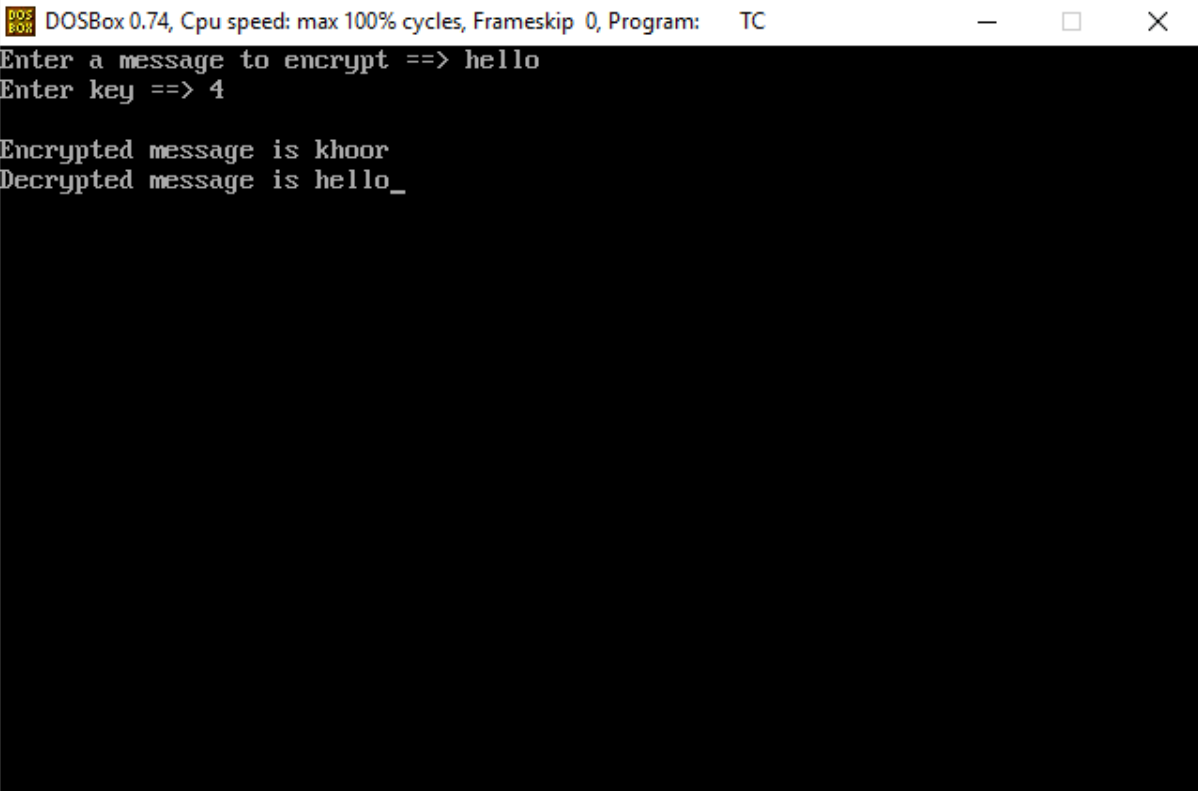# Practical – 1

Aim: Write a program to implement Caesar cipher Encryption - Decryption.

```c
#include<stdio.h>
#include<conio.h>
void main()
{
        char message[100], ch;
        int i, key;
        clrscr();
        printf("Enter a message to encrypt ==> ");
        gets(message);
        printf("Enter key ==> ");
        scanf("%d",&key);
        for(i = 0; message[i] != '\0'; ++i)
        {
                ch = message[i];
                if(ch >= 'a' && ch <= 'z')
                {
                        ch = ch + key;
                        if(ch > 'z')
                        {
                                ch = ch - 'z' + 'a' - 1;
                        }
                        message[i] = ch;
                }
                else if(ch >= 'A' && ch <= 'Z')
                {
                        ch = ch + key;
                        if(ch > 'Z')
                        {
                                ch = ch - 'Z' + 'A' - 1;
                        }
                        message[i] = ch;
```

```c
            }
    }
    printf("\nEncrypted message is %s",message);
    for(i = 0; message[i] != '\0'; i++)
    {
            ch = message[i];
            if(ch >= 'a' && ch <= 'z')
            {
                    ch = ch - key;
                    if(ch < 'a')
                    {
                            ch = ch + 'z' - 'a' + 1;
                    }
                    message[i] = ch;
            }
            else if(ch >= 'A' && ch <= 'Z')
            {
                    ch = ch - key;
                    if(ch < 'A')
                    {
                            ch = ch + 'Z' - 'A' + 1;
                    }
                    message[i] = ch;
            }
    }
    printf("\nDecrypted message is %s",message);
    getch();
}
```

