1.)

a.) Shift Cipher:

Encryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

const int n=26;

void encrypt(char p[100],int k)

{

int i=0,j=0;

k=k%n;

char c[100];

for(i=0;p[i]!='\0';i++){

if(p[i]==' ')

continue;

if(islower(p[i]))

c[j++]=(char)((int)(p[i]+k-97)%n+65);

else if(isupper(p[i]))

c[j++]=(char)((int)(p[i]+k-65)%n+97);

}

c[j]='\0';

printf("%s",c);

return;

}

int main()

{

char p[100];

int k;

printf("--------------------------------------------------AT ALICE'S END-------------------------------------------\n");

printf("Enter the plain-text to be transmitted: ");

scanf("%[^\n]s",p);

printf("Enter the shared secret key: ");

scanf("%d",&k);

printf("The cipher text to be transmitted is: ");

encrypt(p,k);

return 0;

}

Decryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

const int n=26;

int additive\_inverse(int k){

return ((n-(k%n))%n);

}

void decrypt(char c[100],int k)

{

int i=0;

char p[100];

for(i=0;c[i]!='\0';i++){

if(islower(c[i]))

p[i]=(char)(((int)(c[i]-97)+additive\_inverse(k))%n+65);

else if(isupper(c[i]))

p[i]=(char)(((int)(c[i]-65)+additive\_inverse(k))%n+97);

else

p[i]=c[i];

}

p[i]='\0';

printf("%s",p);

return;

}

int main()

{

char c[100];

int k;

printf("--------------------------------------------------AT BOB'S END---------------------------------------------\n");

printf("Enter the cipher text to be received: ");

scanf("%[^\n]s",c);

printf("Enter the shared secret key: ");

scanf("%d",&k);

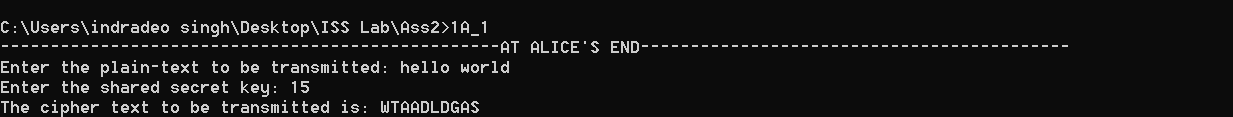
printf("The plain text decrypted is: ");

decrypt(c,k);

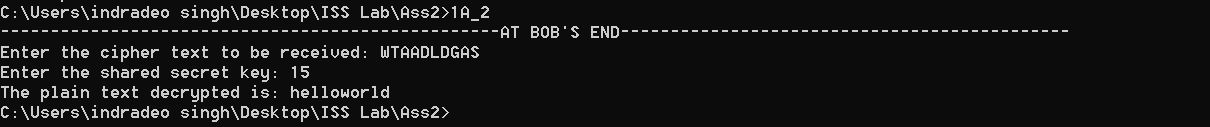
return 0;

}

Encryption Output



Decryption Output



b.) Multiplicative Cipher

Encryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

const int n=26;

int multiplicative\_inverse(int k){

int i=0;

for(i=0;i<n;i++){

if((i\*k)%n==1){

return i;

}

}

return -1;

}

void encrypt(char p[100],int k)

{

int i=0,j=0;

k=k%n;

char c[100];

for(i=0;p[i]!='\0';i++){

if(p[i]==' ')

continue;

if(islower(p[i]))

c[j++]=(char)((int)((p[i]-97)\*k)%n+65);

else if(isupper(p[i]))

c[j++]=(char)((int)((p[i]-65)\*k)%n+97);

}

c[j]='\0';

printf("%s",c);

return;

}

int main()

{

char p[100];

int k;

printf("--------------------------------------------------AT ALICE'S END-------------------------------------------\n");

printf("Enter the plain-text to be transmitted: ");

scanf("%[^\n]s",p);

printf("Enter the shared secret key: ");

scanf("%d",&k);

if(multiplicative\_inverse(k)==-1)

printf("Warning!!! Please enter appropriate key whose multiplicative inverse is existent!!!!");

else{

printf("The cipher text to be transmitted is: ");

encrypt(p,k);

}

return 0;

}

Decryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

const int n=26;

int multiplicative\_inverse(int k){

int i=0;

for(i=0;i<n;i++){

if((i\*k)%n==1){

return i;

}

}

return -1;

}

void decrypt(char c[100],int k)

{

int i=0;

char p[100];

for(i=0;c[i]!='\0';i++){

if(islower(c[i]))

p[i]=(char)(((int)(c[i]-97)\*multiplicative\_inverse(k))%n+65);

else if(isupper(c[i]))

p[i]=(char)(((int)(c[i]-65)\*multiplicative\_inverse(k))%n+97);

else

p[i]=c[i];

}

p[i]='\0';

printf("%s",p);

return;

}

int main()

{

char c[100];

int k;

printf("--------------------------------------------------AT BOB'S END---------------------------------------------\n");

printf("Enter the cipher text to be received: ");

scanf("%[^\n]s",c);

printf("Enter the shared secret key: ");

scanf("%d",&k);

if(multiplicative\_inverse(k)==-1)

printf("Warning!!! Please enter appropriate key whose multiplicative inverse is existent!!!!");

else{

printf("The plain text decrypted is: ");

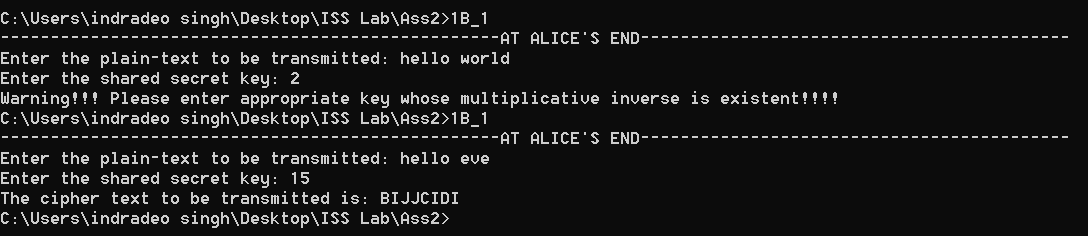
decrypt(c,k);

