**CRYPTOGRAPHY AND NETWORK SECURITY**

**1.CEASER CIPHER :**

**Program :**

#include <stdio.h>

#include <string.h>

#include<conio.h>

#include <ctype.h>

int main()

{

char plain[10], cipher[10];

int key,i,length;

int result;

printf("\n Enter the plain text:");

scanf("%s", plain);

printf("\n Enter the key value:");

scanf("%d", &key);

printf("\n \n \t PLAIN TEXt: %s",plain);

printf("\n \n \t ENCRYPTED TEXT: ");

for(i = 0, length = strlen(plain); i < length; i++)

{

cipher[i]=plain[i] + key;

if (isupper(plain[i]) && (cipher[i] > 'Z'))

cipher[i] = cipher[i] - 26;

if (islower(plain[i]) && (cipher[i] > 'z'))

cipher[i] = cipher[i] - 26;

printf("%c", cipher[i]);

}

printf("\n \n \t AFTER DECRYPTION : ");

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

{

plain[i]=cipher[i]-key;

if(isupper(cipher[i])&&(plain[i]<'A'))

plain[i]=plain[i]+26;

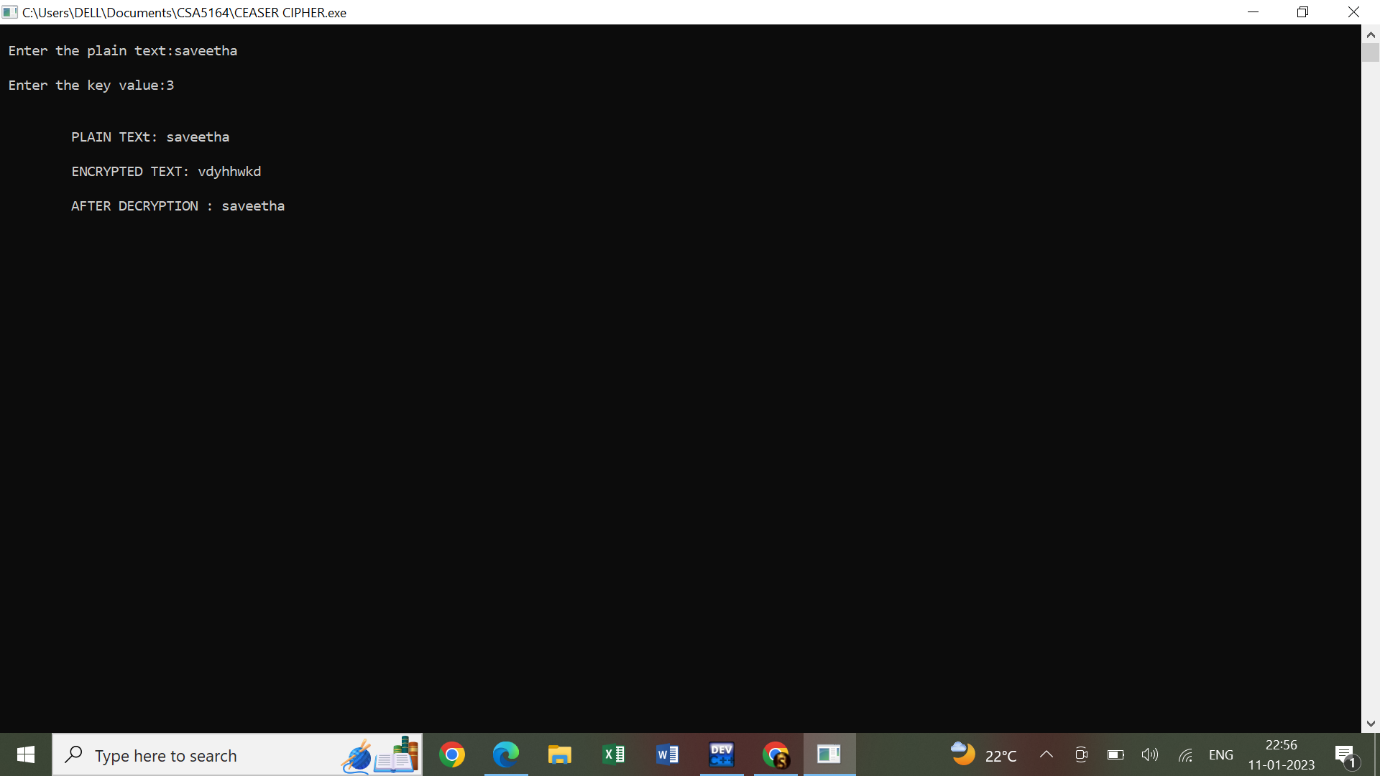
if(islower(cipher[i])&&(plain[i]<'a'))

plain[i]=plain[i]+26;

printf("%c",plain[i]);

}

getch();

}**OUT PUT :**

**2.VIGENERE CIPHER :**

**PROGRAM :**

#include<stdio.h>

#include<string.h>

#include<ctype.h>

#include<stdlib.h>

main()

{

int i,j,k,numstr[100],numkey[100],numcipher[100];

char str[100],key[100];

printf("Enter a string\n");

gets(str);

for(i=0,j=0;i<strlen(str);i++)

{

if(str[i]!=' ')

{

str[j]=toupper(str[i]);

j++;

}

}

str[j]='\0';

printf("Entered string is : %s \n",str);

for(i=0;i<strlen(str);i++)

{

numstr[i]=str[i]-'A';

}

printf("Enter a key\n");

gets(key);

for(i=0,j=0;i<strlen(key);i++)

{

if(key[i]!=' ')

{

key[j]=toupper(key[i]);

j++;

}

}

key[j]='\0';

for(i=0;i<strlen(str);)

{

for(j=0;(j<strlen(key))&&(i<strlen(str));j++)

{

numkey[i]=key[j]-'A';

i++;

}

}

for(i=0;i<strlen(str);i++)

{

numcipher[i]=numstr[i]+numkey[i];

}

for(i=0;i<strlen(str);i++)

{

if(numcipher[i]>25)

{

numcipher[i]=numcipher[i]-26;

}

}

printf("Vigenere Cipher text is\n");

for(i=0;i<strlen(str);i++)