● **Output:**
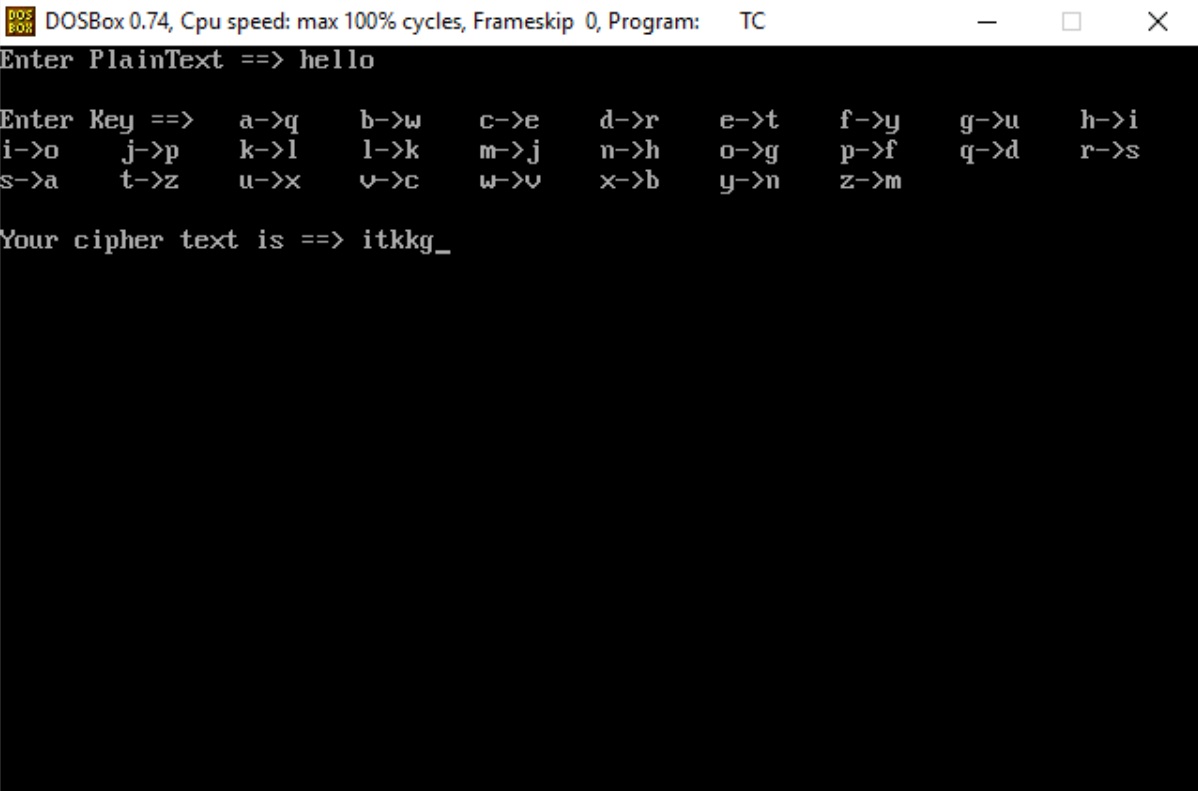
Practical – 2

Aim: Write a program to implement Mono-alphabetic cipher Encryption.

```c
#include<stdio.h>
#include<conio.h>
void main()
{
        char p[30],k[30],c[30];
        int i,index,len;
        clrscr();
        printf("Enter PlainText ==> ");
        gets(p);
        len=strlen(p);
        printf("\nEnter Key ==>");
        for(i=0;i<26;i++)
        {
                printf("\t");
                printf("%c->",i+97);
                k[i]=getch();
                printf("%c",k[i]);
        }
        for(i=0;i<len;i++)
        {
                index=p[i]-97;
                c[i]=k[index];
        }
        c[i]=NULL;
        printf("\n\nYour cipher text is ==> %s",c);
        getch();

}
```
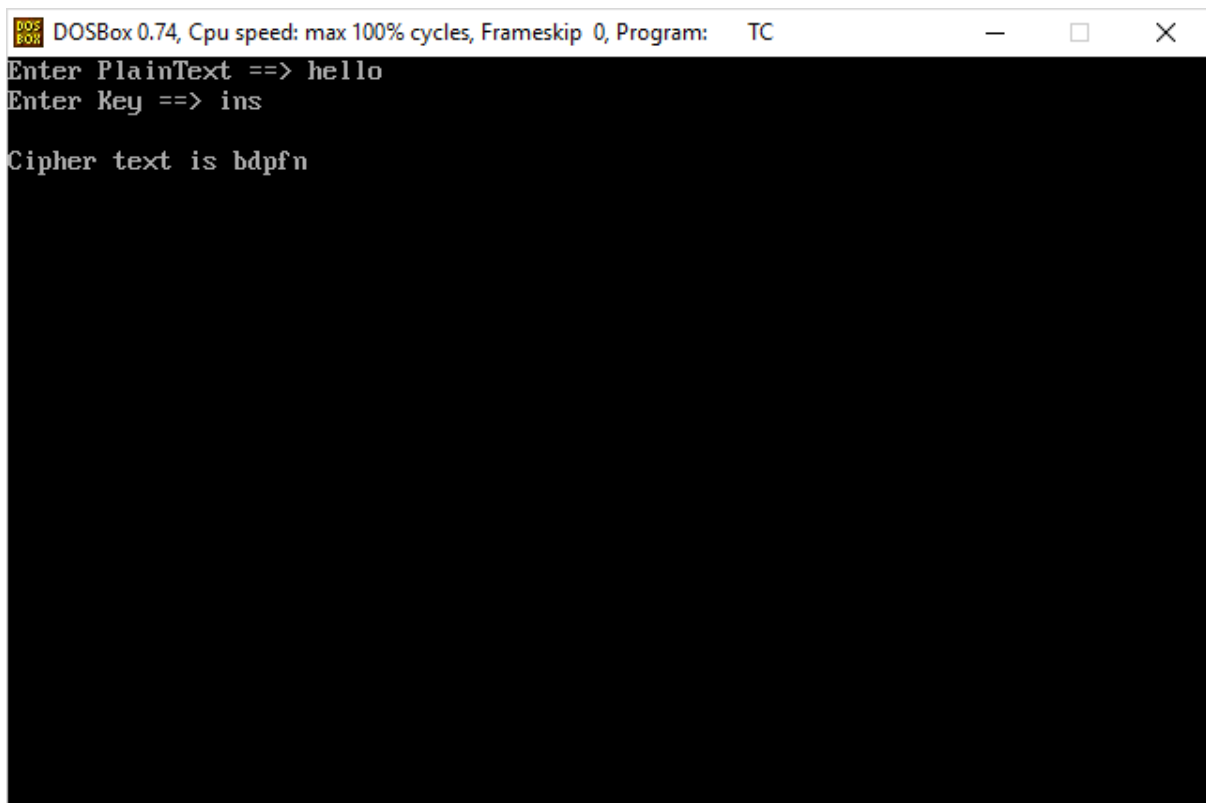
- **Output:**


```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:    TC         —    □    ×
Enter PlainText ==> hello

Enter Key ==>    a->q     b->w     c->e     d->r     e->t     f->y     g->u     h->i
i->o     j->p     k->l     l->k     m->j     n->h     o->g     p->f     q->d     r->s
s->a     t->z     u->x     v->c     w->v     x->b     y->n     z->m

Your cipher text is ==> itkkg_
```

Practical – 3

Aim: Write a program to implement Poly-alphabetic cipher Encryption.

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char p[30],c[30],k[30];
        int i,j=0,len;
        clrscr();
        printf("Enter PlainText ==> ");
        gets(p);
        printf("Enter Key ==> ");
        gets(k);
        len=strlen(k);
        for(i=0;p[i]!=NULL;i++)
        {
                c[i]=((p[i]+k[j])%26)+'a';
                j++;
                if(j==len)
                {
                        j=0;
                }
        }
        c[i]=NULL;
        printf("\nCipher text is %s",c);
        getch();
}
```

- **Output:**

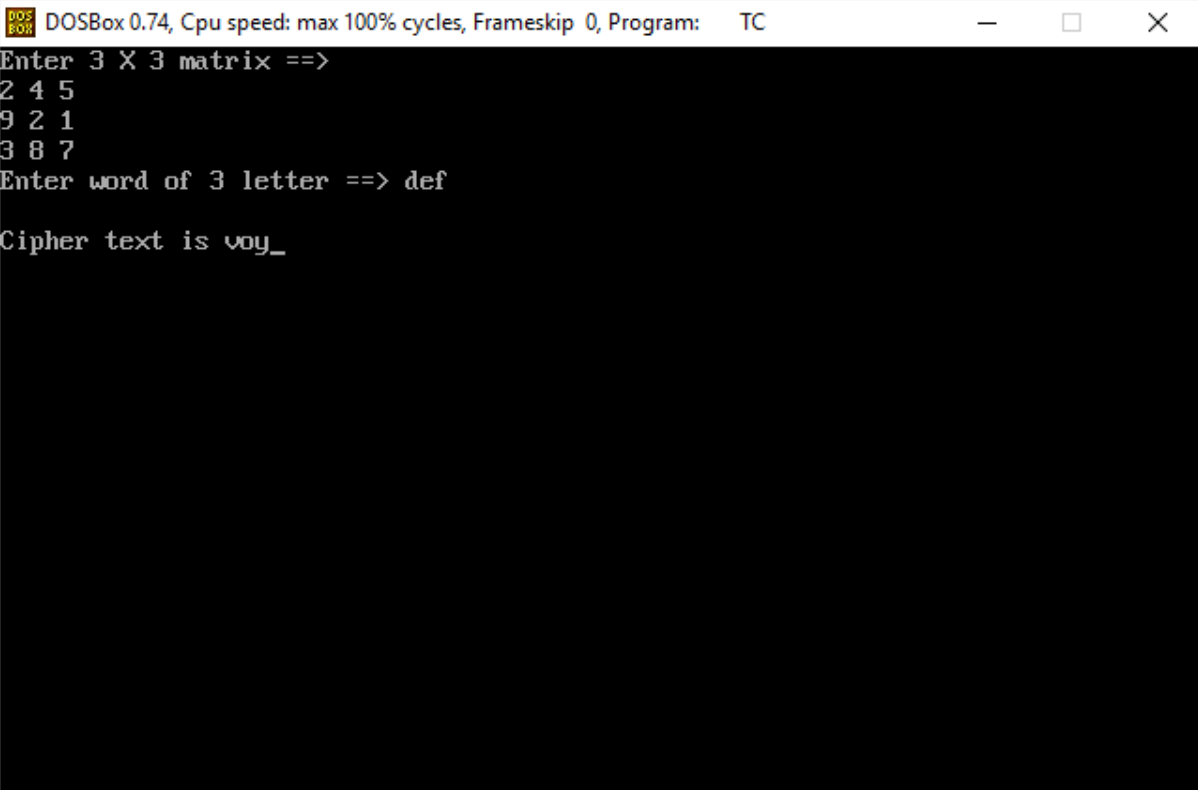# Practical – 4

Aim: Write a program to implement Hill cipher Encryption.