}

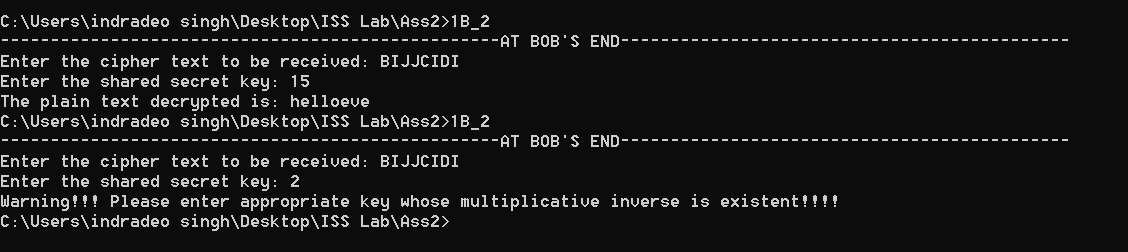
return 0;

}

Encryption Output



Decryption Output



c.) Affine Cipher

Encryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

const int n=26;

int multiplicative\_inverse(int k){

int i=0;

for(i=0;i<n;i++){

if((i\*k)%n==1){

return i;

}

}

return -1;

}

void encrypt(char p[100],int k1,int k2)

{

int i=0,j=0;

k2=k2%n;

k1=k1%n;

char c[100];

for(i=0;p[i]!='\0';i++){

if(p[i]==' ')

continue;

if(islower(p[i]))

c[j++]=(char)((int)((p[i]-97)\*k1+k2)%n+65);

else if(isupper(p[i]))

c[j++]=(char)((int)((p[i]-65)\*k1+k2)%n+97);

}

c[j]='\0';

printf("%s",c);

return;

}

int main()

{

char p[100];

int k1,k2;

printf("--------------------------------------------------AT ALICE'S END-------------------------------------------\n");

printf("Enter the plain-text to be transmitted: ");

scanf("%[^\n]s",p);

printf("Enter the shared secret key (k1): ");

scanf("%d",&k1);

printf("Enter the shared secret key (k2): ");

scanf("%d",&k2);

if(multiplicative\_inverse(k1)==-1)

printf("Warning!!! Please enter appropriate key k1 whose multiplicative inverse is existent!!!!");

else{

printf("The cipher text to be transmitted is: ");

encrypt(p,k1,k2);

}

return 0;

}

Decryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

const int n=26;

int additive\_inverse(int k){

return ((n-(k%n))%n);

}

int multiplicative\_inverse(int k){

int i=0;

for(i=0;i<n;i++){

if((i\*k)%n==1){

return i;

}

}

return -1;

}

void decrypt(char c[100],int k1,int k2)

{

int i=0;

char p[100];

for(i=0;c[i]!='\0';i++){

if(islower(c[i]))

p[i]=(char)((int)((((c[i]-97)+additive\_inverse(k2))\*multiplicative\_inverse(k1))%n+65));

else if(isupper(c[i]))

p[i]=(char)((int)((((c[i]-65)+additive\_inverse(k2))\*multiplicative\_inverse(k1))%n+97));

else

p[i]=c[i];

}

p[i]='\0';

printf("%s",p);

return;

}

int main()

{

char c[100];

int k1,k2;

printf("--------------------------------------------------AT BOB'S END---------------------------------------------\n");

printf("Enter the cipher text to be received: ");

scanf("%[^\n]s",c);

printf("Enter the shared secret key (k1): ");

scanf("%d",&k1);

printf("Enter the shared secret key (k2): ");

scanf("%d",&k2);

if(multiplicative\_inverse(k1)==-1)

printf("Warning!!! Please enter appropriate key whose multiplicative inverse is existent!!!!");

else{

printf("The plain text decrypted is: ");

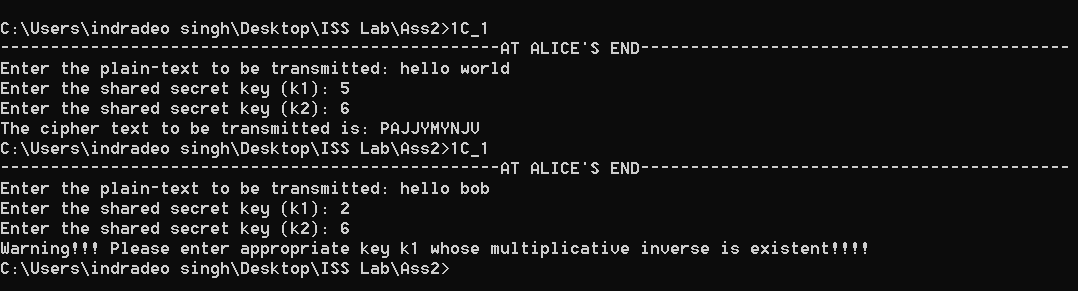
decrypt(c,k1,k2);

}

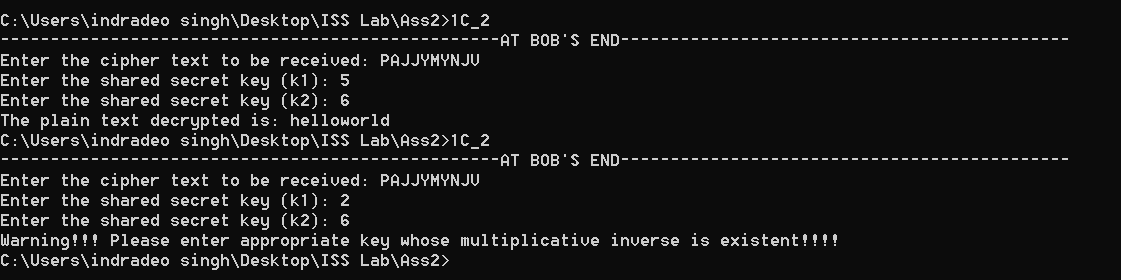
return 0;