{

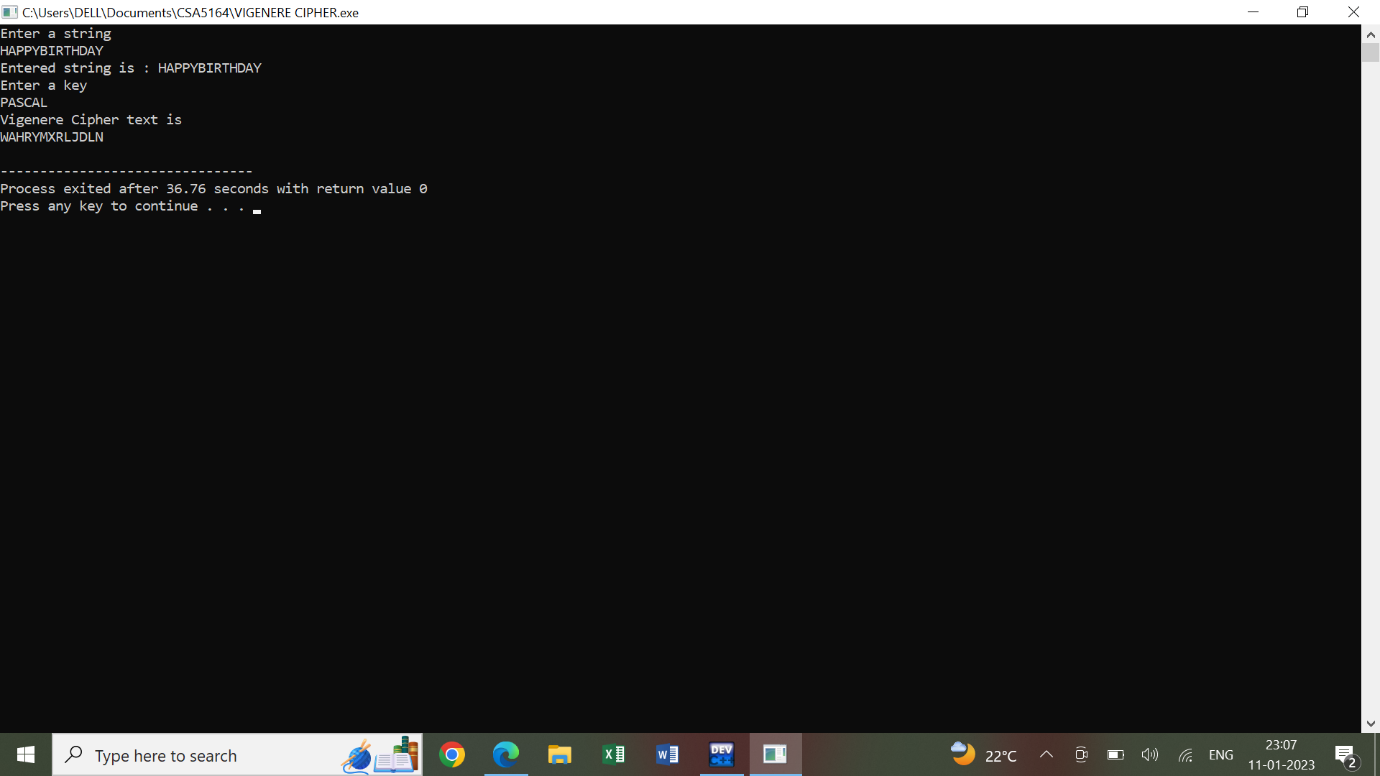
printf("%c",(numcipher[i]+'A'));

}

printf("\n");

}

**OUT PUT :**



**3.PLAYFAIR CIPHER :**

**PROGRAM :**

#include<stdio.h>

#include<string.h>

#include<ctype.h>

int removerepeated(int size,int a[]);

int insertelementat(int position,int a[],int size);

main()

{

int i,j,k,numstr[100],numcipher[100],numkey[100],lenkey,templen,tempkey[100],flag=-1,size,cipherkey[5][5],lennumstr,row1,row2,col1,col2;

char str[100],key[100];

printf("Enter a string\n");

gets(str);

for(i=0,j=0;i<strlen(str);i++)

{

if(str[i]!=' ')

{

str[j]=toupper(str[i]);

j++;

}

}

str[j]='\0';

printf("Entered String is %s\n",str);

size=strlen(str);

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

{

if(str[i]!=' ')

numstr[i]=str[i]-'A';

}

lennumstr=i;

printf("Enter the key (Non repeated elements if possible)\n");

gets(key);

for(i=0,j=0;i<strlen(key);i++)

{

if(key[i]!=' ')

{

key[j]=toupper(key[i]);

j++;

}

}

key[j]='\0';

printf("%s\n",key);

k=0;

for(i=0;i<strlen(key)+26;i++)

{

if(i<strlen(key))

{

if(key[i]=='J')

{

flag=8;

printf("%d",flag);

}

numkey[i]=key[i]-'A';

}

else

{

if(k!=9 && k!=flag)

{

numkey[i]=k;

}

k++;

}

}

templen=i;

lenkey=removerepeated(templen,numkey);

printf("Entered key converted according to Play Fair Cipher rule\n");

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

{

printf("%c",numkey[i]+'A');

}

printf("\n");

k=0;

for(i=0;i<5;i++)

{

for(j=0;j<5;j++)

{

cipherkey[i][j]=numkey[k];

k++;

}

}

printf("Arranged key\n");

for(i=0;i<5;i++)

{

for(j=0;j<5;j++)

{

printf("%c ",cipherkey[i][j]+'A');

}

printf("\n");

}

for(i=0;i<lennumstr;i+=2)

{

if(numstr[i]==numstr[i+1])

{

insertelementat(i+1,numstr,lennumstr);

lennumstr++;

}

}

if(lennumstr%2!=0)

{

insertelementat(lennumstr,numstr,lennumstr);

lennumstr++;

}

printf("Entered String/Message After Processing according to Play fair cipher rule\n");

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

{

printf("%c",numstr[i]+'A');

}

for(k=0;k<lennumstr;k+=2)

{

for(i=0;i<5;i++)

{

for(j=0;j<5;j++)

{

if(numstr[k]==cipherkey[i][j])

{

row1=i;

col1=j;

}

if(numstr[k+1]==cipherkey[i][j])

{

row2=i;

col2=j;

}

}

}

if(row1==row2)

{

col1=(col1-1)%5;

col2=(col2-1)%5;

if(col1<0)

{

col1=5+col1;

}

if(col2<0)

{

col2=5+col2;

}

numcipher[k]=cipherkey[row1][col1];

numcipher[k+1]=cipherkey[row2][col2];

}

if(col1==col2)

{

row1=(row1-1)%5;

row2=(row2-1)%5;

if(row1<0)

{

row1=5+row1;

}

if(row2<0)

{

row2=5+row2;

}

numcipher[k]=cipherkey[row1][col1];

numcipher[k+1]=cipherkey[row2][col2];

}

if(row1!=row2&&col1!=col2)

{

numcipher[k]=cipherkey[row1][col2];

numcipher[k+1]=cipherkey[row2][col1];

}

}

printf("\nCipher Text is\n");

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

{

if((numcipher[i]+'A')!='X')

printf("%c",numcipher[i]+'A');

}

printf("\n");

}

int removerepeated(int size,int a[])

{

int i,j,k;

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

{

for(j=i+1;j<size;)

{

if(a[i]==a[j])

{

for(k=j;k<size;k++)

{

a[k]=a[k+1];

}

size--;

}

else

{

j++;

}

}

}

return(size);

}

int insertelementat(int position,int a[],int size)

{

int i,insitem=23,temp[size+1];

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

{

if(i<position)

{

temp[i]=a[i];

}

if(i>position)

{

temp[i]=a[i-1];

}

if(i==position)

{

temp[i]=insitem;

}

}

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

{

a[i]=temp[i];

}

}

**4.RAILFENCE TECHNIQUE :**

**PROGRAM :**

#include<stdio.h>

#include<conio.h>

#include<string.h>

int main()

{

int i,j,k,l;

char a[20],c[20],d[20];

printf("\n\t\t RAIL FENCE TECHNIQUE");

printf("\n\nEnter the input string : ");

gets(a);

l=strlen(a);

for(i=0,j=0;i<l;i++)

{

if(i%2==0)

c[j++]=a[i];

}

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

{

if(i%2==1)

c[j++]=a[i];

}

c[j]='\0';

printf("\nCipher text after applying rail fence :");

printf("\n%s",c);

if(l%2==0)

k=l/2;

else

k=(l/2)+1;

for(i=0,j=0;i<k;i++)

{

d[j]=c[i];

j=j+2;

}

for(i=k,j=1;i<l;i++)

{

d[j]=c[i];

j=j+2;