```c
#include<stdio.h>
#include<conio.h>
void main()
{
        int k[3][3],p1[3][1],c1[3][1],i,j,l;
        char p[5];
        clrscr();
        printf("Enter 3 X 3 matrix ==> \n");
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                {
                        scanf("%d",&k[i][j]);
                }
        }
        printf("Enter word of 3 letter ==> ");
        scanf("%s",p);
        for(i=0;i<3;i++)
        {
                p1[i][0]=p[i]-97;
        }
        for(i=0;i<3;i++)
        {
                for(j=0;j<1;j++)
                {
                        c1[i][j]=0;
                        for(l=0;l<3;l++)
                        {
                                c1[i][j]+=k[i][l]*p1[l][j];
                        }
                }
```

```
        }
        printf("\nCipher text is ");
        for(i=0;i<3;i++)
        {
                for(j=0;j<1;j++)
                {
                        printf("%c",(c1[i][j]%26)+97);
                }
        }
        getch();
}
```
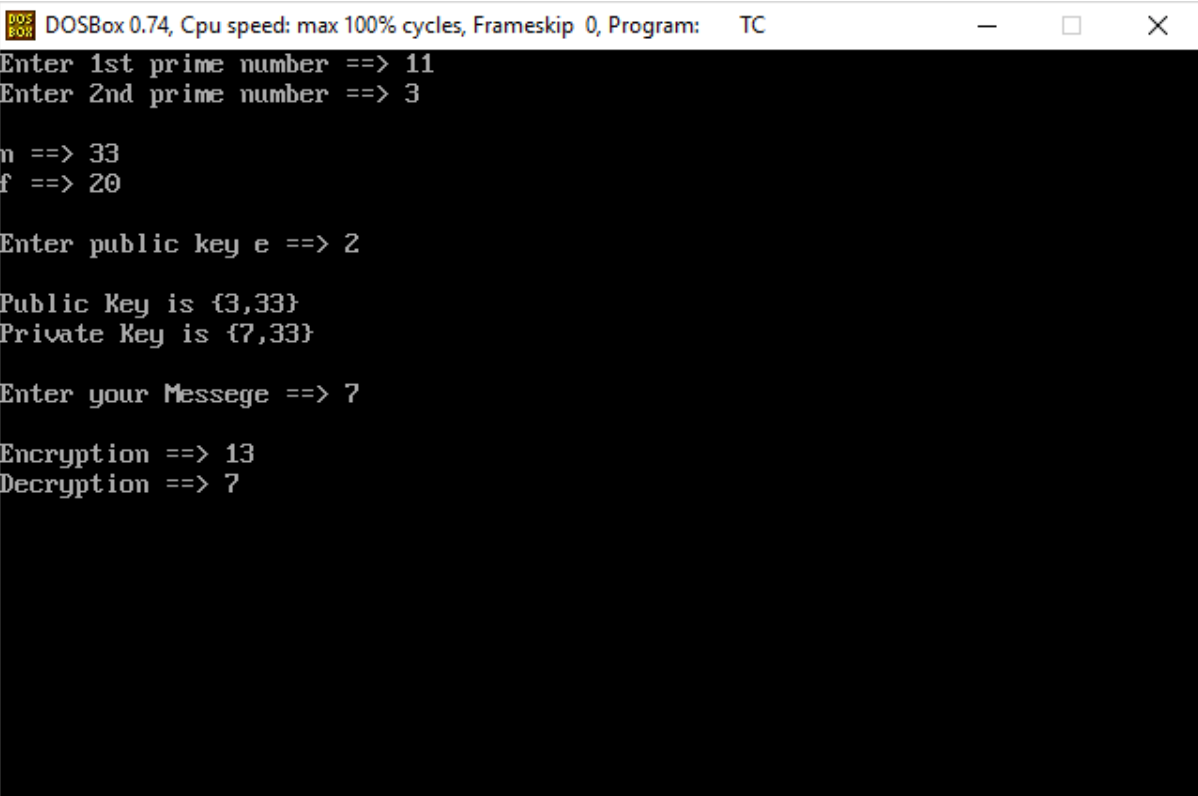
- **Output:**

Practical – 5

Aim: Write a program to implement RSA Encryption – Decryption algorithm.

```c
#include<stdio.h>
#include<conio.h>
#include<math.h>
int gcd(int a,int b)
{
        int i,c;
        for(i=1;i<=a&&i<=b;i++)
        {
                if(a%i==0 && b%i==0)
                {
                        c = i;
                }
        }
        return c;
}
void main()
{
        int p,q,n,f,e,d,s,msg;
        long enc,dec;
        clrscr();
        printf("Enter 1st prime number ==> ");
        scanf("%d",&p);
        printf("Enter 2nd prime number ==> ");
        scanf("%d",&q);
        n = p * q;
        f = (p-1)*(q-1);
        printf("\nn ==> %d",n);
        printf("\nf ==> %d",f);
        printf("\n\nEnter public key e ==> ");
        scanf("%d",&e);
```

```c
        while(e<f)
        {
                if(gcd(e,f)==1)
                {
                        break;
                }
                else
                {
                        e++;
                }
        }
        d=1;
        do
        {
                s = (d*e)%f;
                d++;
        }while(s!=1);
        d=d-1;
        printf("\nPublic Key is {%d,%d}",e,n);
        printf("\nPrivate Key is {%d,%d}",d,n);
        printf("\n\nEnter your Messege ==> ");
        scanf("%d",&msg);
        enc=pow(msg,e);
        enc=fmod(enc,n);
        dec=pow(enc,d);
        dec=fmod(dec,n);
        printf("\nEncryption ==> %ld",enc);
        printf("\nDecryption ==> %ld",dec);
        getch();
    }
```

Output:

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:    TC          —  □  ×
Enter 1st prime number ==> 11
Enter 2nd prime number ==> 3

n ==> 33
f ==> 20

Enter public key e ==> 2

Public Key is {3,33}
Private Key is {7,33}

Enter your Messege ==> 7

Encryption ==> 13
Decryption ==> 7
```
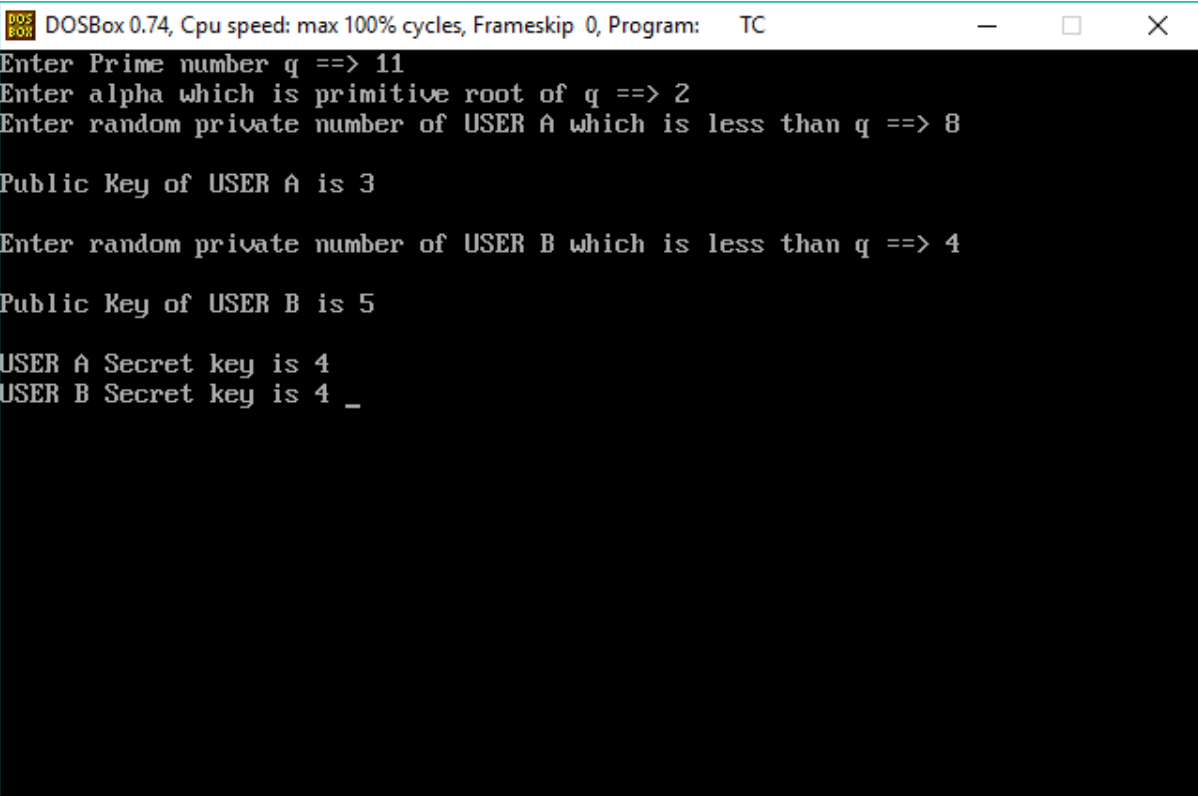
Practical – 6