}

Encryption Output



Decryption Output



d.) Playfair Cipher

Encryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

#include<time.h>

const int n=26;

char key[5][5]={{'L','G','D','B','A'},{'Q','M','H','E','C'},{'U','R','N','I','F'},{'X','V','S','O','K'},{'Z','Y','W','T','P'}};

char findBogusCharacter(char currChar)

{

char ch;

int curr=(int)(currChar-'a');

while(curr==(int)(currChar-'a'))

curr=rand()%26;

return (char)(curr+97);

}

void encrypt(char p[100])

{

// srand(time(0));

int i=0,j=0,k=0,kr1,kc1,kr2,kc2,row,col,flag=0;

char c[100];

char modifiedP[200];

for(i=0;p[i]!='\0';i++)

{

if(p[i]==' ')

continue;

if(i==0)

modifiedP[j++]=((p[i]=='j')?'i':p[i]);

else if((p[i]==p[i-1])||(p[i]=='i'&&p[i-1]=='j')||(p[i]=='j'&&p[i-1]=='i'))

{

char ch='x';

modifiedP[j++]=ch;

modifiedP[j++]=((p[i]=='j')?'i':p[i]);

}

else

modifiedP[j++]=((p[i]=='j')?'i':p[i]);

}

if(j%2==1)

{

char ch='x';

modifiedP[j++]=ch;

}

modifiedP[j]='\0';

printf("The modified plain text to be transmitted is: ");

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

j=0;

for(i=0;modifiedP[i]!='\0';i++)

{

flag=0;

if(i%2==1)

{

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

{

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

{

if(key[row][col]==modifiedP[i]-'a'+'A')

{

kr2=row;

kc2=col;

flag=1;

break;

}

}

if(flag)

break;

}

if(kr1==kr2)

{

c[j++]=key[kr1][(kc1+1)%5];

c[j++]=key[kr2][(kc2+1)%5];

}

else if(kc1==kc2)

{

c[j++]=key[(kr1+1)%5][kc1];

c[j++]=key[(kr2+1)%5][kc2];

}

else

{

c[j++]=key[kr1][kc2];

c[j++]=key[kr2][kc1];

}

}

else

{

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

{

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

{

if(key[row][col]==modifiedP[i]-'a'+'A')

{

kr1=row;

kc1=col;

flag=1;

break;

}

}

if(flag)

break;

}

}

}

c[j]='\0';

printf("The cipher text to be transmitted is: ");

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

return;

}

int main()

{

char p[100];

int k;

printf("--------------------------------------------------AT ALICE'S END-------------------------------------------\n");

printf("Enter the plain-text to be transmitted ([a,z]-{x}): ");

scanf("%[^\n]s",p);

encrypt(p);

return 0;

}

Decryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

const int n=26;

char key[5][5]={{'L','G','D','B','A'},{'Q','M','H','E','C'},{'U','R','N','I','F'},{'X','V','S','O','K'},{'Z','Y','W','T','P'}};

void decrypt(char c[100])

{

int i=0,j=0,k=0,kr1,kc1,kr2,kc2,row,col,flag=0;

char p[100];

for(i=0;c[i]!='\0';i++)

{

flag=0;

if(i%2==1)

{

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

{

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

{

if(key[row][col]==c[i])

{

kr2=row;

kc2=col;

flag=1;

break;

}

}

if(flag)

break;

}

if(kr1==kr2)

{

p[j++]=key[kr1][(kc1-1+5)%5]-'A'+'a';

p[j++]=key[kr2][(kc2-1+5)%5]-'A'+'a';

}

else if(kc1==kc2)

{

p[j++]=key[(kr1-1+5)%5][kc1]-'A'+'a';

p[j++]=key[(kr2-1+5)%5][kc2]-'A'+'a';

}

else

{

p[j++]=key[kr1][kc2]-'A'+'a';

p[j++]=key[kr2][kc1]-'A'+'a';

}

}

else

{

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

{

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

{

if(key[row][col]==c[i])

{

kr1=row;

kc1=col;

flag=1;

break;

}

}

if(flag)

break;

}

}

}

p[j]='\0';

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

return;

}

int main()

{

char c[100];

printf("--------------------------------------------------AT BOB'S END---------------------------------------------\n");

printf("Enter the cipher text to be received: ");

scanf("%[^\n]s",c);

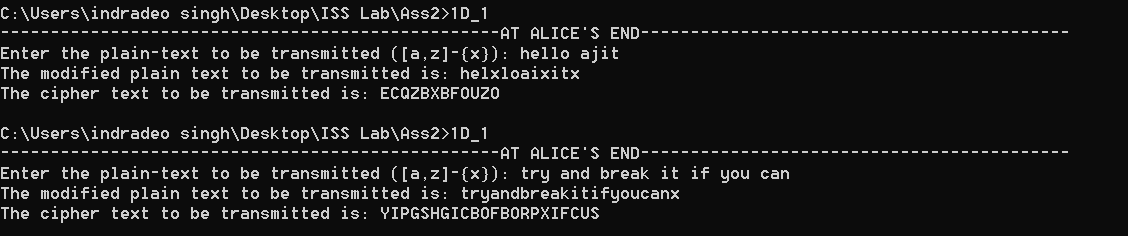
printf("The plain text decrypted is: ");

decrypt(c);

