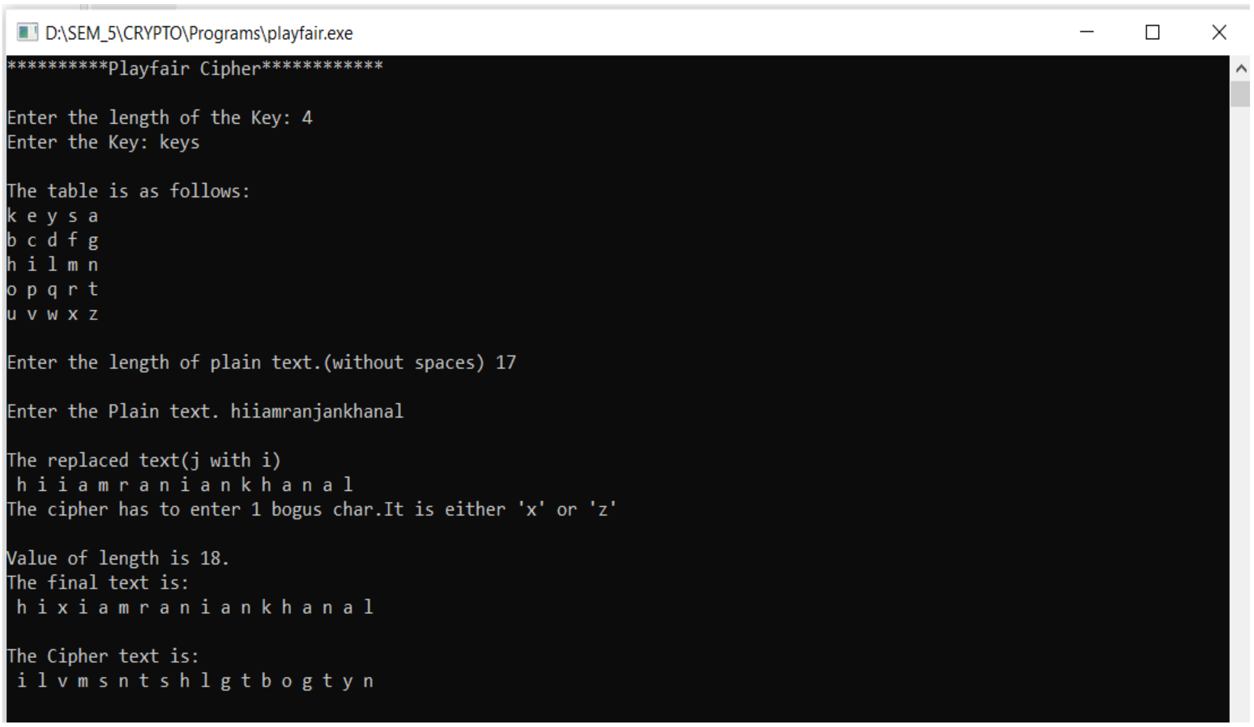
INDEX

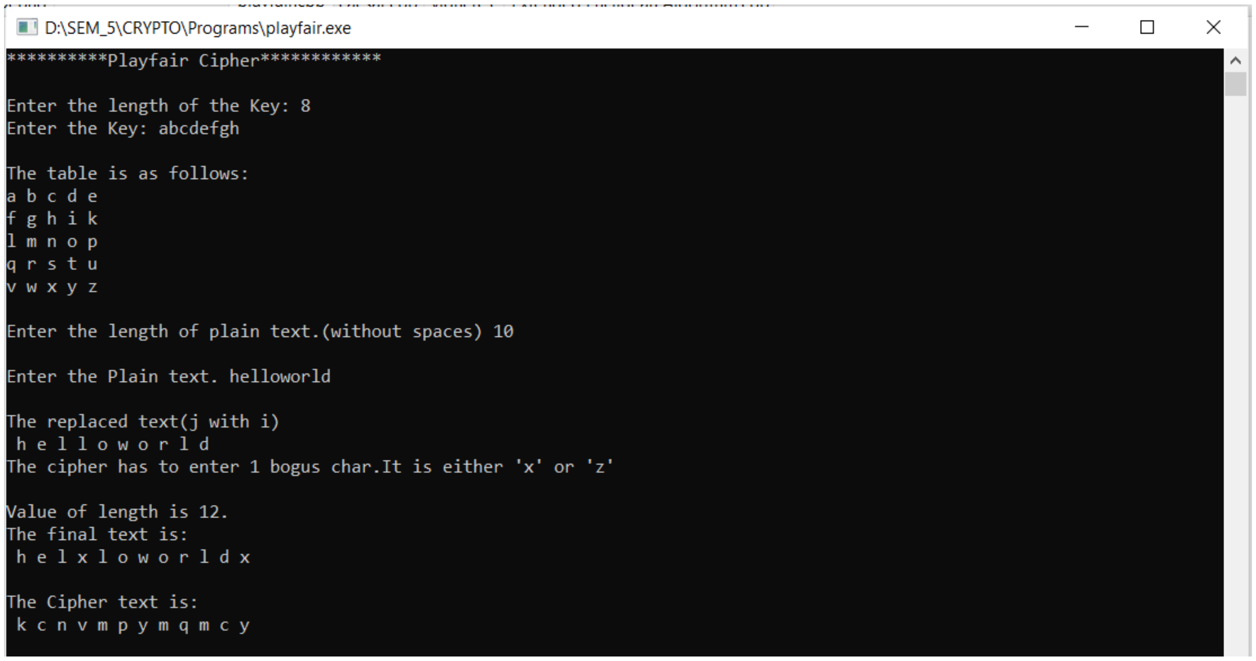
PLAYFAIR CIPHER

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
int check(char table[5][5],char k)
{
 int i,j;
 for(i=0;i<5;++i)
 for(j=0;j<5;++j)
 {
 if(table[i][j]==k)
 return 0;
 }
 return 1;
}
int main()
{
 int i,j,key_len;
 char table[5][5];
 for(i=0;i<5;++i)
 for(j=0;j<5;++j)
 table[i][j]='0';
 printf("**********Playfair
Cipher************\n\n");
 printf("Enter the length of the Key:
");
 scanf("%d",&key_len);
 char key[100];
 printf("Enter the Key: ");
 for(i=-1;i<key_len;++i)
 {
 scanf("%c",&key[i]);
 if(key[i]=='j')
 key[i]='i';
 }
 int flag;
 int count=0;
 // inserting the key into the table
 for(i=0;i<5;++i)
 {
 for(j=0;j<5;++j)
 {
 flag=0;
 while(flag!=1)
 {
 if(count>key_len)
 goto l1;
 flag=check(table,key[count]);
 ++count;
 }// end of while
 table[i][j]=key[(count-1)];
 }// end of inner for
 }// end of outer for
 l1:printf("\n");
 int val=97;
 //inserting other alphabets
 for(i=0;i<5;++i)
 {
 for(j=0;j<5;++j)
 {
 if(table[i][j]>=97 &&
table[i][j]<=123)
 {}
 else
 {
 flag=0;
 while(flag!=1)
 {
 if('j'==(char)val)
 ++val;
 flag=check(table,(char)val);
 ++val;
 }// end of while
 table[i][j]=(char)(val-1);
 }//end of else
 }// end of inner for
 }// end of outer for
 printf("The table is as follows:\n");
 for(i=0;i<5;++i)
 {
 for(j=0;j<5;++j)
 {
 printf("%c ",table[i][j]);
 }
 printf("\n");
 }
 int l=0;