Aim: Write a program to implement Diffie Hellman key exchange algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
        int q,Xa,Xb,Yb,Kb,alpha,Ya;
        long Ka;
        clrscr();
        printf("Enter Prime number q ==> ");
        scanf("%d",&q);
        printf("Enter alpha which is primitive root of q ==> ");
        scanf("%d",&alpha);
        printf("Enter random private number of USER A which is less than q ==> ");
        scanf("%d",&Xa);
        Ya = pow(alpha,Xa);
        Ya = fmod(Ya,q);
        printf("\nPublic Key of USER A is %d ",Ya);
        printf("\n\nEnter random private number of USER B which is less than q ==> ");
        scanf("%d",&Xb);
        Yb = pow(alpha,Xb);
        Yb = fmod(Yb,q);
        printf("\nPublic Key of USER B is %d ",Yb);
        Ka = pow(Yb,Xa);
        Ka = fmod(Ka,q);
        printf("\n\nUSER A Secret key is %ld ",Ka);
        Kb = pow(Ya,Xb);
        Kb = fmod(Kb,q);
        printf("\nUSER B Secret key is %d ",Kb);
        getch();
}
```
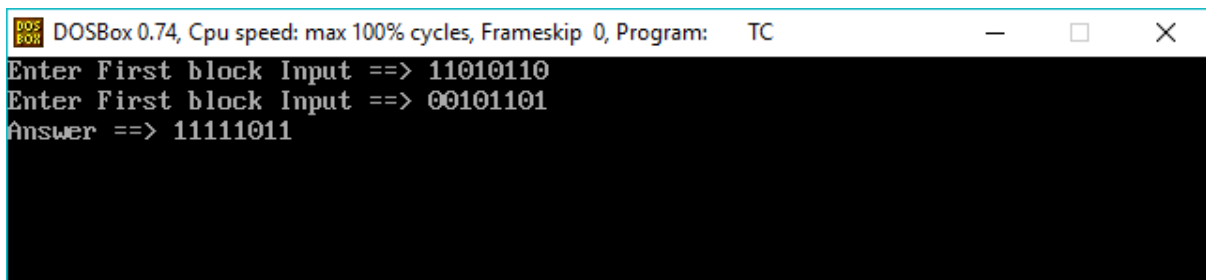
● **Output:**

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC   —   □   ×

```
Enter Prime number q ==> 11
Enter alpha which is primitive root of q ==> 2
Enter random private number of USER A which is less than q ==> 8

Public Key of USER A is 3

Enter random private number of USER B which is less than q ==> 4

Public Key of USER B is 5

USER A Secret key is 4
USER B Secret key is 4 _
```

Practical – 7

Aim: Write a program to implement Simple Hash function.

```c
#include<stdio.h>
#include<conio.h>
void main()
{
        char ip1[20],ip2[20],ans;
        int i,j;
        clrscr();
        printf("Enter First block Input ==> ");
        gets(ip1);
        printf("Enter First block Input ==> ");
        gets(ip2);
        printf("Answer ==>");
        for(i=0;i<8;i++)
        {
                ans=ip1[i]^ip2[i];
                printf("%d",ans);
        }
        getch();
}
```

- **Output:**

Practical – 8
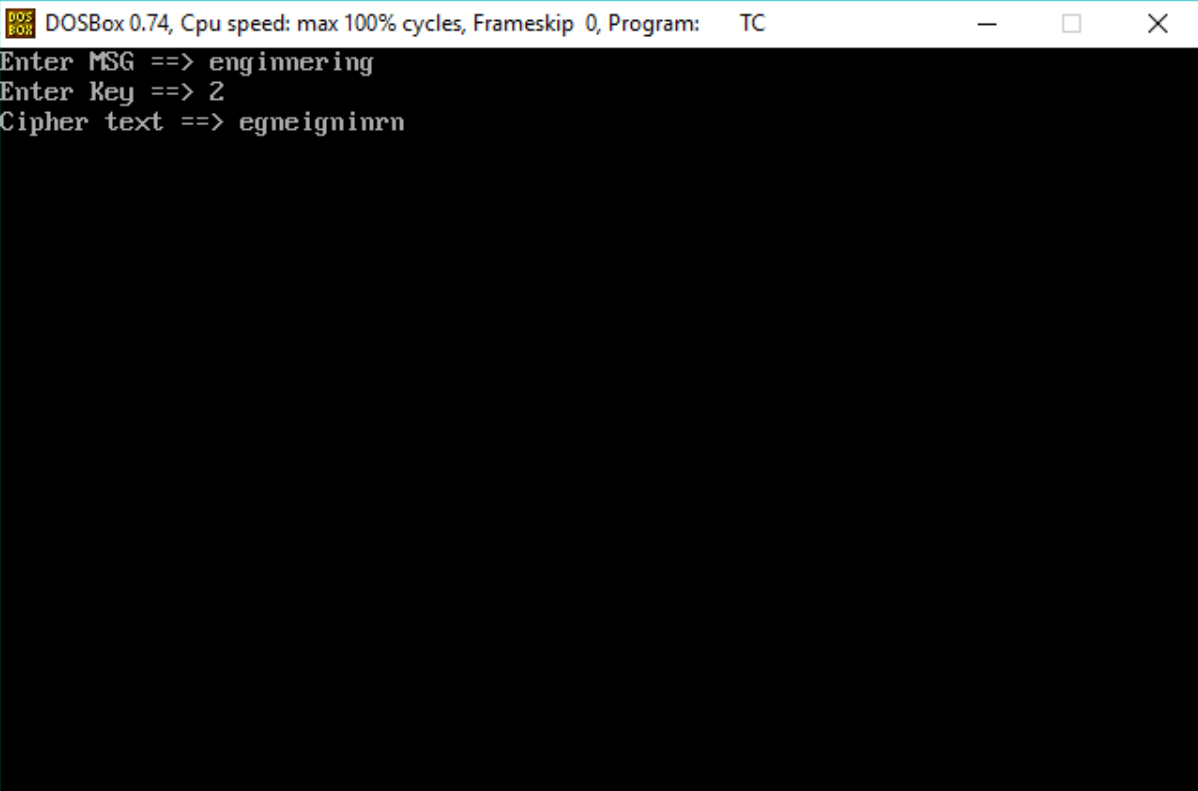
Aim: Write a program to implement Rail Fence cipher encryption.

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char msg[20],ans[20][20];
        int row=0,col=0,k=-1,key,msglen,i,j;
        clrscr();
        printf("Enter MSG ==> ");
        gets(msg);
        printf("Enter Key ==> ");
        scanf("%d",&key);
        msglen=strlen(msg);
        for(i=0;i<key;i++)
        {
                for(j=0;j<msglen;j++)
                {
                        ans[i][j]='\n';
                }
        }
        for(i=0;i<msglen;i++)
        {
                ans[row][col++]=msg[i];
                if(row==0 || row==key-1)
                {
                        k=k*(-1);
                }
                row = row + k;
        }
        printf("Cipher text ==> ");
        for(i=0;i<key;i++)
```

```
        {
                for(j=0;j<msglen;j++)
                {
                        if(ans[i][j]!='\n')
                        {
                                printf("%c",ans[i][j]);
                        }
                }
        }
        getch();
}
```

