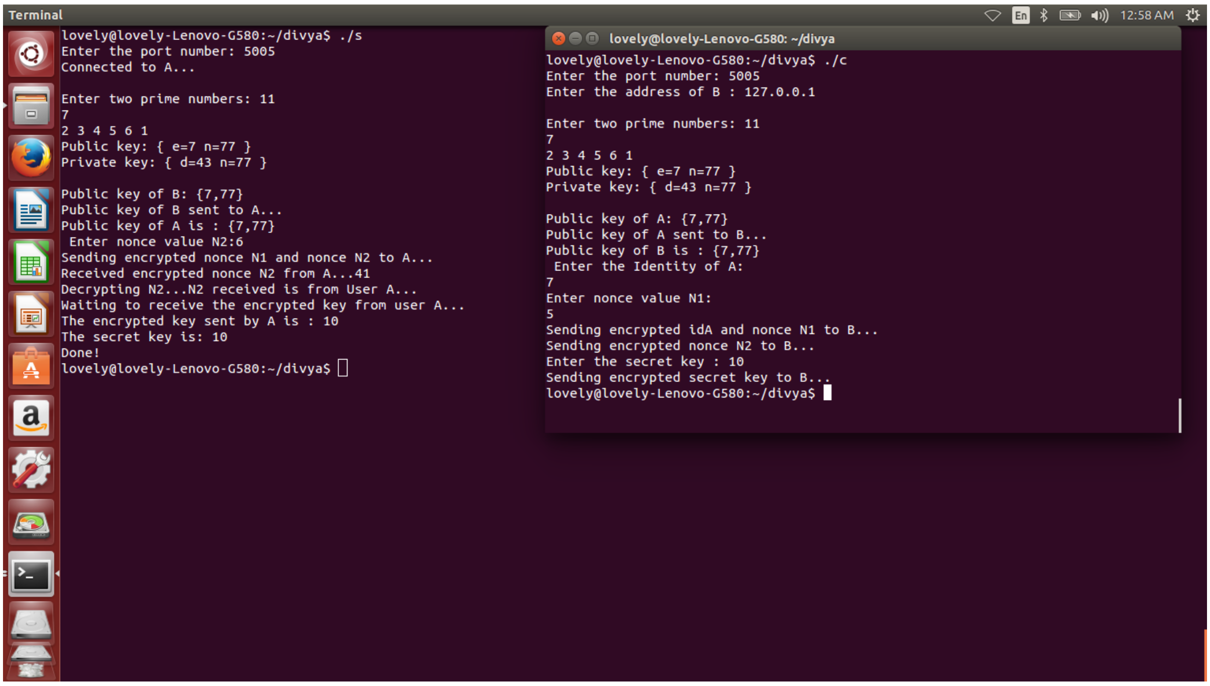
userb.c

```
#include <stdio.h>
#include <stdlib.h>
#include<netinet/in.h>
typedef long int int32;
int32 gcd(int32 a,int32 b)
{
        int32 n=1;
        while(n)
        {
                n= a%b;
                a=b;
                b=n;
        }
        return a;
}
void keygen(int32 *d,int32 *e,int32 *n)
{
   int32 phi,s;
   int32 t,p,q;
   printf("\nEnter two prime numbers: ");
   scanf("%ld%ld",&p,&q);
   *n = p*q;
   phi=(p-1)*(q-1);
   *e=1;
   do
   {
        (*e)++;
        t=gcd(phi,*e);
        printf("%ld ",t);
   }while(t!=1&& (*e)<phi);
   *d = 0;
   do
   {
      (*d)++;
      s = ((*d)*(*e))%phi;
   }while(s!=1);
   printf("\nPublic key: { e=%ld n=%ld }",*e,*n);
   printf("\nPrivate key: { d=%ld n=%ld }\n",*d,*n);
```

```
}
int32 decryptencrypt(int32 key,int32 msg,int32 n)
{
        int32 k;
        int32 i=0,j;
   k=1;
   for(j=0;j<key;j++)
   {
                k=k*msg;
                k=k%n;
   }
   return k;
}
int main()
{
   int32 d,e,n,k;
        int sockfd,clen,nsfd;
        int32 snd[2];
        int32 N1,N2,eN1,eN2,ek,N2_new;
        int32 pu_client[2],pu_server[2];
        struct sockaddr_in server;
        int port;
        printf("Enter the port number: ");
        scanf("%d",&port);
        sockfd = socket(AF_INET,SOCK_STREAM,0);
        server.sin_family = AF_INET;
        server.sin_port = htons(port);
        server.sin_addr.s_addr = INADDR_ANY;
        bind(sockfd, (struct sockaddr*)&server,sizeof(server));
        listen(sockfd,1);
        nsfd = accept(sockfd,NULL,NULL);
        printf("Connected to A...\n");
        close(sockfd);
        keygen(&d,&e,&n);
        pu_server[0] = e;
                // PU{e,n}
        pu_server[1] = n;
        printf("\nPublic key of B: {%ld,%ld}\n",e,n);
        send(nsfd,&pu_server,sizeof(pu_server),0);
        printf("Public key of B sent to A...\n");
   recv(nsfd,&pu_client,sizeof(pu_client),0);
   printf("Public key of A is : {%ld,%ld}\n ",pu_client[0],pu_client[1]);
        recv(nsfd,&snd,sizeof(snd),0);
        N1=decryptencrypt(d,snd[0],n);
        printf("Enter nonce value N2:");
        scanf("%ld",&N2);
        snd[0]=decryptencrypt(pu_client[0],N1,pu_client[1]);
        snd[1]=decryptencrypt(pu_client[0],N2,pu_client[1]);
        printf("Sending encrypted nonce N1 and nonce N2 to A...\n");
        send(nsfd,&snd,sizeof(snd),0);
```

```
          recv(nsfd,&eN2,sizeof(eN2),0);
          printf("Received encrypted nonce N2 from A...%ld\nDecrypting N2...",eN2);
          N2_new=decryptencrypt(d,eN2,n);
          if(N2==N2_new)
          printf("N2 received is from User A...\n");
          printf("Waiting to receive the encrypted key from user A...\n");
          recv(nsfd,&ek,sizeof(ek),0);
          printf("The encrypted key sent by A is : %ld\n",ek);
          k=decryptencrypt(d,ek,n);
          k=decryptencrypt(pu_client[0],k,pu_client[1]);
          printf("The secret key is: %ld",k);
     printf("\nDone!\n");
          close(nsfd);
     return 0;
}
```