 printf("\nEnter the length of plain
text.(without spaces) ");
 scanf("%d",&l);
 printf("\nEnter the Plain text. ");
 char p[100];
 for(i=-1;i<l;++i)
 {
 scanf("%c",&p[i]);
 }
 for(i=-1;i<l;++i)
 {
 if(p[i]=='j')
 p[i]='i';
```

```
 }
 printf("\nThe replaced text(j with
i)");
 for(i=-1;i<l;++i)
 printf("%c ",p[i]);
 count=0;
 for(i=-1;i<l;++i)
 {
 if(p[i]==p[i+1])
 count=count+1;
 }
 printf("\nThe cipher has to enter %d
bogus char.It is either 'x' or
'z'\n",count);
 int length=0;
 if((l+count)%2!=0)
 length=(l+count+1);
 else
 length=(l+count);
 printf("\nValue of length is
%d.\n",length);
 char p1[100];
 //inserting bogus characters.
 //char temp1;
 int count1=0;
 for(i=-1;i<l;++i)
 {
p1[count1]=p[i];
if(p[i]==p[i+1])
{
count1=count1+1;
if(p[i]=='x')
p1[count1]='z';
else
p1[count1]='x';
}
count1=count1+1;
}
//checking for length
//char bogus;
if((l+count)%2!=0)
{
if(p1[length-1]=='x')
p1[length]='z';
else
p1[length]='x';
}
printf("The final text is:");
for(i=0;i<=length;++i)
printf("%c ",p1[i]);
char cipher_text[100];
int r1,r2,c1,c2;
 int k1;
 for(k1=1;k1<=length;++k1)
 {
 for(i=0;i<5;++i)
 {
 for(j=0;j<5;++j)
 {
 if(table[i][j]==p1[k1])
 {
 r1=i;
 c1=j;
 }
 else
 if(table[i][j]==p1[k1+1])
 {
 r2=i;
 c2=j;
 }
 }//end of for with j
 }//end of for with i
 if(r1==r2)
 {
 cipher_text[k1]=table[r1][(c1+1)%5];
 cipher_text[k1+1]=table[r1][(c2+1)%
 5];
}
else
if(c1==c2)
{
cipher_text[k1]=table[(r1+1)%5][c1];