● **Output:**

Practical – 9

Aim: Write a program to implement Play Fair cipher.
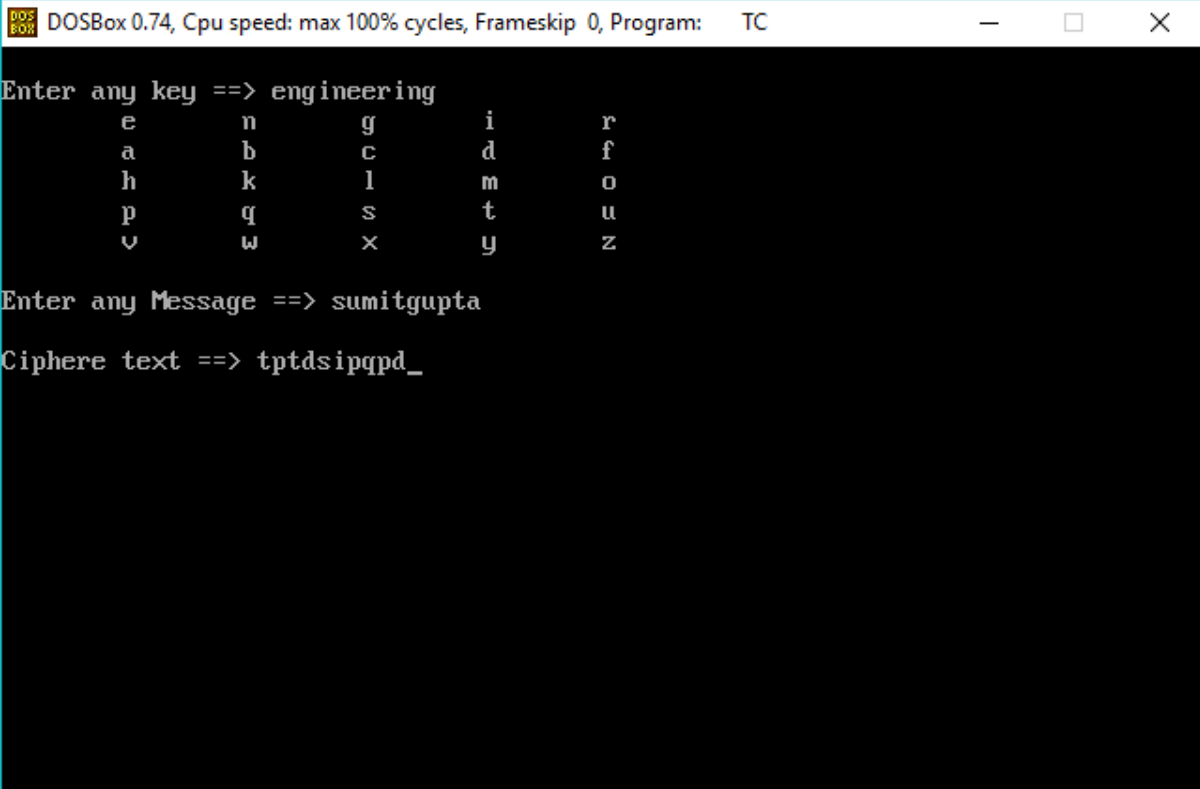
```c
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
    char key[200],play[5][5],msg[20],ct[20];
    int i,size,j,k,c1,c2,p,q,r1,r2,l;
    clrscr();
    printf("\nEnter any key ==> ");
    gets(key);
    strcat(key,"abcdefghiklmnopqrstuvwxyz");
    size=strlen(key);
    for(i=0;i<size;i++)
    {
        if(key[i]=='j')
        {
            key[i]='i';
        }
        for(j=i+1;j<size;)
        {
            if(key[j]==key[i])
            {
                for(k=j;k<size;k++)
                {
                    key[k]=key[k+1];
                }
                size--;
            }
            else
            {
                j++;
```

```
                    }
              }
       }
       for(i=0,k=0;i<5;i++)
       {
              for(j=0;j<5;j++)
              {
                     play[i][j]=key[k];
                     k++;
              }
       }
       for(i=0;i<5;i++)
       {
              for(j=0;j<5;j++)
              {
                     printf("\t%c",play[i][j]);
              }
              printf("\n");
       }
       printf("\nEnter any Message ==> ");
       gets(msg);
       printf("\nCiphere text ==> ");
       j=1;
       for(i=0;i<strlen(msg);)
       {
              r1=0;
              r2=0;
              c1=0;
              c2=0;
              p=0;
              q=0;
              p=msg[i];
              q=msg[j];
              if(p=='j')
```

```c
                {
                        msg[i]='i';
                }
                if(q=='j')
                {
                        msg[j]='i';
                }
                for(k=0;k<5;k++)
                {
                        for(l=0;l<5;l++)
                        {
                                if(play[k][l]==p)
                                {
                                        r1=k;
                                        c1=l;
                                }
                                if(play[k][l]==q)
                                {
                                        r2=k;
                                        c2=l;
                                }
                        }
                }
                if(r1==r2)
                {
                        ct[i]=play[r1][(c1+1)%5];
                        ct[j]=play[r2][(c2+1)%5];
                        printf("%c%c",ct[i],ct[j]);
                }
                else if(c1==c2)
                {
                        ct[i]=play[(r1+1)%5][c1];
                        ct[j]=play[(r2+1)%5][c2];
                        printf("%c%c",ct[i],ct[j]);
```

```
        }
        else
        {
                ct[i]=play[r1][c2];
                ct[j]=play[r2][c1];
                printf("%c%c",ct[i],ct[j]);
        }
        i=i+2;
        j=j+2;
    }
    getch();
}
```

● **Output:**

Practical – 10

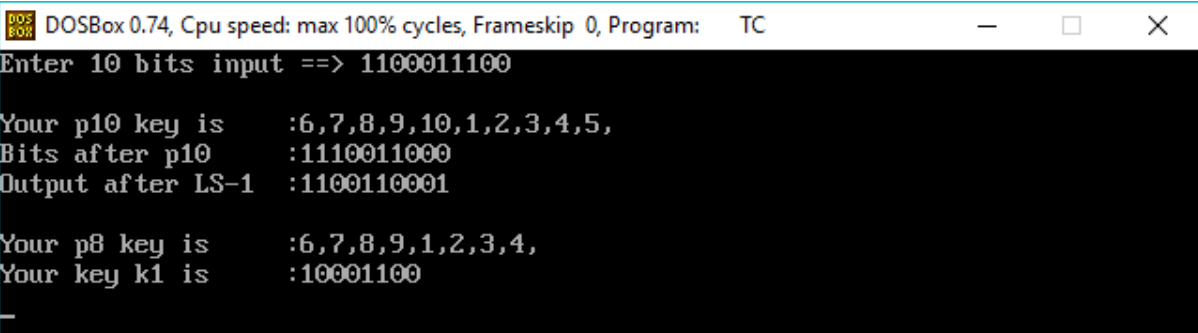**Aim: Write a program to implement S-DES key generation algorithm.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int i, cnt=0, p8[8]={6,7,8,9,1,2,3,4};
        int p10[10]={6,7,8,9,10,1,2,3,4,5};
        char input[11], k1[10], k2[10], temp[11];
        char LS1[5], LS2[5];
        clrscr();
        printf("Enter 10 bits input ==> ");
        scanf("%s",input);
        input[10]='\0';
        for(i=0; i<10; i++)
        {
                cnt = p10[i];
                temp[i] = input[cnt-1];
        }
        temp[i]='\0';
        printf("\nYour p10 key is    :");
        for(i=0; i<10; i++)
        {
                printf("%d,",p10[i]);
        }
        printf("\nBits after p10     :");
        puts(temp);
        for(i=0; i<5; i++)
        {
                if(i==4)
                        temp[i]=temp[0];
                else
                        temp[i]=temp[i+1];
```