}

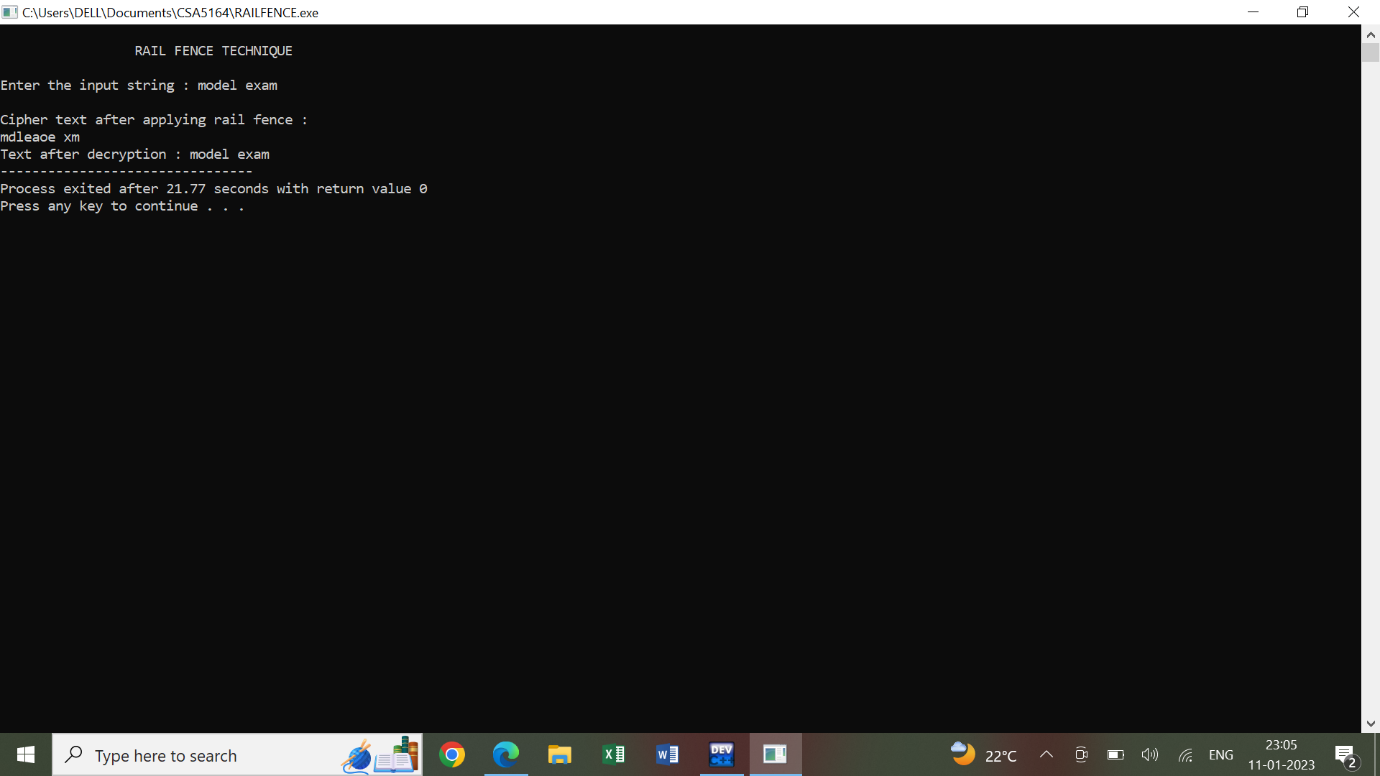
d[l]='\0';

printf("\nText after decryption : ");

printf("%s",d);

}

**OUT PUT :**

****

**5. HILL CIPHER**

**PROGRAM :**

#include<stdio.h>

#include<conio.h>

#include<string.h>

int main()

{

int i,j,k,l;

char a[20],c[20],d[20];

printf("\n\t\t RAIL FENCE TECHNIQUE");

printf("\n\nEnter the input string : ");

gets(a);

l=strlen(a);

for(i=0,j=0;i<l;i++)

{

if(i%2==0)

c[j++]=a[i];

}

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

{

if(i%2==1)

c[j++]=a[i];

}

c[j]='\0';

printf("\nCipher text after applying rail fence :");

printf("\n%s",c);

if(l%2==0)

k=l/2;

else

k=(l/2)+1;

for(i=0,j=0;i<k;i++)

{

d[j]=c[i];

j=j+2;

}

for(i=k,j=1;i<l;i++)

{

d[j]=c[i];

j=j+2;

}

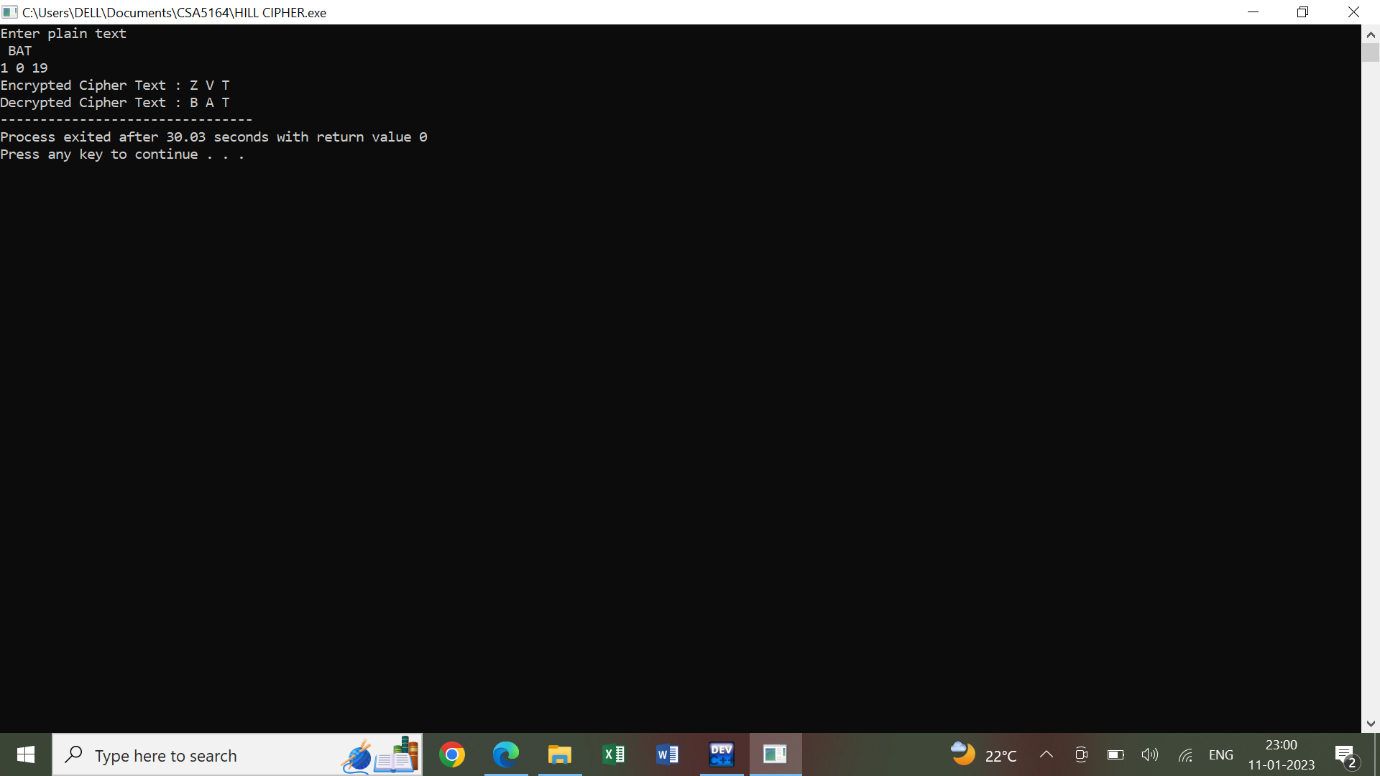
d[l]='\0';

printf("\nText after decryption : ");

printf("%s",d);