 cipher_text[k1+1]=table[(r2+1)%5][c
 1];
}
else
{
cipher_text[k1]=table[r1][c2];
cipher_text[k1+1]=table[r2][c1];
}
k1=k1+1;
}//end of for with k1
printf("\n\nThe Cipher text is:\n ");
for(i=1;i<=length;++i)
printf("%c ",cipher_text[i]);
getch();
 }
```



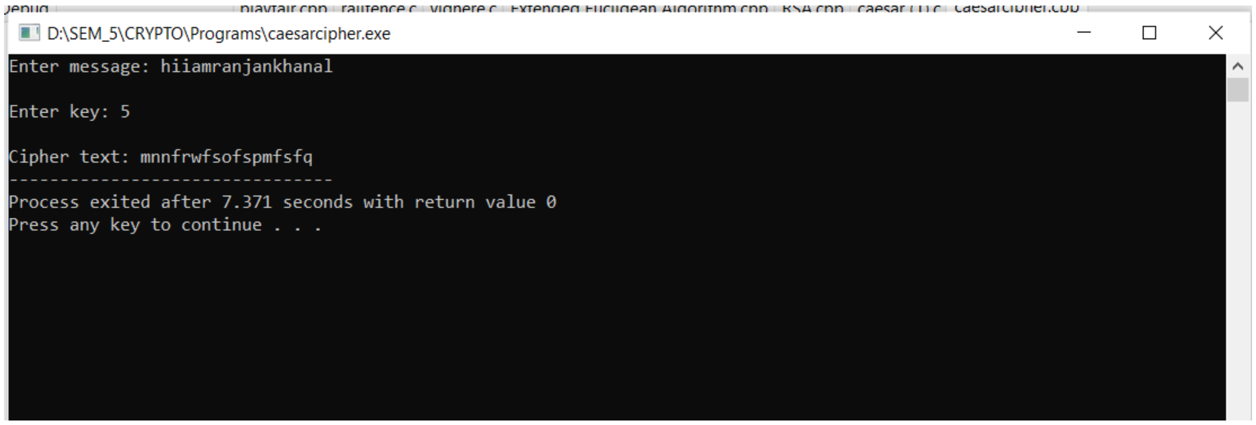
```
D:\SEM_5\CRYPTO\Programs\playfair.exe
**********Playfair Cipher************

Enter the length of the Key: 4
Enter the Key: keys

The table is as follows:
k e y s a
b c d f g
h i l m n
o p q r t
u v w x z

Enter the length of plain text.(without spaces) 17

Enter the Plain text. hiiamranjankhanal

The replaced text(j with i)
 h i i a m r a n i a n k h a n a l
The cipher has to enter 1 bogus char.It is either 'x' or 'z'

Value of length is 18.
The final text is:
 h i x i a m r a n i a n k h a n a l

The Cipher text is:
 i l v m s n t s h l g t b o g t y n
```



```
D:\SEM_5\CRYPTO\Programs\playfair.exe
**********Playfair Cipher************

Enter the length of the Key: 8
Enter the Key: abcdefgh

The table is as follows:
a b c d e
f g h i k
l m n o p
q r s t u
v w x y z

Enter the length of plain text.(without spaces) 10

Enter the Plain text. helloworld

The replaced text(j with i)
 h e l l o w o r l d
The cipher has to enter 1 bogus char.It is either 'x' or 'z'