Output:



```
lovely@lovely-Lenovo-G580:~/divya$ ./s
Enter the port number: 5005
Connected to A...

Enter two prime numbers: 11
7
2 3 4 5 6 1
Public key: { e=7 n=77 }
Private key: { d=43 n=77 }

Public key of B: {7,77}
Public key of B sent to A...
Public key of A is : {7,77}
 Enter nonce value N2:6
Sending encrypted nonce N1 and nonce N2 to A...
Received encrypted nonce N2 from A...41
Decrypting N2...N2 received is from User A...
Waiting to receive the encrypted key from user A...
The encrypted key sent by A is : 10
The secret key is: 10
Done!
lovely@lovely-Lenovo-G580:~/divya$
```

```
lovely@lovely-Lenovo-G580:~/divya$ ./c
Enter the port number: 5005
Enter the address of B : 127.0.0.1

Enter two prime numbers: 11
7
2 3 4 5 6 1
Public key: { e=7 n=77 }
Private key: { d=43 n=77 }

Public key of A: {7,77}
Public key of A sent to B...
Public key of B is : {7,77}
 Enter the Identity of A:
7
Enter nonce value N1:
5
Sending encrypted idA and nonce N1 to B...
Sending encrypted nonce N2 to B...
Enter the secret key : 10
Sending encrypted secret key to B...
lovely@lovely-Lenovo-G580:~/divya$
```

13. Compute common secret key between client and server using Diffie-Hellman key exchange technique. Perform encryption and decryption of message using the shared secret key (Use simple XOR operation to encrypt and decrypt the message.)