}

**OUT PUT :**

****

**6.RSA ALGORITM :**

**PROGRAM :**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<math.h>

#include<string.h>

long int

p,q,n,t,flag,e[100],d[100],temp[100],j,m[100],en[100],i;

char msg[100];

int prime(long int);

void ce();

long int cd(long int);

void encrypt();

void decrypt();

int main()

{

printf("\nENTER FIRST PRIME NUMBER\n");

scanf("%d",&p);

flag=prime(p);

if(flag==0)

{

printf("\nWRONG INPUT\n");

}

printf("\nENTER ANOTHER PRIME NUMBER\n");

scanf("%d",&q);

flag=prime(q);

if(flag==0||p==q)

{

printf("\nWRONG INPUT\n");

}

printf("\nENTER MESSAGE\n");

fflush(stdin);

scanf("%s",msg);

for(i=0;msg[i]!=NULL;i++)

m[i]=msg[i];

n=p\*q;

t=(p-1)\*(q-1);

ce();

printf("\nPOSSIBLE VALUES OF e AND d ARE\n");

for(i=0;i<j-1;i++)

printf("\n%ld\t%ld",e[i],d[i]);

encrypt();

decrypt();

}

int prime(long int pr)

{

int i;

j=sqrt(pr);

for(i=2;i<=j;i++)

{

if(pr%i==0)

return 0;

}

return 1;

}

void ce()

{

int k;

k=0;

for(i=2;i<t;i++)

{

if(t%i==0)

continue;

flag=prime(i);

if(flag==1&&i!=p&&i!=q)

{

e[k]=i;

flag=cd(e[k]);

if(flag>0)

{

d[k]=flag;

k++;

}

if(k==99)

break;

} } }

long int cd(long int x)

{

long int k=1;

while(1)

{

k=k+t;

if(k%x==0)

return(k/x);

} }

void encrypt() {

long int pt,ct,key=e[0],k,len;

i=0;

len=strlen(msg);

while(i!=len) {

pt=m[i];

pt=pt-96;

k=1;

for(j=0;j<key;j++)

{ k=k\*pt;

k=k%n;

}

temp[i]=k;

ct=k+96;

en[i]=ct;

i++;

}

en[i]=-1;

printf("\nTHE ENCRYPTED MESSAGE IS\n");

for(i=0;en[i]!=-1;i++)

printf("%c",en[i]);

}

void decrypt()

{

long int pt,ct,key=d[0],k;

i=0;

while(en[i]!=-1)

{

ct=temp[i];

k=1;

for(j=0;j<key;j++)

{

k=k\*ct;

k=k%n;

}

pt=k+96;

m[i]=pt;

i++;

}

m[i]=-1;

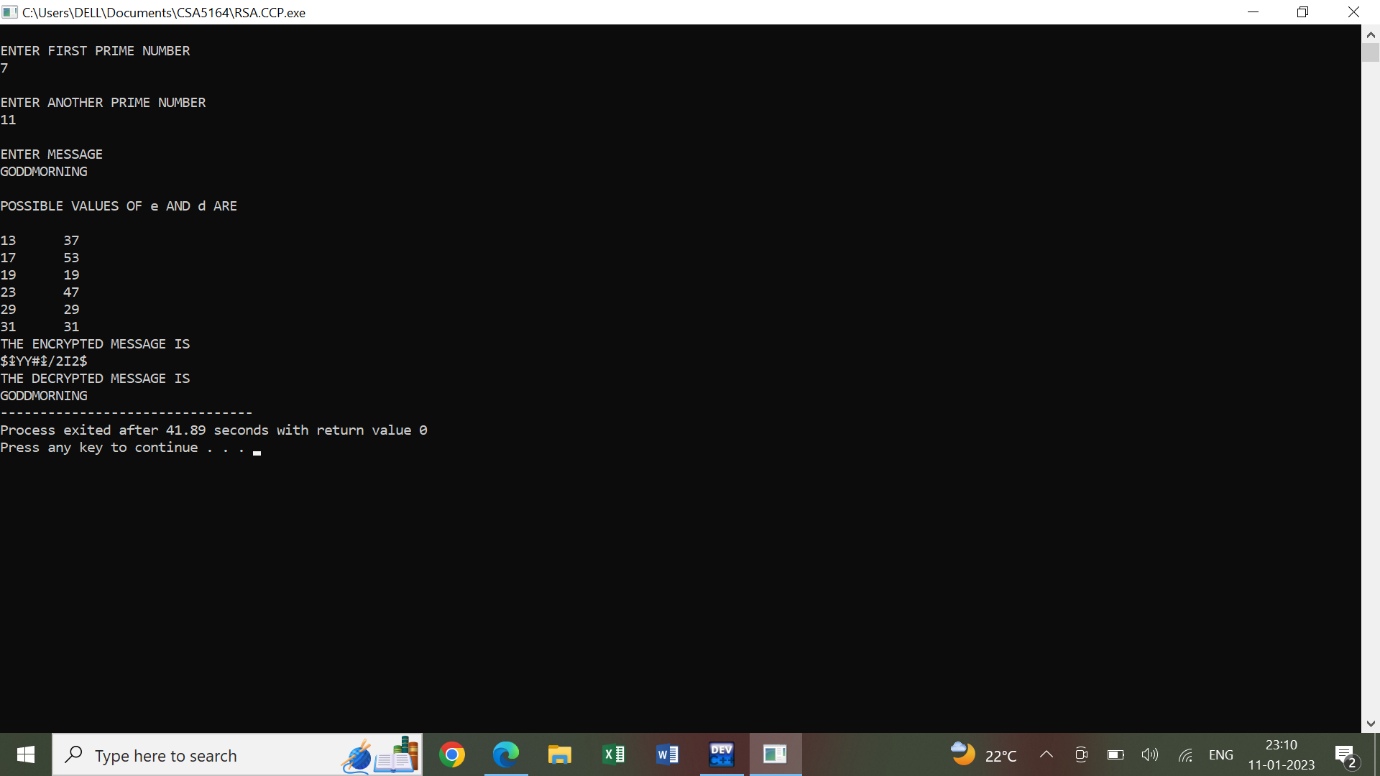
printf("\nTHE DECRYPTED MESSAGE IS\n");

for(i=0;m[i]!=-1;i++)

printf("%c",m[i]);

}

**OUT PUT :**



**7.DIFFIE HELLMAN :**

**PROGRAM :**

#include<stdio.h>

#include<conio.h>

long long int power(int a, int b, int mod)

{

long long int t;

if(b==1)

return a;

t=power(a,b/2,mod);

if(b%2==0)

return (t\*t)%mod;

else

return (((t\*t)%mod)\*a)%mod;

}

long int calculateKey(int a, int x, int n)

{

return power(a,x,n);

}

int main()

{

int n,g,x,a,y,b;

printf("Enter the value of n and g : ");

scanf("%d%d",&n,&g);

printf("Enter the value of x for the first person : ");

scanf("%d",&x);

a=power(g,x,n);

printf("Enter the value of y for the second person : ");

scanf("%d",&y);