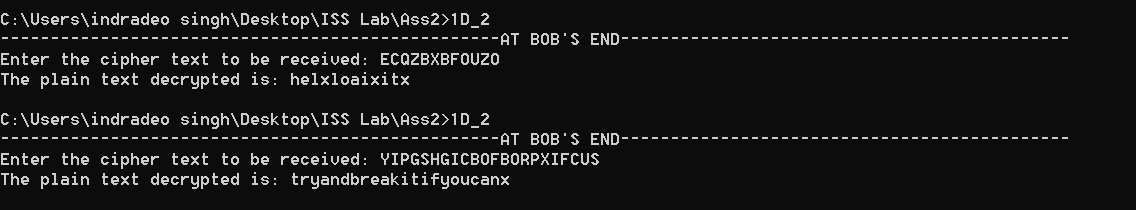
return 0;

}

Encryption Output



Decryption Output



e.) Hill Cipher

Encryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

#include<time.h>

const int n=26;

int N,size=0;

int multiplicative\_inverse(int k){

int i=0;

for(i=0;i<n;i++){

if((i\*k)%n==1){

return i;

}

}

return -1;

}

int additive\_inverse(int k){

return ((n+k)%n);

}

void fillCofactor(int mat[N][N],int temp[N][N],int r,int c){

int i=0,j=0,row=0,col=0;

for(row=0;row<N;row++){

for(col=0;col<N;col++){

if(row!=r&&col!=c){

temp[i][j++]=mat[row][col];

if(j==N-1){

j=0;

i++;

}

}

}

}

}

int findDeterminant(int mat[N][N],int dim){

int res=0;

if(dim==1)

return mat[0][0];

int temp[N][N],sign=1,col=0;

for(col=0;col<dim;col++){

fillCofactor(mat,temp,0,col);

res+=sign\*mat[0][col]\*findDeterminant(temp,dim-1);

// printf("%d\n",res);

sign=-sign;

}

return res;

}

void fillAdjoint(int mat[N][N],int adj[N][N]){

if(!N){

adj[0][0]=1;

return;

}

int sign=1,temp[N][N],i,j;

for(i=0;i<N;i++){

for(j=0;j<N;j++){

fillCofactor(mat,temp,i,j);

sign=(((i+j)%2)?-1:1);

adj[j][i]=sign\*(findDeterminant(temp,N-1));

}

}

}

int findInverse(int mat[N][N],int inv[N][N]){

int det=findDeterminant(mat,N);

printf("The determinant of the key matrix is: %d\n",det);

int detInv=multiplicative\_inverse(det);

int temp;

if(det==0)

return 0;

if(detInv==-1)

return -1;

int adj[N][N],i,j;

fillAdjoint(mat,adj);

for(i=0;i<N;i++){

for(j=0;j<N;j++){

temp=((adj[i][j]%n)\*(detInv%n))%n;

if(temp<0)

inv[i][j]=additive\_inverse(temp);

else

inv[i][j]=temp;

// printf("%d\n",temp );

}

}

return 1;

}

void preprocess(char p[500])

{

int i=0,pf=0,j=0;

char modifiedP[500];

for(i=0;p[i]!='\0';i++)

{

if(p[i]!=' ')

modifiedP[j++]=p[i];

}

modifiedP[j]='\0';

for(i=0;modifiedP[i]!='\0';i++)

p[i]=modifiedP[i];

p[i]='\0';

size=i;

if(size%N!=0)

{

printf("Adding bogus character(s) to make the matrix multiplication compatible.....\n");

while(size%N!=0)

p[i++]='x',size++;

p[i]='\0';

printf("The modified plain text is: %s\n",p);

}

else

{

printf("No need to add bogus character(s)!!!!\n");

}

return;

}

void encrypt(char p[500],int key[N][N])

{

int inv[N][N],i,j,k;

int res=findInverse(key,inv);

char c[500];

// printf("%s\n",p);

if(!res)

{

printf("Inverse of this singular key matrix doesn't exist!!\n");

printf("Please enter suitable key matrix!!!\n");

}

else if(res==-1)

{

printf("Inverse of key matrix is non-existent as multiplicative inverse of its determinant doesn't exist!!");

printf("Please enter suitable key matrix!!!\n");

}

else

{

int pmat[size/N][N];

int cmat[size/N][N];

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

pmat[i/size][i%size]=(int)(p[i]-'a'),cmat[i/size][i%size]=0;

printf("The modified plain text in matrix form is:\n");

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

{

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

printf("%d ",pmat[i][j]);

printf("\n");

}

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

{

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

{

for(k=0;k<N;k++)

cmat[i][j]=(cmat[i][j]+(pmat[i][k]\*key[k][j]));

cmat[i][j]%=n;

}

}

printf("The cipher text to be transmitted in matrix form is:\n");

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

{

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

printf("%d ",cmat[i][j]);

printf("\n");

}

printf("Cipher text in string format is: ");

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

{

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

{

c[N\*i+j]=(char)((int)cmat[i][j]+65);

}

}

c[size]='\0';

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

}

return;

}

int main()

{

char p[500];

printf("--------------------------------------------------AT ALICE'S END-------------------------------------------\n");

printf("Enter the plain-text to be transmitted: ");

scanf("%[^\n]s",p);

printf("Enter the dimensions of key matrix: ");

scanf("%d",&N);

int key[N][N],i,j;

printf("Enter the shared secret key: \n");

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

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

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

preprocess(p);

encrypt(p,key);

return 0;

}

Decryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

const int n=26;

int N,size=0;

int multiplicative\_inverse(int k){

int i=0;

for(i=0;i<n;i++){

if((i\*k)%n==1){

return i;

}

}

return -1;

}

int additive\_inverse(int k){

return ((n+k)%n);