Value of length is 12.
The final text is:
 h e l x l o w o r l d x

The Cipher text is:
 k c n v m p y m q m c y
```

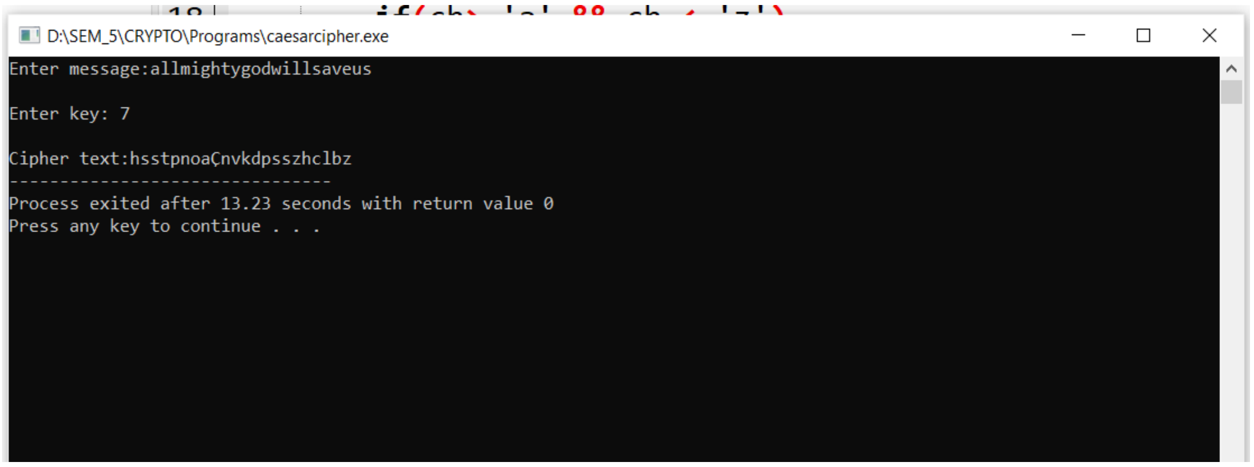
CAESAR CIPHER

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
char message[10],ch,c[10];
int key;
int i,len;
printf("Enter message:");
gets(message);
printf("\nEnter key:");
scanf("%d",&key);
//	len=strlen(message);
//	message[len]='\0';
for(i=0;i<strlen(message);i++)
{
ch=message[i];
if(ch>='a' && ch <='z')
{
ch=ch+key;
if (ch>'z')
ch=ch-26;
}
else if(ch>='A'&& ch<='Z')
{
ch=ch+key;
if (ch>'z')
ch=ch-26;
}
c[i]=ch;
}
c[i]='\0';
printf("\nCipher text:%s",c);
}
```



D:\SEM_5\CRYPTO\Programs\caesarcipher.exe

```
Enter message: hiiamranjankhanal

Enter key: 5

Cipher text: mnnfrwfsofspmfsfq
--------------------------------
Process exited after 7.371 seconds with return value 0
Press any key to continue . . .
```



D:\SEM_5\CRYPTO\Programs\caesarcipher.exe

```
Enter message:allmightygodwillsaveus

Enter key: 7

Cipher text:hsstpnoaÇnvkdpsszhclbz
--------------------------------
Process exited after 13.23 seconds with return value 0
Press any key to continue . . .
```