Program:

dfc.c

```
#include<stdio.h>
#include<netinet/in.h>
#include<time.h>
```

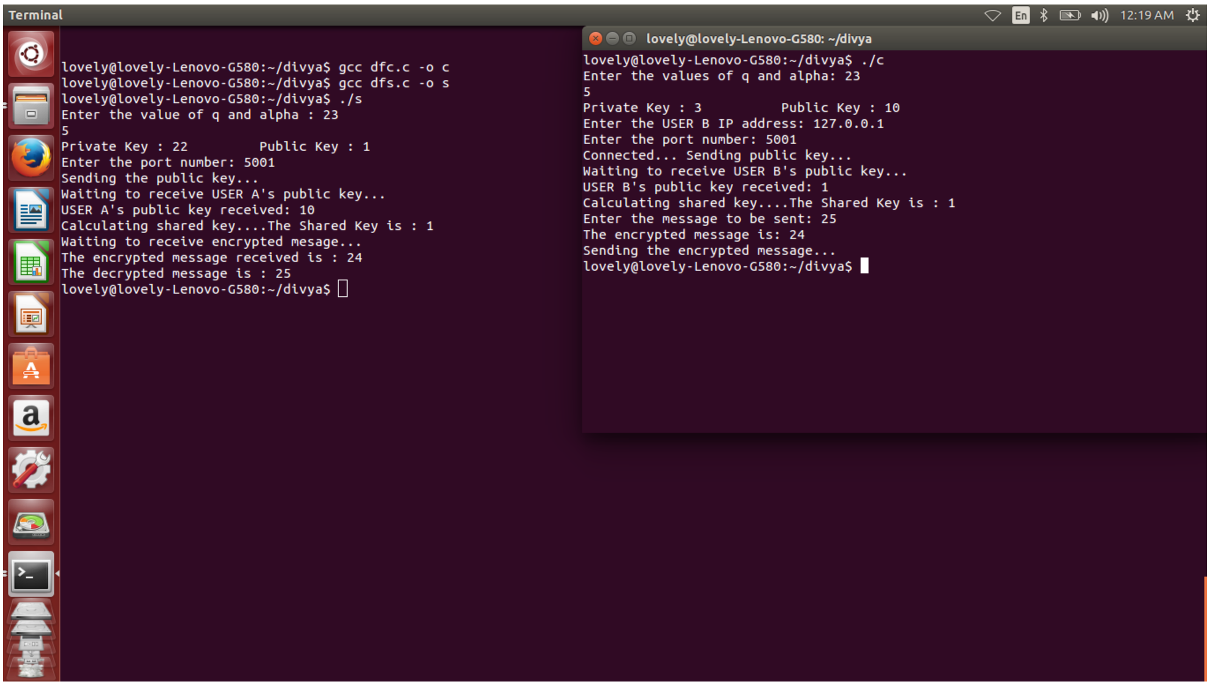
```
int generate(int xa,int q,int num)
{
        int i,ret=1;
        for(i=0;i<xa;i++)
        {
                ret=ret*num;
                ret=ret%q;
        }
        return ret;
}
void main()
{
        int q,alpha;
        int xa,ya=1,yb,k=1,i,port,sockfd,len,p,c;
        char ser[20];
        struct sockaddr_in server;
        srand(time(NULL));
        printf("Enter the values of q and alpha: ");
        scanf("%d%d",&q,&alpha);
        xa = rand()%q;
        ya = generate(xa,q,alpha);
        printf("Private Key : %d \t Public Key : %d\n",xa,ya);
        printf("Enter the USER B IP address: ");
        scanf("%s",ser);
        printf("Enter the port number: ");
        scanf("%d",&port);
        sockfd = socket(AF_INET,SOCK_STREAM,0);
        server.sin_port = htons(port);
        server.sin_addr.s_addr = inet_addr(ser);
        server.sin_family = AF_INET;
        connect(sockfd,(struct sockaddr*)&server,sizeof(server));
        printf("Connected... Sending public key...\n");
        send(sockfd,&ya,sizeof(ya),0);
        printf("Waiting to receive USER B's public key...\n");
        recv(sockfd,&yb,sizeof(yb),0);
        printf("USER B's public key received: %d\nCalculating shared key....",yb);
        k=generate(xa,q,yb);
        printf("The Shared Key is : %d \n",k);
        printf("Enter the message to be sent: ");
        scanf("%d",&p);
        c=p^k;
        printf("The encrypted message is: %d\nSending the encrypted message...\n",c);
        send(sockfd,&c,sizeof(c),0);
        close(sockfd);
}
```