}

void fillCofactor(int mat[N][N],int temp[N][N],int r,int c){

int i=0,j=0,row=0,col=0;

for(row=0;row<N;row++){

for(col=0;col<N;col++){

if(row!=r&&col!=c){

temp[i][j++]=mat[row][col];

if(j==N-1){

j=0;

i++;

}

}

}

}

}

int findDeterminant(int mat[N][N],int dim){

int res=0;

if(dim==1)

return mat[0][0];

int temp[N][N],sign=1,col=0;

for(col=0;col<dim;col++){

fillCofactor(mat,temp,0,col);

res+=sign\*mat[0][col]\*findDeterminant(temp,dim-1);

// printf("%d\n",res);

sign=-sign;

}

return res;

}

void fillAdjoint(int mat[N][N],int adj[N][N]){

if(!N){

adj[0][0]=1;

return;

}

int sign=1,temp[N][N],i,j;

for(i=0;i<N;i++){

for(j=0;j<N;j++){

fillCofactor(mat,temp,i,j);

sign=(((i+j)%2)?-1:1);

adj[j][i]=sign\*(findDeterminant(temp,N-1));

}

}

}

int findInverse(int mat[N][N],int inv[N][N]){

int det=findDeterminant(mat,N);

printf("The determinant of the key matrix is: %d\n",det);

int detInv=multiplicative\_inverse(det);

int temp;

if(det==0)

return 0;

if(detInv==-1)

return -1;

int adj[N][N],i,j;

fillAdjoint(mat,adj);

for(i=0;i<N;i++){

for(j=0;j<N;j++){

temp=((adj[i][j]%n)\*(detInv%n))%n;

if(temp<0)

inv[i][j]=additive\_inverse(temp);

else

inv[i][j]=temp;

// printf("%d\n",temp );

}

}

return 1;

}

void decrypt(char c[100],int key[N][N])

{

int inv[N][N],i,j,k;

int res=findInverse(key,inv);

char p[500];

for(i=0;c[i]!='\0';i++)

size++;

if(!res)

{

printf("Inverse of this singular key matrix doesn't exist!!\n");

printf("Please enter suitable key matrix!!!\n");

}

else if(res==-1)

{

printf("Inverse of key matrix is non-existent as multiplicative inverse of its determinant doesn't exist!!");

printf("Please enter suitable key matrix!!!\n");

}

else

{

int pmat[size/N][N];

int cmat[size/N][N];

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

cmat[i/size][i%size]=(int)(c[i]-'A'),pmat[i/size][i%size]=0;

printf("The cipher text in matrix form is:\n");

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

{

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

printf("%d ",cmat[i][j]);

printf("\n");

}

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

{

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

{

for(k=0;k<N;k++)

pmat[i][j]=(pmat[i][j]+(cmat[i][k]\*inv[k][j]));

pmat[i][j]%=n;

}

}

printf("The plain text decrypted in matrix form is:\n");

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

{

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

printf("%d ",pmat[i][j]);

printf("\n");

}

printf("Plain text in string format is: ");

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

{

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

{

p[N\*i+j]=(char)(pmat[i][j]+97);

}

}

p[size]='\0';

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

}

return;

}

int main()

{

char c[100];

printf("--------------------------------------------------AT BOB'S END---------------------------------------------\n");

printf("Enter the cipher text to be received: ");

scanf("%[^\n]s",c);

printf("Enter the dimensions of key matrix: ");

scanf("%d",&N);

int key[N][N],i,j;

printf("Enter the shared secret key: \n");

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

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

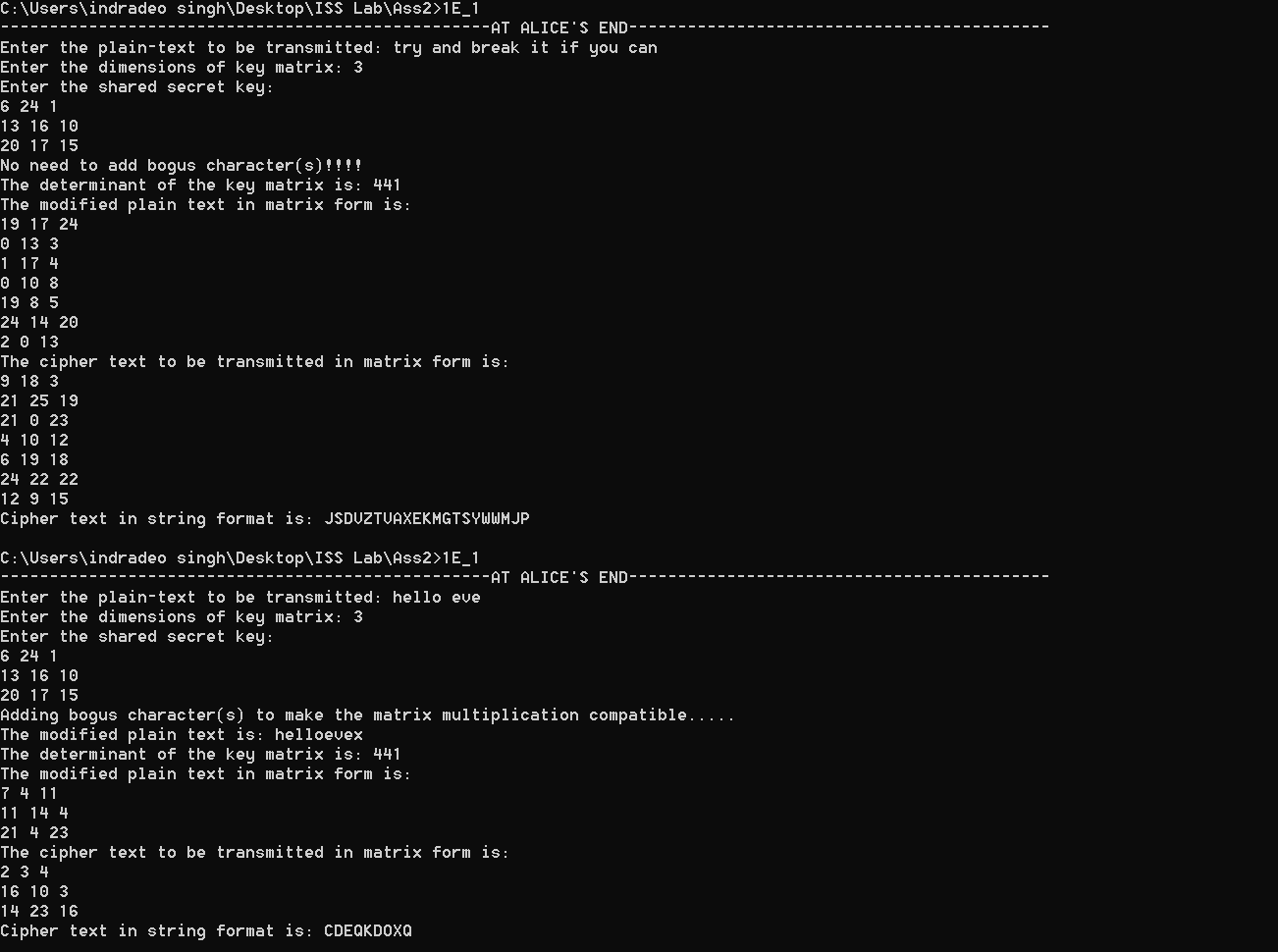
scanf("%d",&key[i][j]);