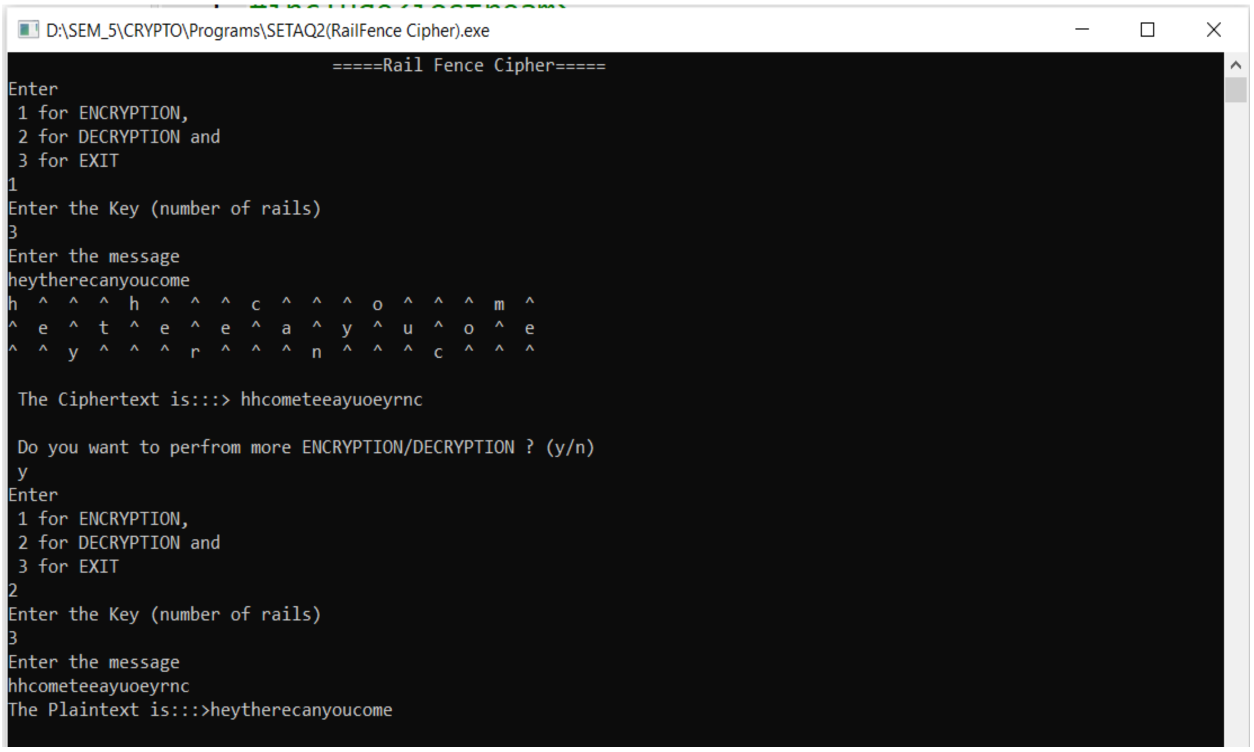
RAIL-FENCE CIPHER

```
//Rail Fence Cipher
#include<iostream>
#include<string>
using namespace std;
class RailFence{
public:
int nrow,ncol;
int getKey(){
int key;
cout<<"Enter the Key (number of rails) \n";
cin>>key;
return key;
}
string getMessage(){
string msg;
cout<<"Enter the message  \n";
cin.ignore();
getline(cin,msg);
return msg;
}
void encrypt(string msg, int key){
// creating a matrix to encrypt msg with key
// key = rows , length of msg=no. of characters = columns
nrow= key;
ncol= msg.length();
char rail_matrix[nrow][ncol];
// filling the rail matrix with ^ symbol
for (int i=0; i < nrow; i++) {
for (int j = 0; j < ncol; j++){
rail_matrix[i][j] ='^';
}
}
// to find the direction
bool downward = false;
int r = 0, c = 0;
string ciphertext;
for (int i=0; i < msg.length(); i++) {
// checking  the direction of flow
// reverse the direction if the top or bottom rail is just filled
if (r == 0 || r == key-1)
downward = !downward;
```

```
// filling  with  characters in the
plaintext

rail_matrix[r][c++] = msg[i];

// find the next row using direction

downward ?r++ : r--;

}

//to print the rail matrix

for (int i=0; i < nrow; i++) {

for (int j = 0; j < ncol; j++){

cout<< rail_matrix[i][j]<<"  ";

}

cout<<"\n";

}

// generating  the ciphertext using the
rail_matrix

for (int i=0; i < key; i++) {

for (int j=0; j < msg.length(); j++) {

if (rail_matrix[i][j]!='^')

ciphertext.push_back(rail_matrix[i][j]
); //appending a character

}

}

cout<<"\n The Ciphertext is:::>
"<<ciphertext<<"\n";

}

void decrypt(string msg, int key){

// creating a matrix to encrypt msg
with key

// key = rows , length of msg=no. of
characters = columns

nrow= key;

ncol= msg.length();

char rail_matrix[nrow][ncol];

string plaintext;

// filling the rail matrix with ^ symbol

for (int i=0; i < nrow; i++) {

for (int j = 0; j < ncol; j++){

rail_matrix[i][j] ='^';

}

}

// to find the direction

bool downward;

int r = 0, c= 0;

// marking  the places with '~'

for (int i=0; i < msg.length(); i++) {

// check the direction of flow

if (r == 0)

downward = true;

if (r == key-1)

downward = false;

// place the marker

rail_matrix[r][c++] = '~';
```

```
// find the next row using direction flag

downward?r++ : r--;

}

// filling  the rail matrix

int indx = 0;

for (int i=0; i<key; i++) {

for (int j=0; j<msg.length(); j++) {

if (rail_matrix[i][j] == '~' && indx<msg.length())

rail_matrix[i][j] = msg[indx++];

}

}

//  reading  the matrix in zig-zag order to get the plaintext

r = 0, c = 0;

for (int i=0; i< msg.length(); i++)

{

// check the direction of flow

if (r == 0)

downward = true;

if (r == key-1)

downward = false;

// checking  the marker

if (rail_matrix[r][c] != '~')

plaintext.push_back(rail_matrix[r][c++]); //appending

// finding  the next row using direction flag

downward?r++: r--;

}

cout<<"The Plaintext is:::>"<<plaintext<<"\n";

}

};

int main(){

cout<<"
          =====Rail Fence Cipher===== \n";

int choice;

char more;

RailFence rf;

int k;

string m;

do{

cout<<"Enter\n 1 for ENCRYPTION,\n 2 for DECRYPTION and\n 3 for EXIT \n";

cin>>choice;

switch(choice){

case 1:

k= rf.getKey();

m= rf.getMessage();

rf.encrypt(m,k);
```

```
break;
case 2:
k= rf.getKey();
m= rf.getMessage();
rf.decrypt(m,k);
break;
case 3:
break;
default:
cout<<"\n INVALID CHOICE! \n";
}
cout<<"\n Do you want to perfrom more ENCRYPTION/DECRYPTION ? (y/n)\n ";
cin>>more;
}
while(more=='y'|| more=='Y');
cout<<"\n\n Thank You! \n\n";
}
```