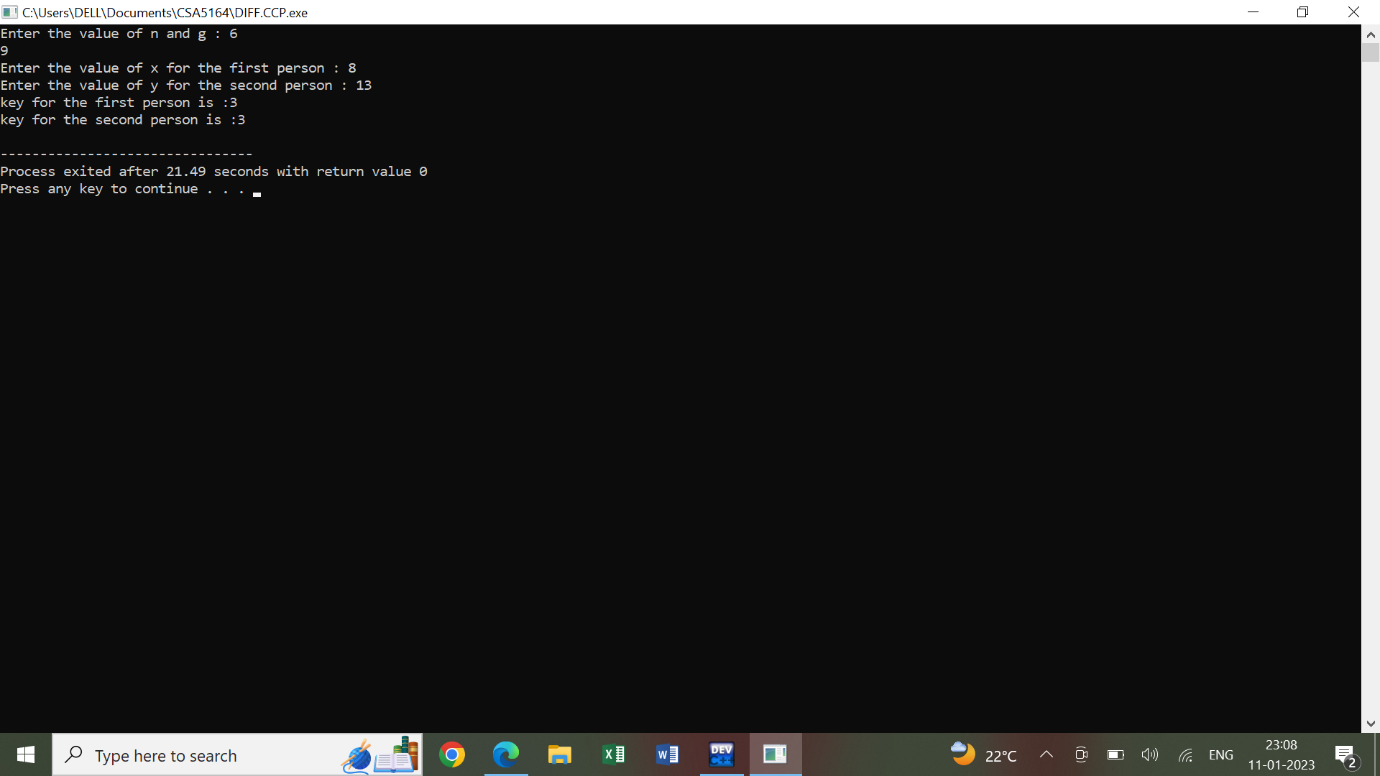
b=power(g,y,n);

printf("key for the first person is :%lld\n",power(b,x,n));

printf("key for the second person is :%lld\n",power(a,y,n));

}

**OUT PUT :**

****

**8.DSA ALGORITM :**

**PROGRAM :**

#include<stdio.h>

#include<conio.h>

#include<math.h>

long int ext\_eucledian(long int m,long int b)

{

int a1=1,a2=0,a3=m,b1=0,b2=1,b3=b,q,t1,t2,t3;

while(1)

{

if(b3==0)

{

return 0;

}

if(b3==1)

{

if(b2<0)

b2+=m;

return b2;

}

q=a3/b3;

t1=a1-(q\*b1);

t2=a2-(q\*b2);

t3=a3-(q\*b3);

a1=b1;

a2=b2;

a3=b3;

b1=t1;

b2=t2;

b3=t3;

}

}

long int power(long int a, long int j, long int c)

{

int f,i;

f=1;

for(i=1;i<=j;i++)

{

f=(f\*a)%c;

}

f=f%c;

return f;

}

int main()

{

long int p,q,g,x,hm,k,y,r,s,s1,w,u1,u2,v,v1,v2,v3;

printf("enter the value of p:");

scanf("%ld",&p);

printf("enter the value of q:");

scanf("%ld",&q);

printf("enter the value of g:");

scanf("%ld",&g);

printf("enter the value of x:");

scanf("%ld",&x);

printf("enter the value of hm:");

scanf("%ld",&hm);

printf("enter the value of k:");

scanf("%ld",&k);

y=power(g,x,p);

printf("\nvalue of y:%ld",y);

r=power(g,k,p);

r=r%q;

printf("\nvalue of r:%ld",r);

s=ext\_eucledian(q,k);

s1=(hm+(x\*r));

s=(s\*s1)%q;

printf("\nvalue of s:%ld",s);

w=ext\_eucledian(q,s);

printf("\nsignature (r,s):%ld %ld",r,s);

printf("\nvalue of w:%ld",w);

u1=(hm\*w)%q;

printf("\nvalue of u1:%ld",u1);

u2=(r\*w)%q;

printf("\nvalue of u2:%ld",u2);

v=power(g,u1,p);

v1=power(y,u2,p);

v2=(v\*v1)%p;

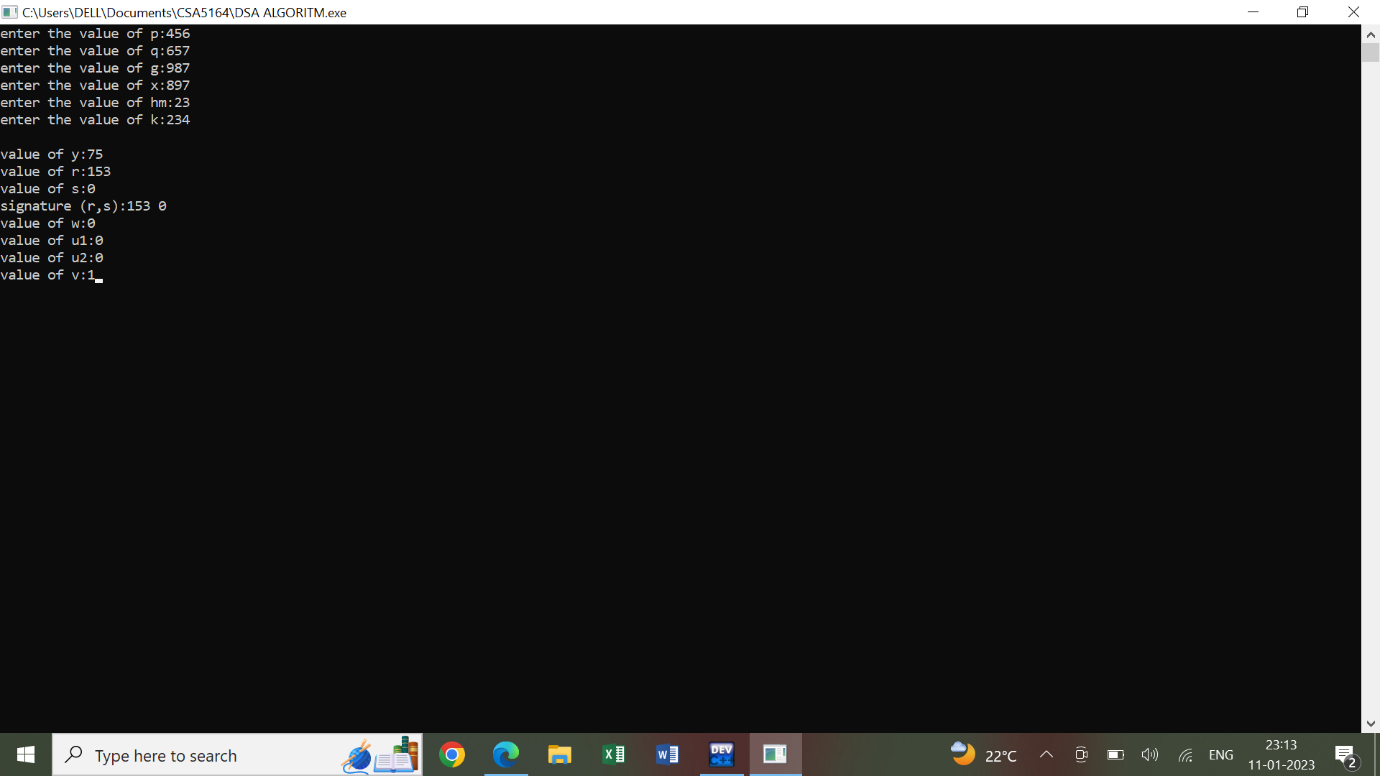
v3=v2%q;