dfs.c

```
#include<stdio.h>
#include<netinet/in.h>
```

```
#include<string.h>
#include<time.h>
int generate(int x,int q,int num)
{
        int i,ret=1;
        for(i=0;i<x;i++)
        {
                ret=ret*num;
                ret=ret%q;
        }
        return ret;
}
void main()
{
        int q,alpha;
        int xb,yb=1,ya,k=1,i,port,sockfd,len,p,c,newsockfd;
        struct sockaddr_in server;
        srand(time(NULL));
        printf("Enter the value of q and alpha : ");
        scanf("%d%d",&q,&alpha);
        xb = rand()%q;
        yb = generate(xb,q,alpha);
        printf("Private Key : %d \t Public Key : %d\n",xb,yb);
        printf("Enter the port number: ");
        scanf("%d",&port);
        sockfd = socket(AF_INET,SOCK_STREAM,0);
        server.sin_family= AF_INET;
        server.sin_port = htons(port);
        server.sin_addr.s_addr = INADDR_ANY;
        bind(sockfd,(struct sockaddr*)&server,sizeof(server));
        listen(sockfd,1);
        newsockfd = accept(sockfd,NULL,NULL);
        close(sockfd);
        printf("Sending the public key...\n");
        send(newsockfd,&yb,sizeof(yb),0);
        printf("Waiting to receive USER A's public key...\n");
        recv(newsockfd,&ya,sizeof(ya),0);
        printf("USER A's public key received: %d\nCalculating shared key....",ya);
        k = generate(xb,q,ya);
        printf("The Shared Key is : %d \n",k);
        printf("Waiting to receive encrypted mesage...\n");
        recv(newsockfd,&c,sizeof(c),0);
        printf("The encrypted message received is : %d\n", c);
        p=c^k;
        printf("The decrypted message is : %d\n",p);
        close(newsockfd);
}
```

Output:



```
lovely@lovely-Lenovo-G580:~/divya$ gcc dfc.c -o c
lovely@lovely-Lenovo-G580:~/divya$ gcc dfs.c -o s
lovely@lovely-Lenovo-G580:~/divya$ ./s
Enter the value of q and alpha : 23
5
Private Key : 22         Public Key : 1
Enter the port number: 5001
Sending the public key...
Waiting to receive USER A's public key...
USER A's public key received: 10
Calculating shared key....The Shared Key is : 1
Waiting to receive encrypted mesage...
The encrypted message received is : 24
The decrypted message is : 25
lovely@lovely-Lenovo-G580:~/divya$
```

```
lovely@lovely-Lenovo-G580:~/divya$ ./c
Enter the values of q and alpha: 23
5
Private Key : 3          Public Key : 10
Enter the USER B IP address: 127.0.0.1
Enter the port number: 5001
Connected... Sending public key...
Waiting to receive USER B's public key...
USER B's public key received: 1
Calculating shared key....The Shared Key is : 1
Enter the message to be sent: 25
The encrypted message is: 24
Sending the encrypted message...
lovely@lovely-Lenovo-G580:~/divya$
```

14. Implement DSS algorithm for signing and verification of messages between two parties (obtain H(M) using simple XOR method of hash computation on M).