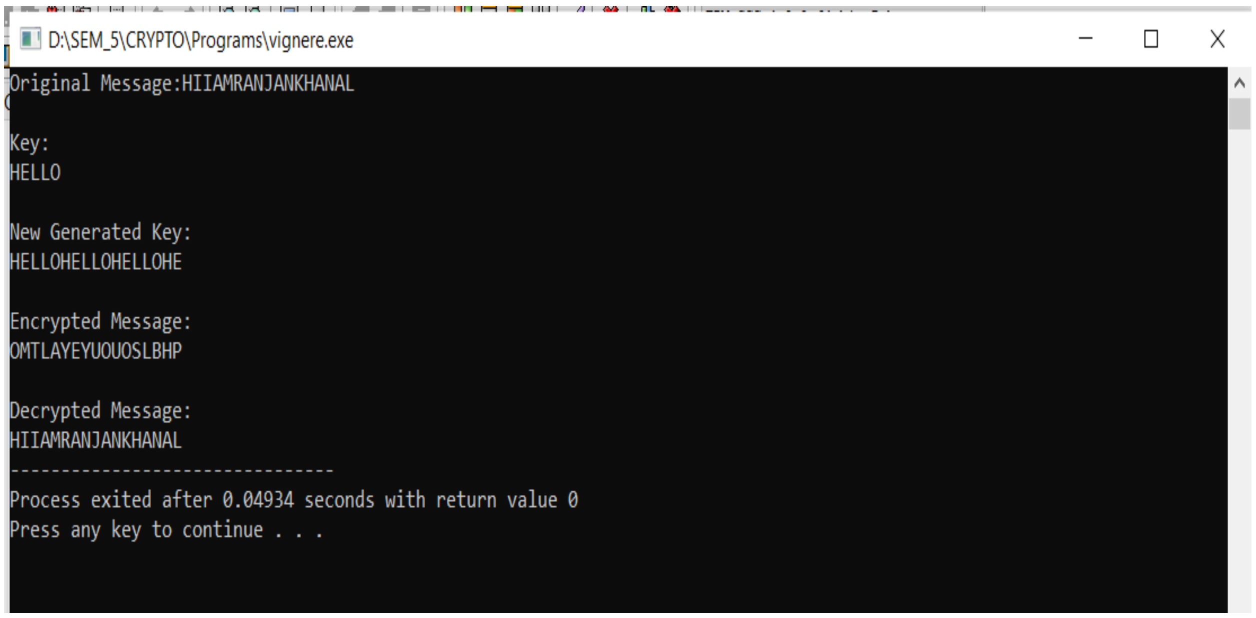


D:\SEM_5\CRYPTO\Programs\SETAQ2(RailFence Cipher).exe

```
                                =====Rail Fence Cipher=====
Enter
 1 for ENCRYPTION,
 2 for DECRYPTION and
 3 for EXIT
1
Enter the Key (number of rails)
3
Enter the message
heytherecanyoucome
h  ^  ^  ^  h  ^  ^  ^  c  ^  ^  ^  o  ^  ^  ^  m  ^
^  e  ^  t  ^  e  ^  e  ^  a  ^  y  ^  u  ^  o  ^  e
^  ^  y  ^  ^  ^  r  ^  ^  ^  n  ^  ^  ^  c  ^  ^  ^

 The Ciphertext is:::> hhcometeeayuoeyrnc

 Do you want to perfrom more ENCRYPTION/DECRYPTION ? (y/n)
 y
Enter
 1 for ENCRYPTION,
 2 for DECRYPTION and
 3 for EXIT
2
Enter the Key (number of rails)
3
Enter the message
hhcometeeayuoeyrnc
The Plaintext is:::>heytherecanyoucome
```

VIGNERE CIPHER

```
#include<stdio.h>
#include<string.h>
int main(){
char msg[] =
"HIIAMRANJANKHANAL";
char key[] = "HELLO";
int msgLen = strlen(msg), keyLen =
strlen(key), i, j;
char newKey[msgLen],
encryptedMsg[msgLen],
decryptedMsg[msgLen];
//generating new key
for(i = 0, j = 0; i < msgLen; ++i, ++j){
if(j == keyLen)
j = 0;
newKey[i] = key[j];
}
newKey[i] = '\0';
//encryption
for(i = 0; i < msgLen; ++i)
encryptedMsg[i] = ((msg[i] +
newKey[i]) % 26) + 'A';
encryptedMsg[i] = '\0';
//decryption
for(i = 0; i < msgLen; ++i)
decryptedMsg[i] = (((encryptedMsg[i]
- newKey[i]) + 26) % 26) + 'A';
decryptedMsg[i] = '\0';
printf("Original Message:%s", msg);
printf("\nKey:\n%s", key);
printf("\nNew Generated Key:\n%s",
newKey);
printf("\nEncrypted Message:\n%s",
encryptedMsg);
printf("\nDecrypted Message:\n%s",
decryptedMsg);
return 0;
}
```