```
        }
        for(i=5; i<10; i++)
        {
                if(i==9)
                        temp[i]=temp[5];
                else
                        temp[i]=temp[i+1];
        }
        printf("Output after LS-1  :");
        puts(temp);
        printf("\nYour p8 key is    :");
        for(i=0; i<8; i++)
        {
                printf("%d,",p8[i]);
        }
        for(i=0; i<8; i++)
        {
                cnt = p8[i];
                k1[i] = temp[cnt-1];
        }
        k1[i]='\0';
        printf("\nYour key k1 is    :");
        puts(k1);
        getch();
}
```

● **Output:**

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:    TC         —    □    ×
Enter 10 bits input ==> 1100011100

Your p10 key is    :6,7,8,9,10,1,2,3,4,5,
Bits after p10     :1110011000
Output after LS-1  :1100110001

Your p8 key is     :6,7,8,9,1,2,3,4,
Your key k1 is     :10001100
```
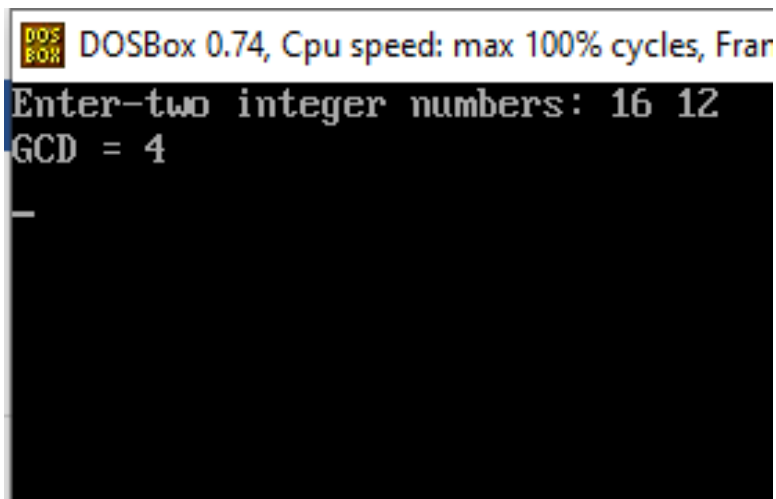
Practical – 11

Aim: Implement Euclid algorithm to find GCD.

```c
#include<stdio.h>
#include<conio.h>
void main() {
    int m, n; /* given numbers */
    clrscr();
    printf("Enter-two integer numbers: ");
    scanf ("%d %d", &m, &n);
    while (n > 0) {
        int r = m % n;
        m = n;
        n = r;
    }
    printf ("GCD = %d \n",m);
    getch();
}
```

- **Output:**

Practical – 12

Aim: Perform various Encryption-Decryption techniques with cryptool.

● **Introduction to cryptool**

CrypTool is an open source-learning tool illustrating cryptographic and cryptanalytic concepts. CrypTool implements more than 300 algorithms. Users can adjust these with own parameters. The graphical interface, online documentation, analytic tools and algorithms of CrypTool introduce users to the field of cryptography. Classical ciphers are available alongside asymmetric cryptography including RSA, elliptic curve cryptography, digital signatures, homomorphic encryption, and Diffie–Hellman key exchange, many of which are visualized by animations.CrypTool also contains some didactical games, and an animated tutorial about primes and elementary number theory.
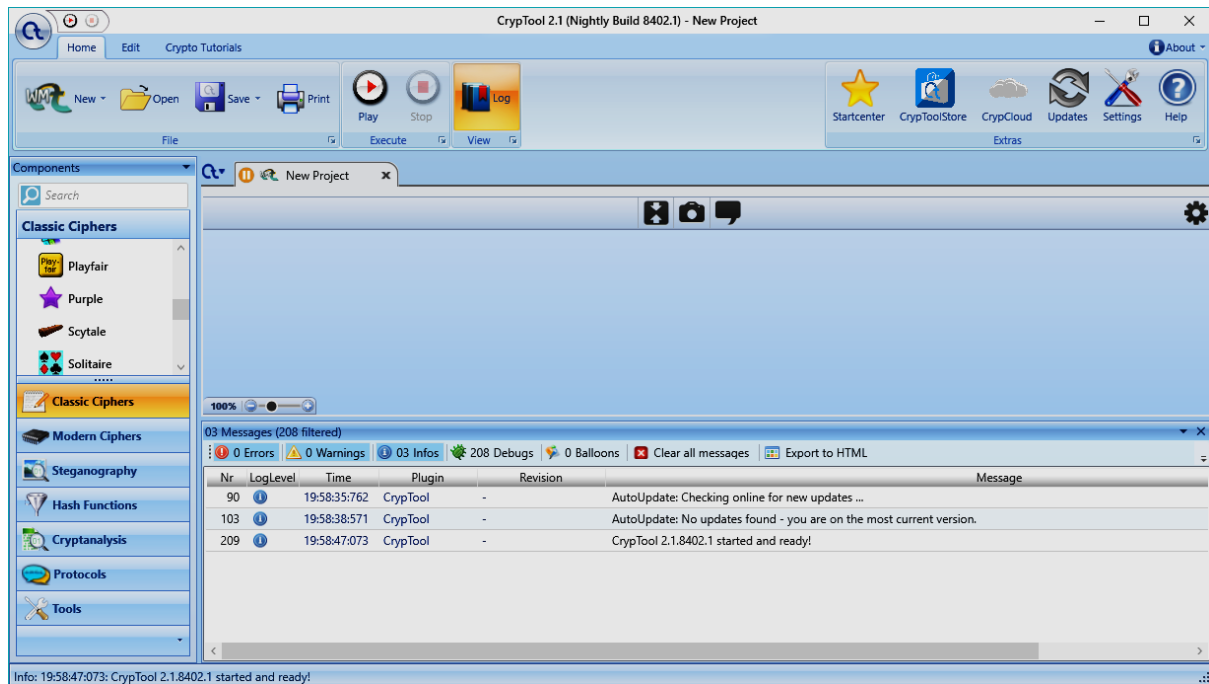
Step 1:

First download the cryptool. Then install the cryptool. Open cryptool. When it is first time launch it seen like as given in figure.
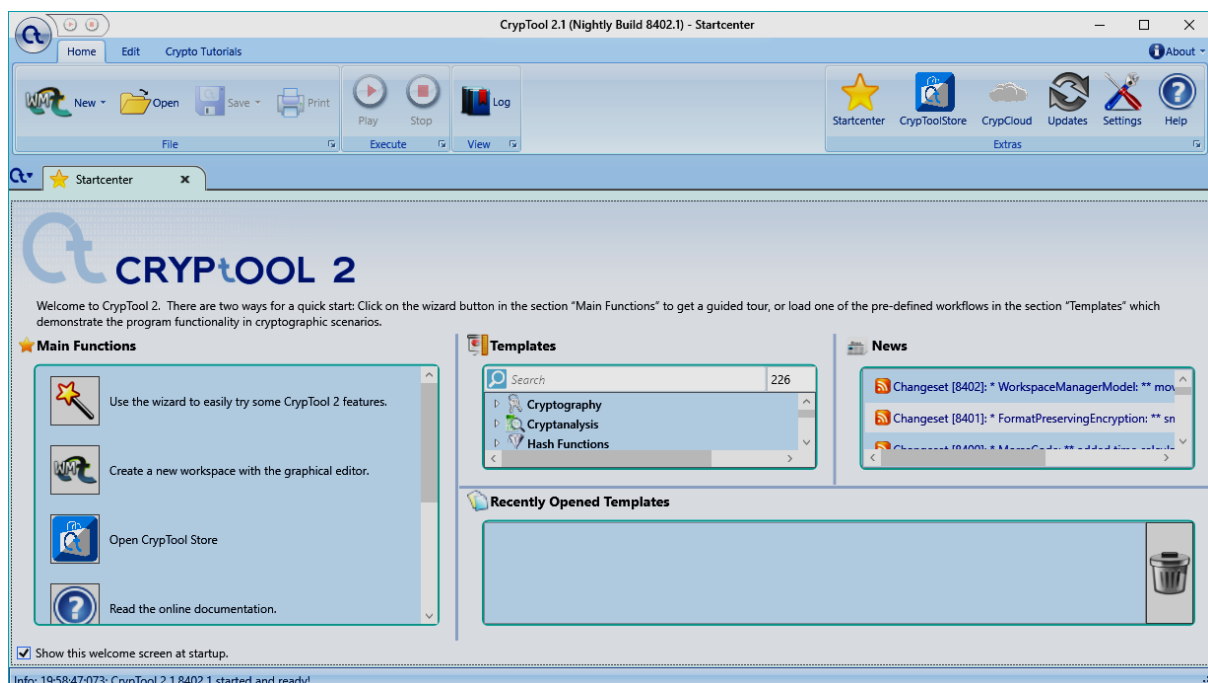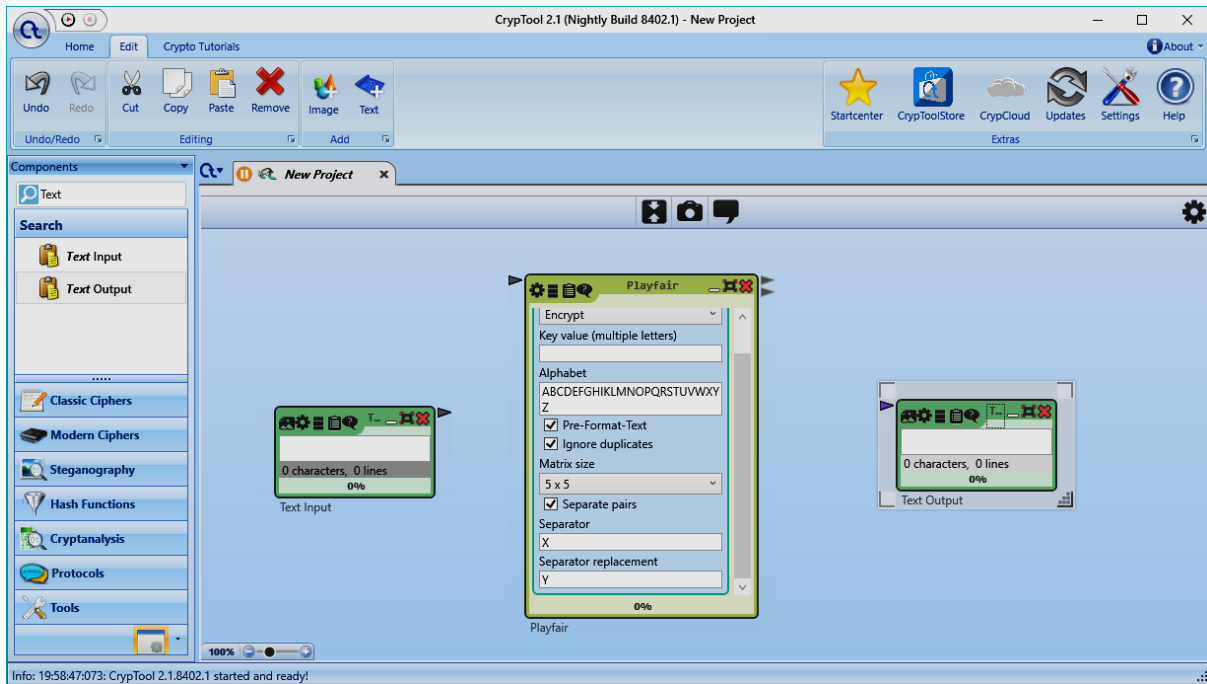
Step 2:

Click on new for performing any Encryption – Decryption techniques.


Step 3:

Here we want to perform playfair cipher encryption so from classic cipher drag playfair and drop to new project screen. And also drag Text input and output.
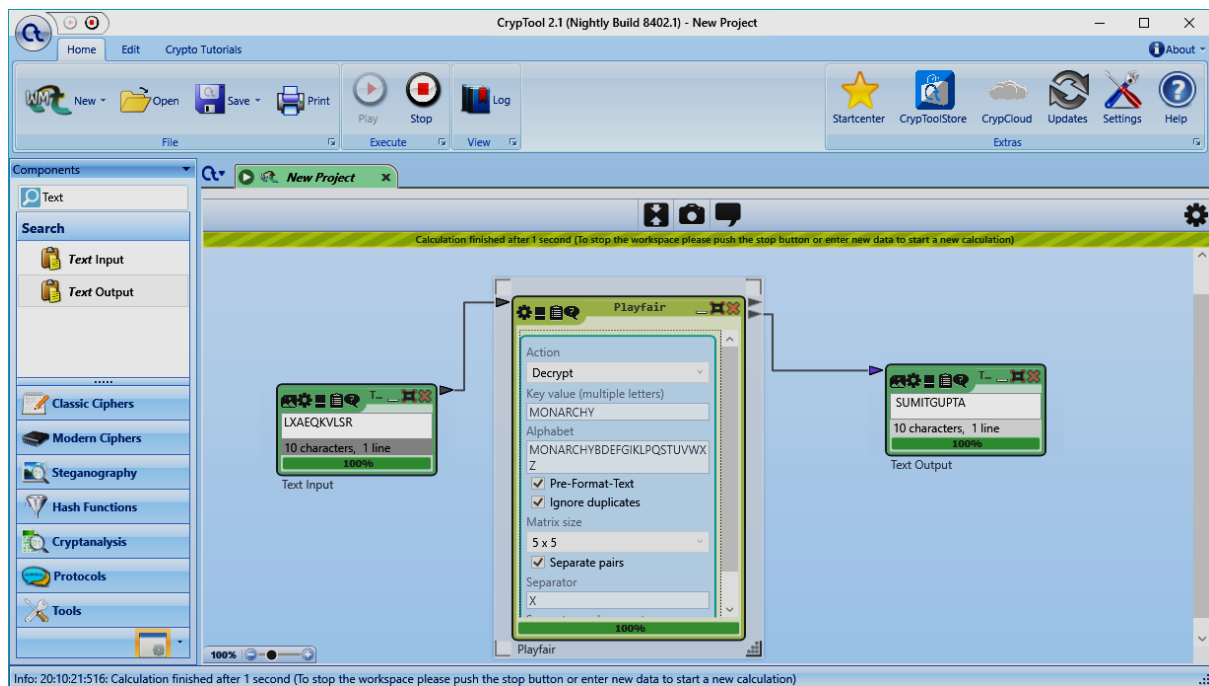
Step 4:

Connect all connections properly. playfair has plain text as text input and key is given by playfair box and ere we have to select encrypt or decrypt. Output side there are two text output in that one is for output and another is for preformatted string.
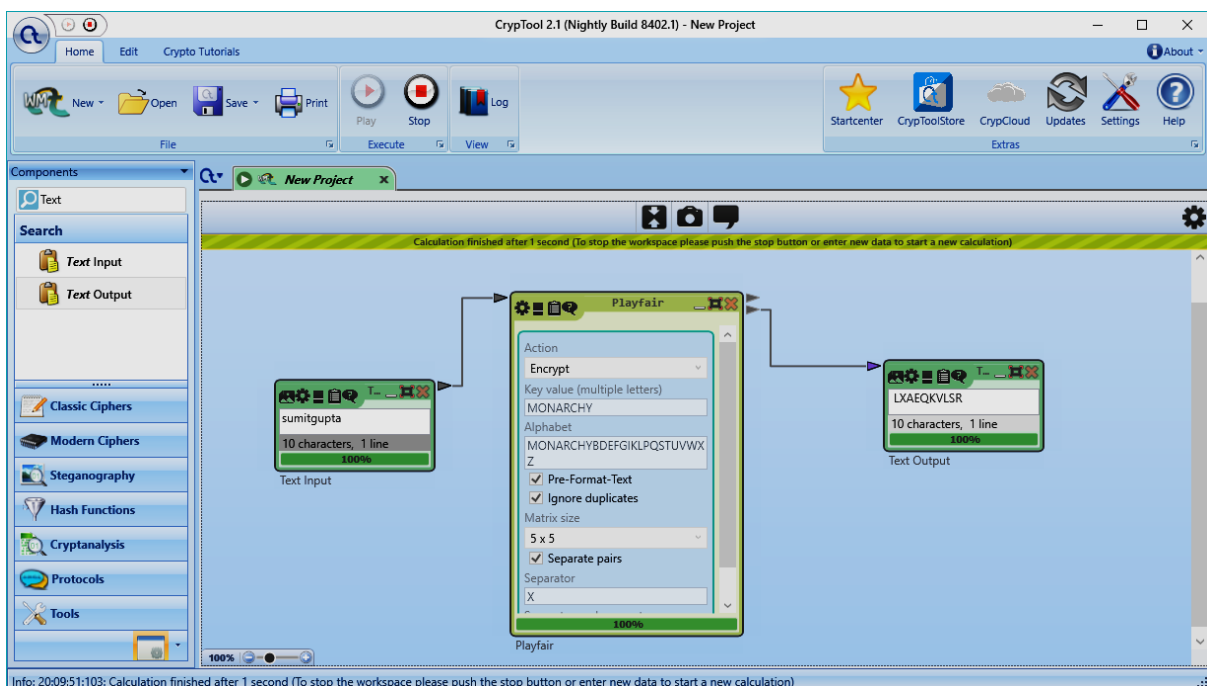
Step 5:

This is for decryption.