decrypt(c,key);

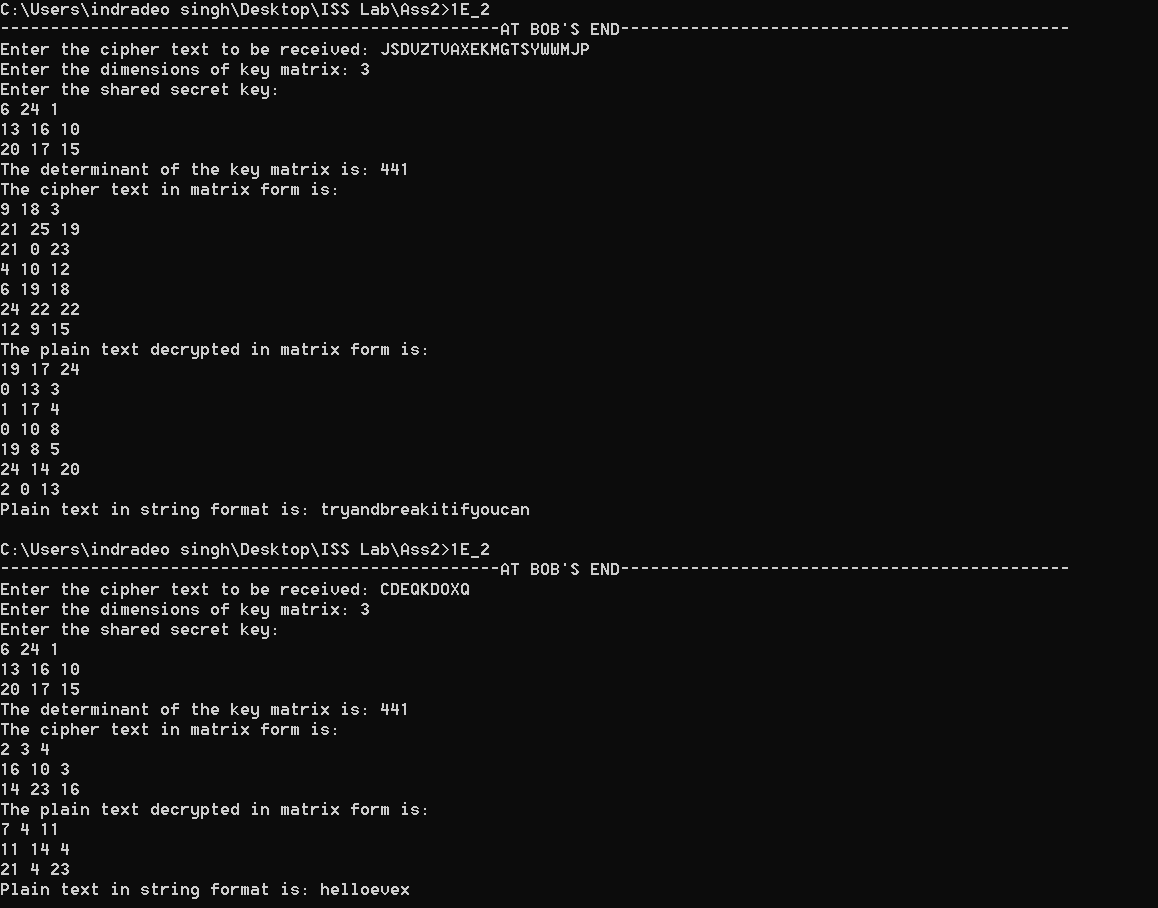
return 0;

}

Encryption Output



Decryption Output



2.) **Exhaustive Search Attacks**

a.) Shift Cipher

Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

const int n=26;

int additive\_inverse(int k)

{

return ((n-k)%n);

}

void bruteForceAttack(char c[100])

{

int key=0,i,j;

char p[100];

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

{

int keyInv=additive\_inverse(key);

for(j=0;c[j]!='\0';j++)

{

p[j]=(char)(((int)(c[j]-'A')+keyInv)%n+97);

}

printf("<----------For key = %d--------->\n",key);

printf("Plain-text decrypted is: %s\n\n\n",p);

}

return;

}

int main()

{

char c[100];

printf("--------------------------------------------------AT EVE'S END-------------------------------------------\n");

printf("Enter the cipher-text which is intercepted: ");