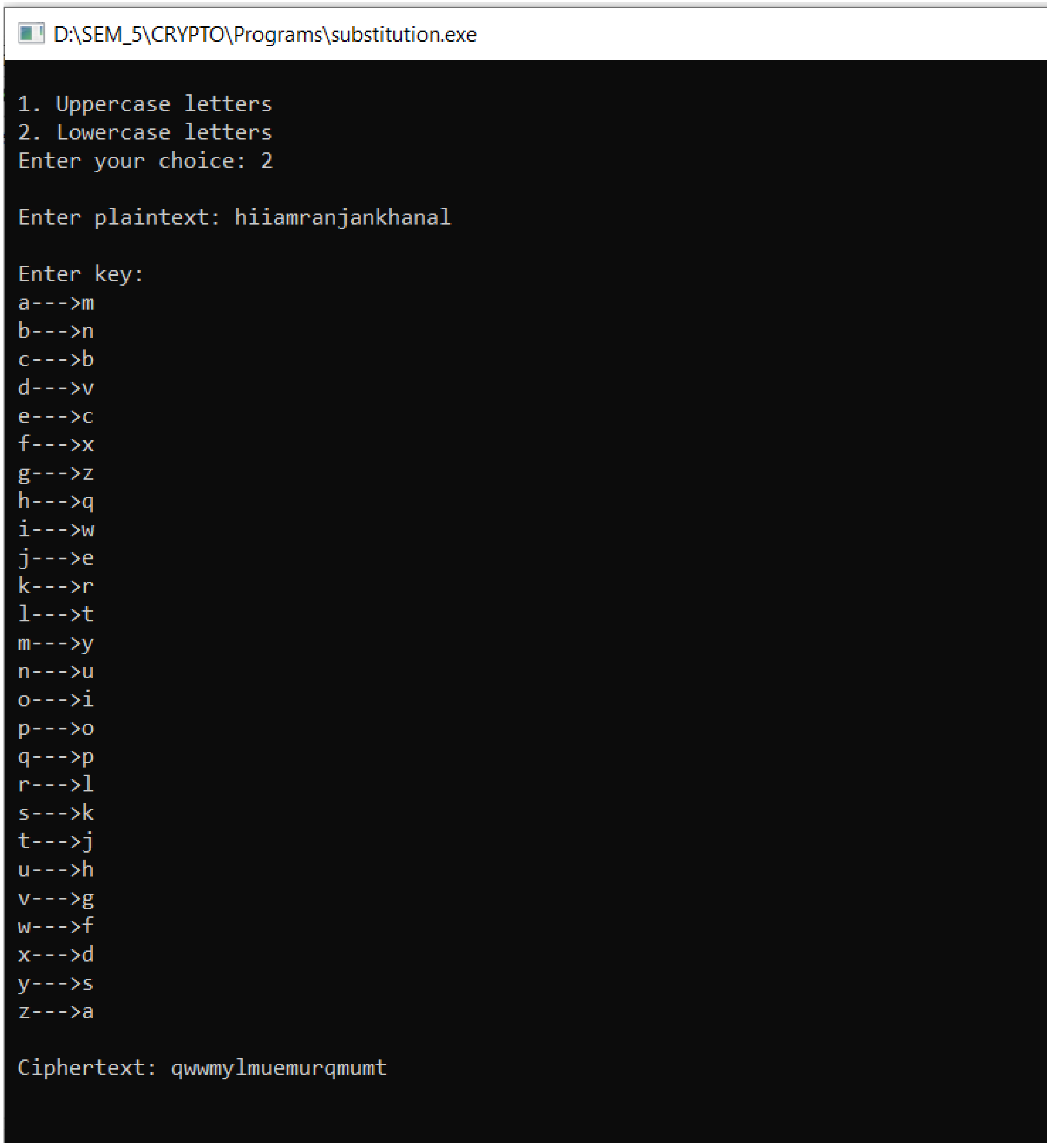

D:\SEM_5\CRYPTO\Programs\vignere.exe

```
Original Message:HIIAMRANJANKHANAL

Key:
HELLO

New Generated Key:
HELLOHELLOHELLOHE

Encrypted Message:
OMTLAYEYUOUOSLBHP

Decrypted Message:
HIIAMRANJANKHANAL
--------------------------------
Process exited after 0.04934 seconds with return value 0
Press any key to continue . . .
```