Program:

/* DSS Signing */

```
#include<stdio.h>
#include<netinet/in.h>
#include<string.h>
#include<time.h>
#include<stdlib.h>
int powermod(int num,int power,int mod)
{
        int i,ret=1;
        for(i=0;i<power;i++)
        {
                ret=ret*num;
                ret=ret%mod;
        }
        return ret;
}
int inverse(int k,int mod)
{
        int i;
        for(i=0;i<mod;i++)
                if((k*i)%mod==1)
                        return i;
        return -1;
}
void main()
{
	int p,q,h,x,k,sockfd,port,snd[5],hash;
	int r,s,g,y;
	int i=0;
	struct sockaddr_in server;
	char host[20];
	sockfd=socket(AF_INET,SOCK_STREAM,0);
	printf("Enter the port number: ");
	scanf("%d",&port);
	printf("Enter the receiver's IP address: ");
	scanf("%s",host);
	server.sin_family = AF_INET;
	server.sin_port = htons(port);
	server.sin_addr.s_addr = inet_addr(host);
	connect(sockfd,(struct sockaddr*)&server,sizeof(server));
	srand(time(NULL));
	printf("Enter the values of p & q :");
	scanf("%d%d",&p,&q);
	h=rand()%(p-1);
	if(h==1) h++;
	g = powermod(h,(p-1)/q,p);
	do{
		x = rand()%q;
		k = rand()%q;
	}while(x==0||k==0);
	printf("Enter the hash value: ");
	scanf("%d",&hash);
	y = powermod(g,x,p);
	r = powermod(g,k,p)%q;
	s = ( inverse(k,q) * (hash+x*r) )%q;
	snd[0]=hash;
	snd[1]=s;
	snd[2]=r;
	snd[3]=y;
	snd[4]=g;
	printf("\np: %d\nq: %d\ny: %d\nhash: %d\ns: %d\nr: %d\ng: %d\n",p,q,y,hash,s,r,g);
	write(sockfd,snd,sizeof(snd));
	printf("Done!");
	close(sockfd);
}
```