Practical – 13

Aim: Study and use the Wire shark for the various network protocols.

- **What is Wireshark?**

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible. In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, all that has changed. Wireshark is perhaps one of the best open source packet analyzers available today.

- **Features:**

Available for UNIX and Windows.

Capture live packet data from a network interface.

Open files containing packet data captured with tcpdump/WinDump, Wireshark, and a number of other packet capture programs.

Import packets from text files containing hex dumps of packet data.

Display packets with very detailed protocol information.

Save packet data captured.

Export some or all packets in a number of capture file formats.

Filter packets on many criteria.

Search for packets on many criteria.

Colorize packet display based on filters.

Create various statistics.

Data can be captured from the wire from a live network connection or read from a file that recorded already captured packets.

Captured network data can be browsed via a GUI, or Command Line.

Captured files can be programmatically edited or converted via command-line switches to the "editcap" program.

Data display can be refined using a display filter.

Plug-ins can be created for dissecting new protocols.

VoIP calls in the captured traffic can be detected. If encoded in a compatible encoding, the media flow can even be played.

- **Running Wireshark**

When you run the Wireshark program, the Wireshark graphical user interface shown in Figure 1 will be displayed. Initially, no data will be displayed in the various windows.

Packet list - Displays all of the packets in the trace in the order they were recorded.

Time – the timestamp at which the packet crossed the interface.

Source – the originating host of the packet.

Destination – the host to which the packet was sent.

Protocol – the highest level protocol that Wireshark can detect.

Length – the length in bytes of the packet on the wire.

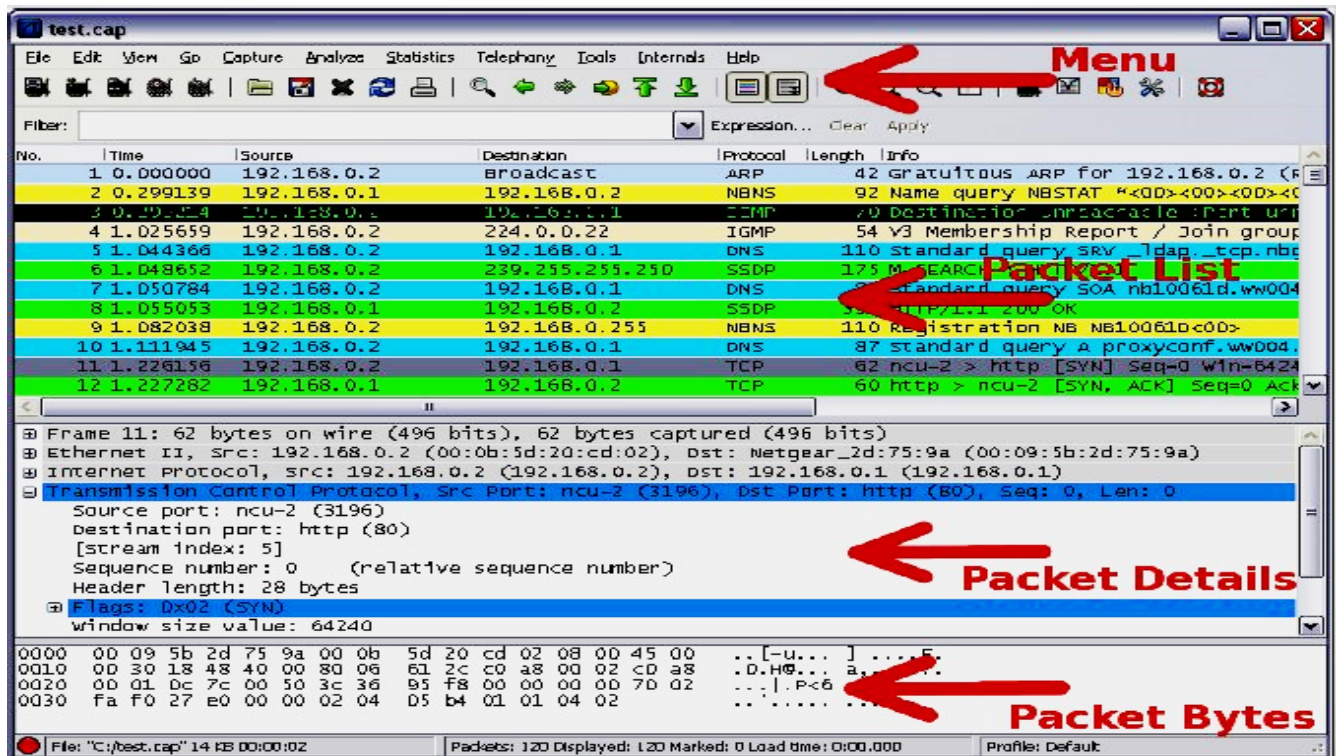Info – an informational message pertaining to the protocol in the protocol column.

Default Coloring Gray – TCP packets

Black with red letters – TCP Packets with errors

Green – HTTP Packets

Light Blue – UDP Packets

Pale Blue – ARP Packets

Lavender – ICMP Packets

Black with green letters – ICMP Packets with error

The Wireshark interface has five major components:

1. **command menus**

2. **packet-listing window**

3. **packet-header details window**

4. **packet-contents window**

5. **packet display filter field**

1. command menus

The command menus are standard pulldown menus located at the top of the window. Of interest to us now are the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data, and exit the Wireshark application. The Capture menu allows you to begin packet capture.

2. packet-listing window

The packet-listing window displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is not a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.

3. packet-header details window

The packet-header details window provides details about the packet selected (highlighted) in the packet listing window. (To select a packet in the packet listing window, place the cursor over the packet's one-line summary in the packet listing window and click with the left mouse button.). These details include information about the Ethernet frame (assuming the packet was sent/received over an Ethernet interface) and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or minimized by clicking on the plus-or-minus boxes to the left of the Ethernet frame or IP datagram line in the packet details window. If the packet has been carried over TCP or UDP, TCP or UDP details will also be displayed, which can similarly be expanded or minimized. Finally, details about the highest level protocol that sent or received this packet are also provided.

4. packet-contents window

The packet-contents window displays the entire contents of the captured frame, in both ASCII and hexadecimal format.

5. packet displayfilter field

Towards the top of the Wireshark graphical user interface, is the packet displayfilter field, into which a protocol name or other information can be entered in order to *filter* the information displayed in the packet-listing window (and hence the packet-header and packet-contents windows). In the example below, we'll use the packet-display filter field to have Wireshark hide (not display) packets except those that correspond to HTTP messages.

● **Wireshark problem:**

Wireshark isn't an intrusion detection system. It will not warn you when someone does strange things on your network that he/she isn't allowed to do. However, if strange things happen, Wireshark might help you figure out what is really going on.

Wireshark will not manipulate things on the network, it will only "measure" things from it. Wireshark doesn't send packets on the network or do other active things (except for name resolutions, but even that can be disabled)