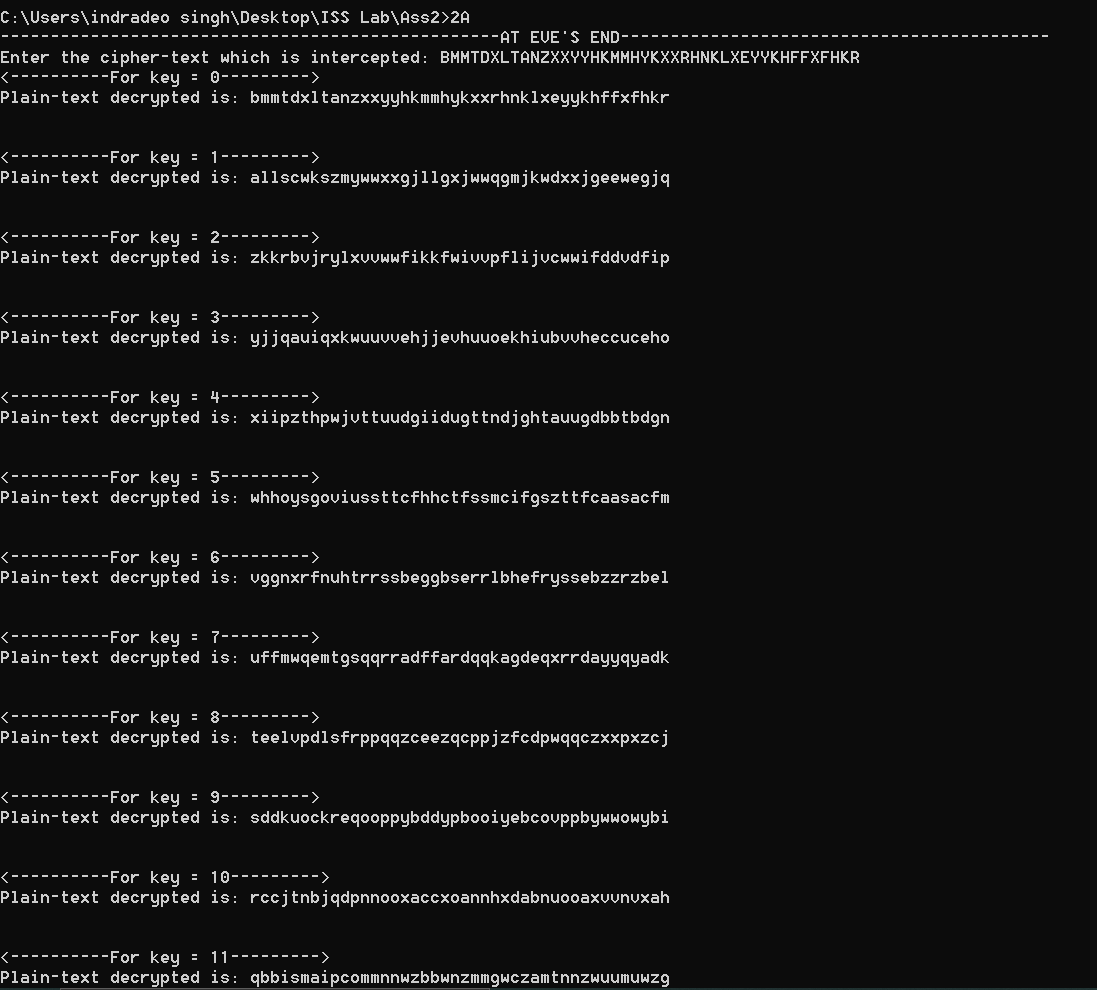
scanf("%[^\n]s",c);

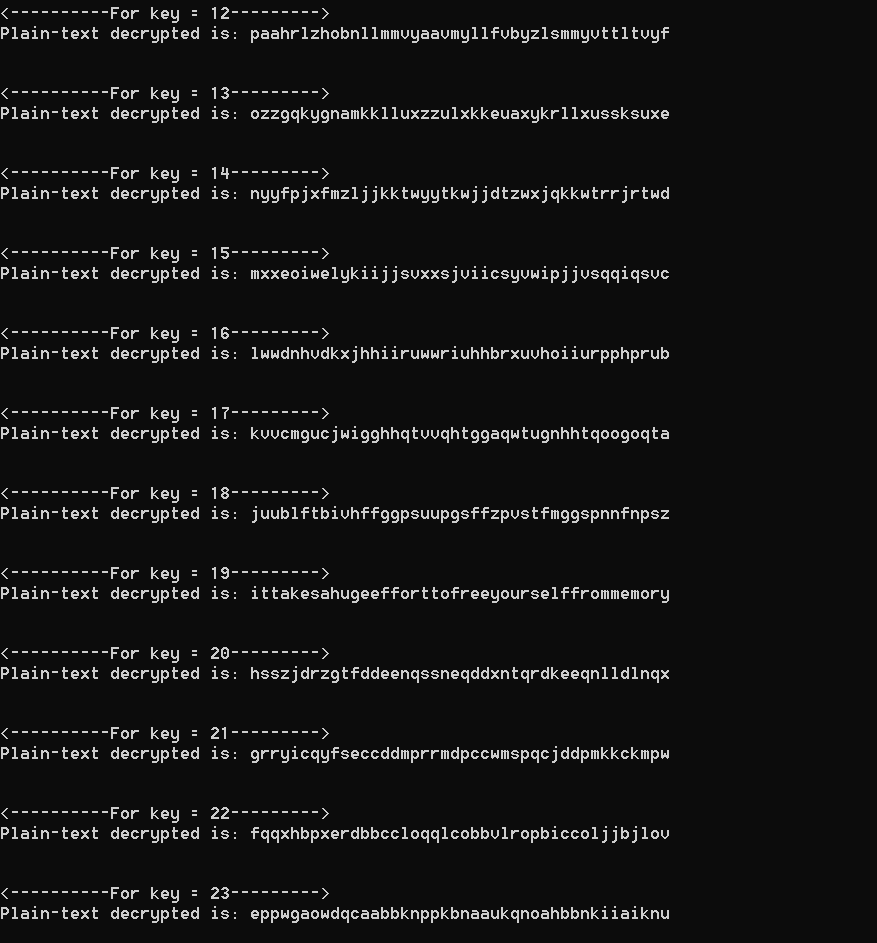
bruteForceAttack(c);