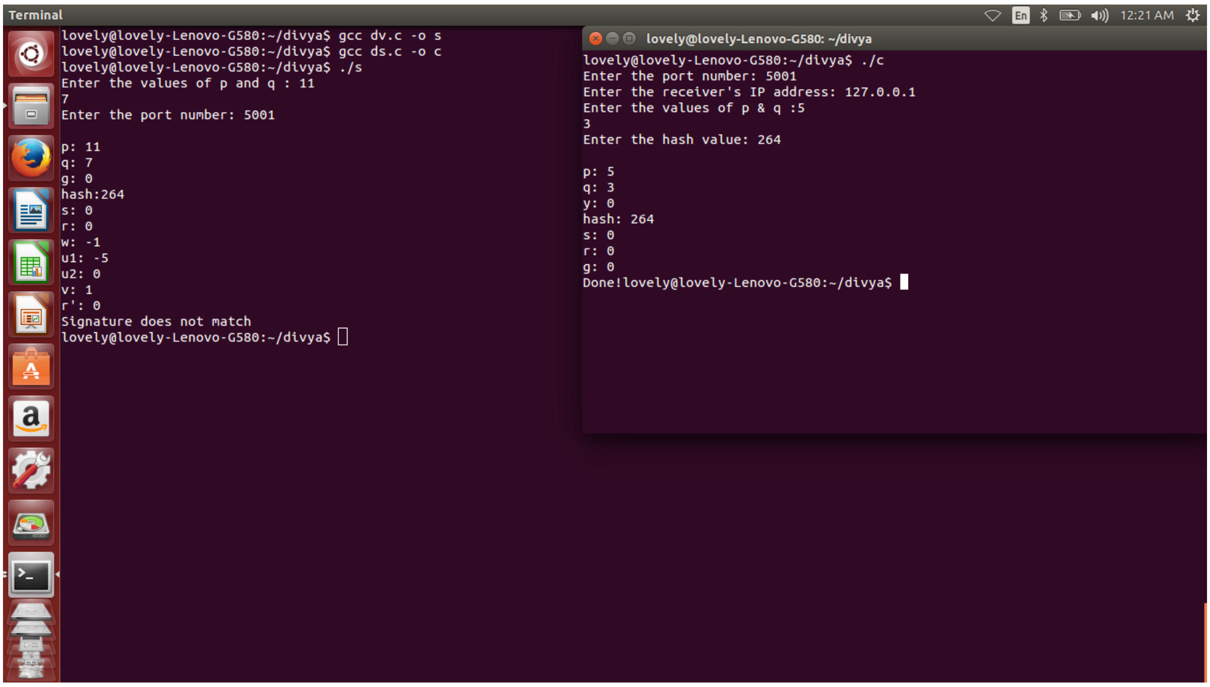
/* DSS Verifying */

```
#include<stdio.h>
#include<netinet/in.h>
#include<string.h>
#include<time.h>
#include<stdlib.h>
int powermod(int num,int power,int mod)
```

```
{
        int i,ret=1;
        for(i=0;i<power;i++)
        {
                ret=ret*num;
                ret=ret%mod;
        }
        return ret;
}
int inverse(int k,int mod)
{
        int i;
        for(i=0;i<mod;i++)
                if((k*i)%mod==1)
                        return i;
        return -1;
}
void verify(int p,int q,int g,int s1, int r1,int y, int hash)
{
   int w,u1,u2,v;
   w=inverse(s1,q);
   printf("w: %d\n",w);
   u1=(hash*w)%q;
   printf("u1: %d\n",u1);
   u2=(r1*w)%q;
   printf("u2: %d\n",u2);
   v=((powermod(g,u1,p)*powermod(y,u2,p))%p)%q;
   printf("v: %d\nr': %d\n",v,r1);
   if(v==r1)
        printf("Signature verified\n");
   else
        printf("Signature does not match\n");
}
void main()
{
        struct sockaddr_in server;
        int port,sockfd,newsockfd,rcv[5],p,q;
        printf("Enter the values of p and q : ");
        scanf("%d%d",&p,&q);
        sockfd=socket(AF_INET,SOCK_STREAM,0);
        printf("Enter the port number: ");
        scanf("%d",&port);
        server.sin_family = AF_INET;
        server.sin_port = htons(port);
        server.sin_addr.s_addr = INADDR_ANY;
        bind(sockfd,(struct sockaddr*)&server,sizeof(server));
        listen(sockfd,1);
        newsockfd = accept(sockfd,NULL,NULL);
        close(sockfd);
        read(newsockfd,rcv,sizeof(rcv));
```

```
	printf("\np: %d\nq: %d\ng: %d\nhash:%d\ns: %d\nr:
%d\n",p,q,rcv[4],rcv[0],rcv[1],rcv[2]);
	verify(p,q,rcv[4],rcv[1],rcv[2],rcv[3],rcv[0]);
	close(newsockfd);
}
```

Output:



```
lovely@lovely-Lenovo-G580:~/divya$ gcc dv.c -o s
lovely@lovely-Lenovo-G580:~/divya$ gcc ds.c -o c
lovely@lovely-Lenovo-G580:~/divya$ ./s
Enter the values of p and q : 11
7
Enter the port number: 5001

p: 11
q: 7
g: 0
hash:264
s: 0
r: 0
w: -1
u1: -5
u2: 0
v: 1
r': 0
Signature does not match
lovely@lovely-Lenovo-G580:~/divya$ 
```

```
lovely@lovely-Lenovo-G580:~/divya$ ./c
Enter the port number: 5001
Enter the receiver's IP address: 127.0.0.1
Enter the values of p & q :5
3
Enter the hash value: 264

p: 5
q: 3
y: 0
hash: 264
s: 0
r: 0
g: 0
Done!lovely@lovely-Lenovo-G580:~/divya$ 
```