MONOALPHABETIC SUBSTITUTION CIPHER

```
#include<iostream>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
using namespace std;
char p[100], c[100],k[100];
int i,j,index;
void upcipher();
void lowcipher();
int check_unique(char k[],int i)
{
for(j=0;j<i;j++)
{
if(k[j]==k[i])
{
return (1);
}
else
{
return(0);
}
}
}
```

```
void upcipher()
{
int u;
cout<<"\n Plaintext: ";
cin>>p;
cout<<"\n Enter key: "<<endl;
for(i=0; i<26; i++)
{
loop:
cout<<" "<<char(i+65)<<"--->";
cin>>k[i];
u=check_unique(k,i);
if(u==1)
{
cout<<"\n Enter unique key";
goto loop;
}
}
for(i=0;i<strlen(p);i++)
{
index=p[i]-65;
c[i]=k[index];
}
cout<<"\n Ciphertext: "<<c;
}
void lowcipher()
{
int u;
cout<<"\n Enter plaintext: ";
cin>>p;
cout<<"\n Enter key: "<<endl;
for(i=0; i<26; i++)
{
loop:
cout<<" "<<char(i+97)<<"--->";
cin>>k[i];
u=check_unique(k,i);
if(u==1)
{
cout<<"\n Enter unique key";
goto loop;
}
}
for(i=0;i<strlen(p);i++)
{
index=p[i]-97;
c[i]=k[index];
}
cout<<"\n Ciphertext: "<<c;
}
int main()
{
int ch;
cout<<"\n 1. Uppercase letters \n 2.
Lowercase letters";
cout<<"\n Enter your choice: ";
cin>>ch;
if(ch==1)
upcipher();
else if (ch==2)
lowcipher();
else
cout<<"\n Invalid choice"; getch();
}
```

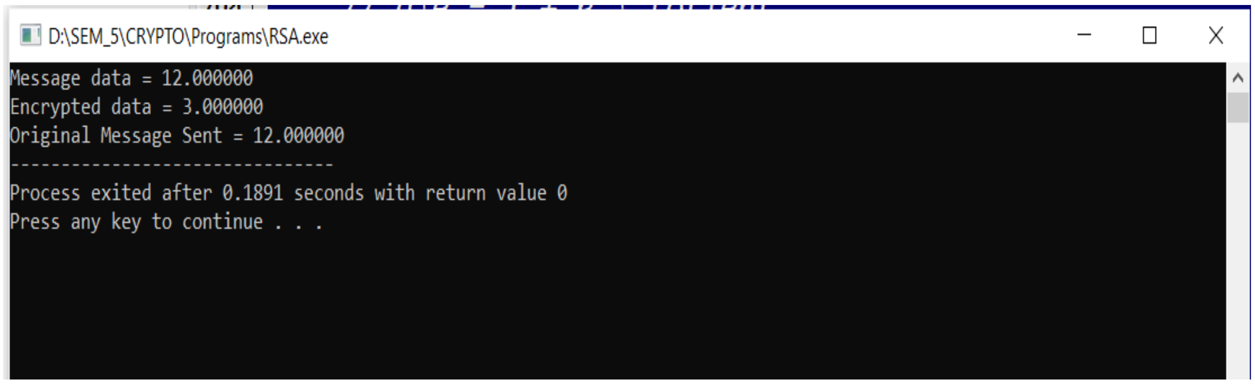

D:\SEM_5\CRYPTO\Programs\substitution.exe

```
1. Uppercase letters
2. Lowercase letters
Enter your choice: 2

Enter plaintext: hiiamranjankhanal

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

Ciphertext: qwwmylmuemurqmumt
```

ASSYMETRIC ALGORITHM (RSA)

For demonstration values are relatively small compared to practical application

```
#include<stdio.h>
#include<math.h>
// Returns gcd of a and b
int gcd(int a, int h)
{
int temp;
while (1)
{
temp = a%h;
if (temp == 0)
return h;
a = h;
h = temp;
}
}
// Code to demonstrate RSA algorithm
int main()
{
// Two random prime numbers
double p = 3;
double q = 7;
// First part of public key:
double n = p*q;
// Finding other part of public key.
// e stands for encrypt
double e = 2;
double phi = (p-1)*(q-1);
while (e < phi)
{
// e must be co-prime to phi and
// smaller than phi.
if (gcd(e, phi)==1)
break;
else
e++;
}
// Private key (d stands for decrypt)
// choosing d such that it satisfies
// d*e = 1 + k * totient
int k = 2; // A constant value
double d = (1 + (k*phi))/e;
// Message to be encrypted
double msg = 12;
printf("Message data = %lf", msg);
// Encryption c = (msg ^ e) % n
double c = pow(msg, e);
c = fmod(c, n);
printf("\nEncrypted data = %lf", c);
// Decryption m = (c ^ d) % n
double m = pow(c, d);
m = fmod(m, n);
printf("\nOriginal Message Sent = %lf", m);
return 0;
}
```



D:\SEM_5\CRYPTO\Programs\RSA.exe

```
Message data = 12.000000
Encrypted data = 3.000000
Original Message Sent = 12.000000
--------------------------------
Process exited after 0.1891 seconds with return value 0
Press any key to continue . . .
```