printf("\nvalue of v:%ld",v3);

getch();

}

**OUT PUT :**

****

**9.MD5 ALGORITHM :**

**PROGRAM :**

#include <iostream>

#include <string>

#include <cstring>

using namespace std;

#define MD5\_INPUT\_LENGTH 64

typedef unsigned char uchar\_8;

typedef unsigned int uint\_32;

typedef unsigned long uint\_64;

static const uint\_32 T[64] = {

0xd76aa478, 0xe8c7b756, 0x242070db, 0xc1bdceee,

0xf57c0faf, 0x4787c62a, 0xa8304613, 0xfd469501,

0x698098d8, 0x8b44f7af, 0xffff5bb1, 0x895cd7be,

0x6b901122, 0xfd987193, 0xa679438e, 0x49b40821,

0xf61e2562, 0xc040b340, 0x265e5a51, 0xe9b6c7aa,

0xd62f105d, 0x02441453, 0xd8a1e681, 0xe7d3fbc8,

0x21e1cde6, 0xc33707d6, 0xf4d50d87, 0x455a14ed,

0xa9e3e905, 0xfcefa3f8, 0x676f02d9, 0x8d2a4c8a,

0xfffa3942, 0x8771f681, 0x6d9d6122, 0xfde5380c,

0xa4beea44, 0x4bdecfa9, 0xf6bb4b60, 0xbebfbc70,

0x289b7ec6, 0xeaa127fa, 0xd4ef3085, 0x04881d05,

0xd9d4d039, 0xe6db99e5, 0x1fa27cf8, 0xc4ac5665,

0xf4292244, 0x432aff97, 0xab9423a7, 0xfc93a039,

0x655b59c3, 0x8f0ccc92, 0xffeff47d, 0x85845dd1,

0x6fa87e4f, 0xfe2ce6e0, 0xa3014314, 0x4e0811a1,

0xf7537e82, 0xbd3af235, 0x2ad7d2bb, 0xeb86d391

};

static const uint\_32 A0 = 0x67452301;

static const uint\_32 B0 = 0xefcdab89;

static const uint\_32 C0 = 0x98badcfe;

static const uint\_32 D0 = 0x10325476;

inline uint\_32 F(uint\_32 x, uint\_32 y, uint\_32 z) {

return ((x & y) | (~x & z));

}

inline uint\_32 G(uint\_32 x, uint\_32 y, uint\_32 z) {

return ((x & y) | (y & ~z));

}

inline uint\_32 H(uint\_32 x, uint\_32 y, uint\_32 z) {

return (x ^ y ^ z);

}

inline uint\_32 I(uint\_32 x, uint\_32 y, uint\_32 z) {

return (y ^ (x | ~z));

}

inline uint\_32 shift(uint\_32 x, uint\_32 s) {

return ((x << s) | (x >> (32 - s)));

}

inline void FF(uint\_32 &a, uint\_32 b, uint\_32 c, uint\_32 d, uint\_32 x, uint\_32 s, uint\_32 t) {

a += F(b, c, d) + x + t;

a = shift(a, s) + b;

}

inline void GG(uint\_32 &a, uint\_32 b, uint\_32 c, uint\_32 d, uint\_32 x, uint\_32 s, uint\_32 t) {

a += G(b, c, d) + x + t;

a = shift(a, s) + b;

}

inline void HH(uint\_32 &a, uint\_32 b, uint\_32 c, uint\_32 d, uint\_32 x, uint\_32 s, uint\_32 t) {

a += H(b, c, d) + x + t;

a = shift(a, s) + b;

}

inline void II(uint\_32 &a, uint\_32 b, uint\_32 c, uint\_32 d, uint\_32 x, uint\_32 s, uint\_32 t) {

a += I(b, c, d) + x + t;

a = shift(a, s) + b;

}

uint\_32 \*md5Pad(char \*charBuf, uint\_64 len) {

// Number of characters in buffer \* 8 mod 64

uint\_64 newLen;

for (newLen = len \* 8 + 1; newLen % 512 != 448; newLen++);

newLen /= 8;

uint\_32 \*buf = new uint\_32[newLen + 64];

memset(buf, 0, newLen + 64);

memcpy(buf, charBuf, len);

buf[len] = 0x80;

uint\_32 bitsLen = len \* 8;

memcpy(buf + newLen, &bitsLen, 4);

return buf;

}

void MD5(uint\_32 \*outBuf, uint\_32 \*inBuf) {

uint\_32 a, b, c, d;

const uint\_32 a0 = 0x67452301;

const uint\_32 b0 = 0xEFCDAB89;

const uint\_32 c0 = 0x98BADCFE;

const uint\_32 d0 = 0x10325476;

a = a0;

b = b0;

c = c0;

d = d0;

static const uchar\_8 S11 = 7;

static const uchar\_8 S12 = 12;

static const uchar\_8 S13 = 17;

static const uchar\_8 S14 = 22;

FF ( a, b, c, d, inBuf[ 0], S11, T[ 0]); /\* 1 \*/

FF ( d, a, b, c, inBuf[ 1], S12, T[ 1]); /\* 2 \*/

FF ( c, d, a, b, inBuf[ 2], S13, T[ 2]); /\* 3 \*/

FF ( b, c, d, a, inBuf[ 3], S14, T[ 3]); /\* 4 \*/

FF ( a, b, c, d, inBuf[ 4], S11, T[ 4]); /\* 5 \*/

FF ( d, a, b, c, inBuf[ 5], S12, T[ 5]); /\* 6 \*/

FF ( c, d, a, b, inBuf[ 6], S13, T[ 6]); /\* 7 \*/

FF ( b, c, d, a, inBuf[ 7], S14, T[ 7]); /\* 8 \*/

FF ( a, b, c, d, inBuf[ 8], S11, T[ 8]); /\* 9 \*/

FF ( d, a, b, c, inBuf[ 9], S12, T[ 9]); /\* 10 \*/

FF ( c, d, a, b, inBuf[10], S13, T[10]); /\* 11 \*/

FF ( b, c, d, a, inBuf[11], S14, T[1]); /\* 12 \*/

FF ( a, b, c, d, inBuf[12], S11, T[12]); /\* 13 \*/

FF ( d, a, b, c, inBuf[13], S12, T[13]); /\* 14 \*/

FF ( c, d, a, b, inBuf[14], S13, T[14]); /\* 15 \*/

FF ( b, c, d, a, inBuf[15], S14, T[15]); /\* 16 \*/

// Shift amounts 2nd round

static const uchar\_8 S21 = 5;

static const uchar\_8 S22 = 9;

static const uchar\_8 S23 = 14;

static const uchar\_8 S24 = 20;

GG ( a, b, c, d, inBuf[ 1], S21, T[16]); /\* 17 \*/

GG ( d, a, b, c, inBuf[ 6], S22, T[17]); /\* 18 \*/

GG ( c, d, a, b, inBuf[11], S23, T[18]); /\* 19 \*/

GG ( b, c, d, a, inBuf[ 0], S24, T[19]); /\* 20 \*/

GG ( a, b, c, d, inBuf[ 5], S21, T[20]); /\* 21 \*/

GG ( d, a, b, c, inBuf[10], S22, T[21]); /\* 22 \*/

GG ( c, d, a, b, inBuf[15], S23, T[22]); /\* 23 \*/

GG ( b, c, d, a, inBuf[ 4], S24, T[23]); /\* 24 \*/

GG ( a, b, c, d, inBuf[ 9], S21, T[24]); /\* 25 \*/

GG ( d, a, b, c, inBuf[14], S22, T[25]); /\* 26 \*/

GG ( c, d, a, b, inBuf[ 3], S23, T[26]); /\* 27 \*/

GG ( b, c, d, a, inBuf[ 8], S24, T[27]); /\* 28 \*/

GG ( a, b, c, d, inBuf[13], S21, T[28]); /\* 29 \*/

GG ( d, a, b, c, inBuf[ 2], S22, T[29]); /\* 30 \*/

GG ( c, d, a, b, inBuf[ 7], S23, T[30]); /\* 31 \*/

GG ( b, c, d, a, inBuf[12], S24, T[31]); /\* 32 \*/

// Shift amounts 3rd round

static const uchar\_8 S31 = 4;

static const uchar\_8 S32 = 11;

static const uchar\_8 S33 = 16;

static const uchar\_8 S34 = 23;

HH ( a, b, c, d, inBuf[ 5], S31, T[32]); /\* 33 \*/

HH ( d, a, b, c, inBuf[ 8], S32, T[33]); /\* 34 \*/

HH ( c, d, a, b, inBuf[11], S33, T[34]); /\* 35 \*/

HH ( b, c, d, a, inBuf[14], S34, T[35]); /\* 36 \*/

HH ( a, b, c, d, inBuf[ 1], S31, T[36]); /\* 37 \*/

HH ( d, a, b, c, inBuf[ 4], S32, T[37]); /\* 38 \*/

HH ( c, d, a, b, inBuf[ 7], S33, T[38]); /\* 39 \*/

HH ( b, c, d, a, inBuf[10], S34, T[39]); /\* 40 \*/

HH ( a, b, c, d, inBuf[13], S31, T[40]); /\* 41 \*/

HH ( d, a, b, c, inBuf[ 0], S32, T[41]); /\* 42 \*/

HH ( c, d, a, b, inBuf[ 3], S33, T[42]); /\* 43 \*/

HH ( b, c, d, a, inBuf[ 6], S34, T[43]); /\* 44 \*/

HH ( a, b, c, d, inBuf[ 9], S31, T[44]); /\* 45 \*/

HH ( d, a, b, c, inBuf[12], S32, T[45]); /\* 46 \*/

HH ( c, d, a, b, inBuf[15], S33, T[46]); /\* 47 \*/

HH ( b, c, d, a, inBuf[ 2], S34, T[47]); /\* 48 \*/

// Shift amounts 4th round

static const uchar\_8 S41 = 6;

static const uchar\_8 S42 = 10;

static const uchar\_8 S43 = 15;

static const uchar\_8 S44 = 21;

II ( a, b, c, d, inBuf[ 0], S41, T[48]); /\* 49 \*/

II ( d, a, b, c, inBuf[ 7], S42, T[49]); /\* 50 \*/

II ( c, d, a, b, inBuf[14], S43, T[50]); /\* 51 \*/

II ( b, c, d, a, inBuf[ 5], S44, T[51]); /\* 52 \*/

II ( a, b, c, d, inBuf[12], S41, T[52]); /\* 53 \*/

II ( d, a, b, c, inBuf[ 3], S42, T[53]); /\* 54 \*/

II ( c, d, a, b, inBuf[10], S43, T[54]); /\* 55 \*/

II ( b, c, d, a, inBuf[ 1], S44, T[55]); /\* 56 \*/

II ( a, b, c, d, inBuf[ 8], S41, T[56]); /\* 57 \*/

II ( d, a, b, c, inBuf[15], S42, T[57]); /\* 58 \*/

II ( c, d, a, b, inBuf[ 6], S43, T[58]); /\* 59 \*/

II ( b, c, d, a, inBuf[13], S44, T[59]); /\* 60 \*/

II ( a, b, c, d, inBuf[ 4], S41, T[60]); /\* 61 \*/

II ( d, a, b, c, inBuf[11], S42, T[61]); /\* 62 \*/

II ( c, d, a, b, inBuf[ 2], S43, T[62]); /\* 63 \*/

II ( b, c, d, a, inBuf[ 9], S44, T[63]); /\* 64 \*/

a += a0;

b += b0;

c += c0;

d += d0;

outBuf[0] = a;

outBuf[1] = b;

outBuf[2] = c;

outBuf[3] = d;

return;

}

int main() {

char \*srcStr = " ";

uint\_32 \*padded = md5Pad(srcStr, strlen(srcStr));

uint\_32 result[4] = {0};

MD5(result, padded);

uchar\_8 digestChars[16] = {0};

memcpy(digestChars, result, 16);

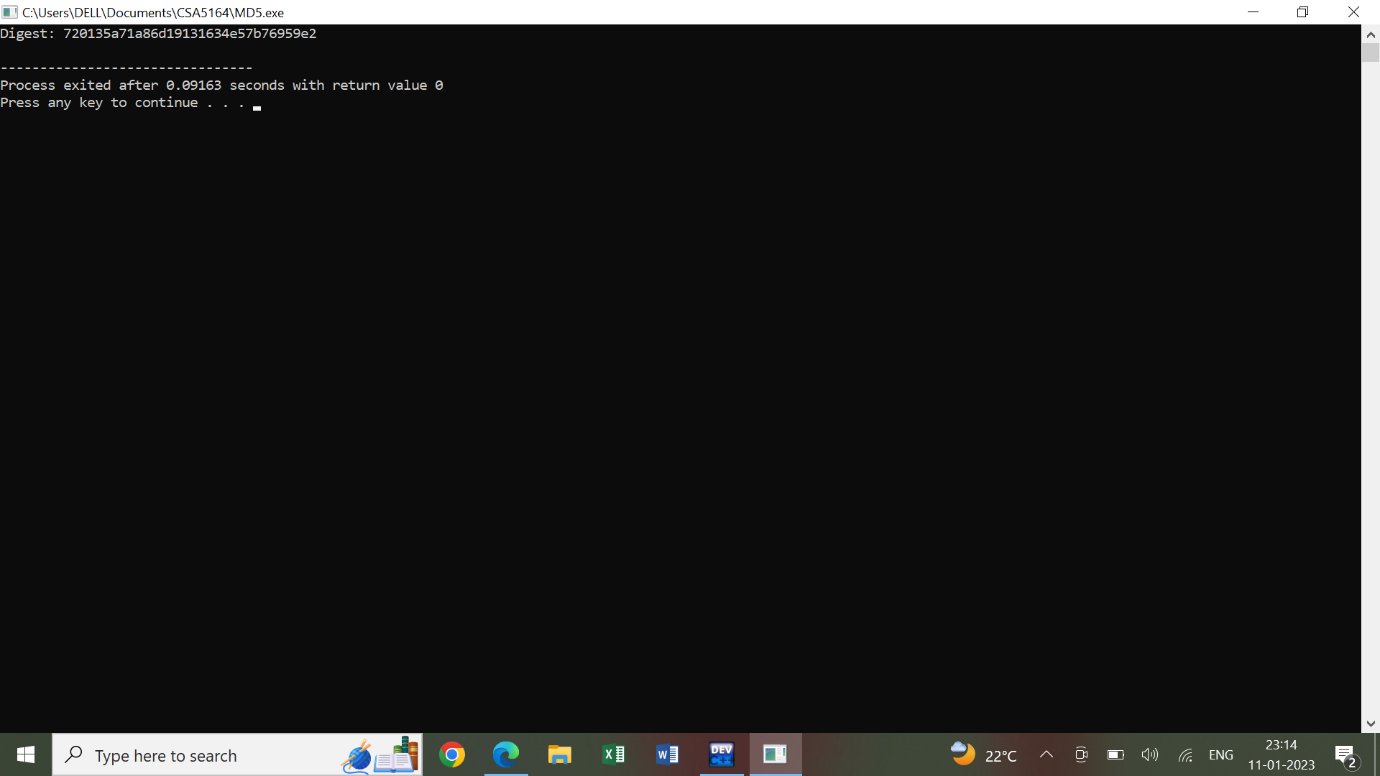
printf("Digest: %2.2x%2.2x%2.2x%2.2x\n", result[0], result[1], result[2], result[3]);

delete[] padded;

return 0;

}

**OUT PUT :**

****

**10.DES ALGORITHM :**

**PROGRAM :**

#include <bits/stdc++.h>

using namespace std;

int permChoiceOne[] = {

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 permChoiceTwo[] = {

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 };

int leftShiftTable[] = {1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};

string rotateSubKey(string s , int rot)

{

return s.substr(rot, s.length()-rot) + s.substr(0, rot) ;

}

string firstPermute(string input)

{

string res = "" ;

for(int i=0 ; i<56 ; i++)

{

res += input[permChoiceOne[i]-1];

}

return res ;

}

string secondPermute(string input)

{

string res = "" ;

for(int i=0 ; i<48 ; i++)

{

res += input[permChoiceTwo[i]-1];

}

return res ;

}

void genKeys(string left, string right)

{

ofstream fout ;

fout.open("keygen.txt");

for (int i=0; i<16; i++)

{

left = rotateSubKey(left , leftShiftTable[i]);

right = rotateSubKey(right, leftShiftTable[i]);

string key = secondPermute(left+right);

cout << "key " << i+1 << " \t: " << key << endl;

fout << key << endl;

}

}

int main()

{

unsigned long long hexkey;

cout << "\nEnter 64-bit key in hexadecimal(16-digits) : " ;

cin >> hex >> hexkey;

string key = bitset<64>(hexkey).to\_string();

cout << "Binary key (k) \t: " << key << endl;

key = firstPermute(key) ;

cout << "PC-1 key (k+) \t: " << key << endl;

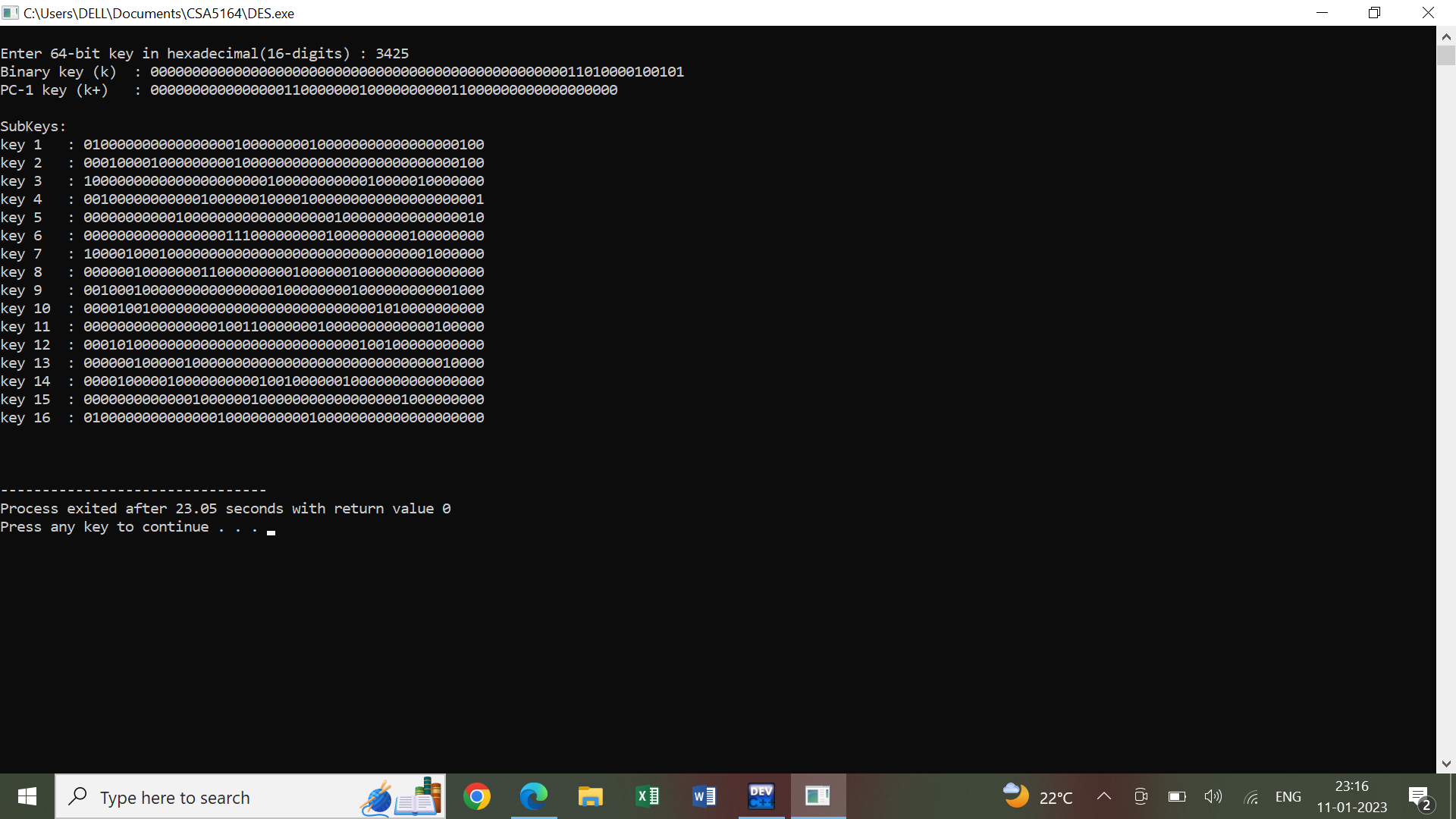
cout << "\nSubKeys: " << endl;

genKeys(key.substr(0,28) , key.substr(28,28));

cout<<endl<<endl ;

}

**OUT PUT :**

****