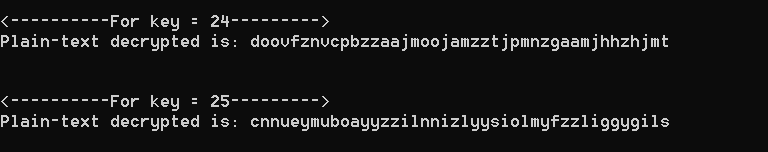
return 0;

}

Output







b.) Multiplicative Cipher

Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

const int n=26;

int multiplicative\_inverse(int k){

int i=0;

for(i=0;i<n;i++){

if((i\*k)%n==1){

return i;

}

}

return -1;

}

void bruteForceAttack(char c[100])

{

int key=0,i,j;

char p[100];

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

{

int keyInv=multiplicative\_inverse(key);

if(keyInv==-1)

{

printf("<----------For key = %d--------->\n",key);

printf("Key = %d is not in Z\* \n\n\n",key);

}

else

{

for(j=0;c[j]!='\0';j++)

{

p[j]=(char)(((int)(c[j]-'A')\*keyInv)%n+97);

}

printf("<----------For key = %d--------->\n",key);

printf("Plain-text decrypted is: %s\n\n\n",p);

}

}

return;

}

int main()

{

char c[100];

printf("--------------------------------------------------AT EVE'S END-------------------------------------------\n");

printf("Enter the cipher-text which is intercepted: ");

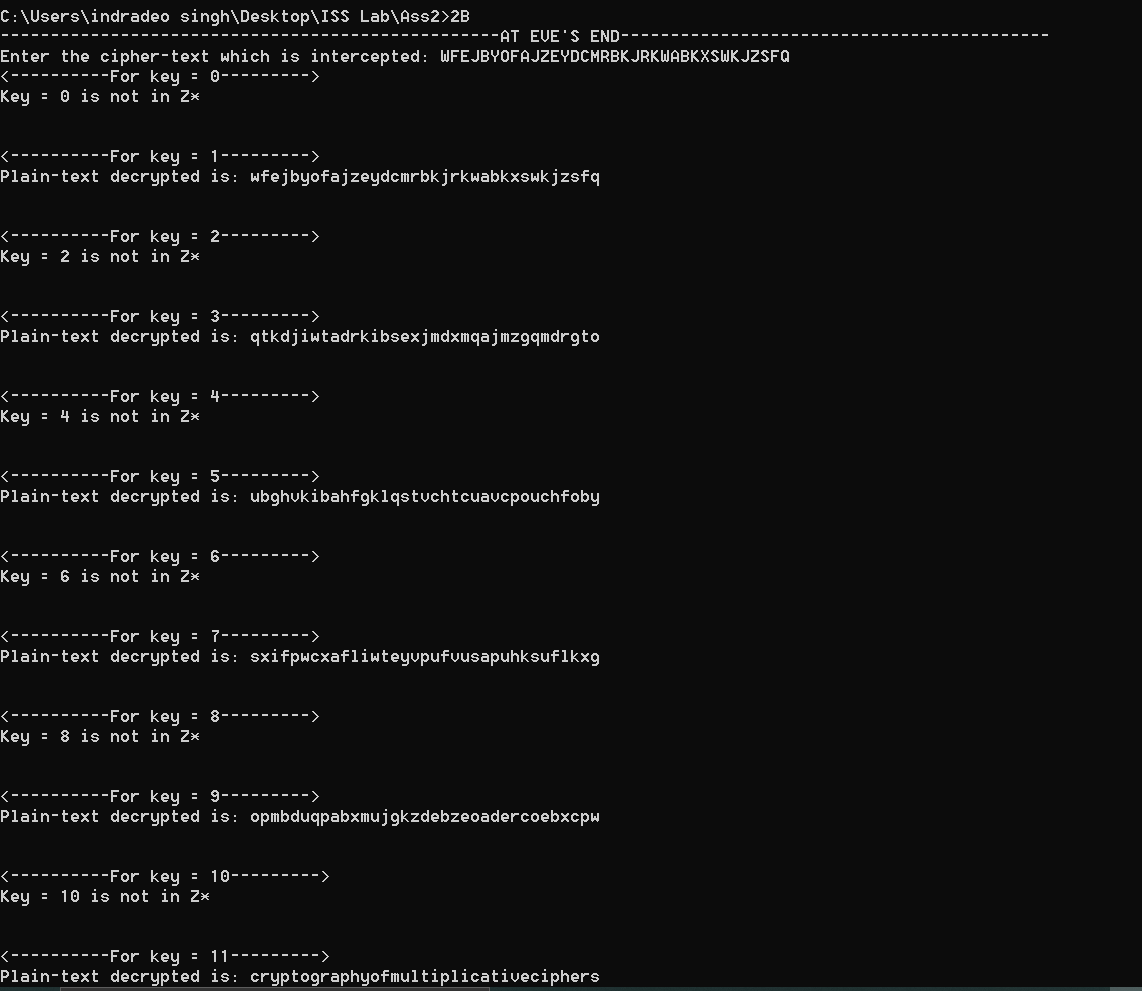
scanf("%[^\n]s",c);

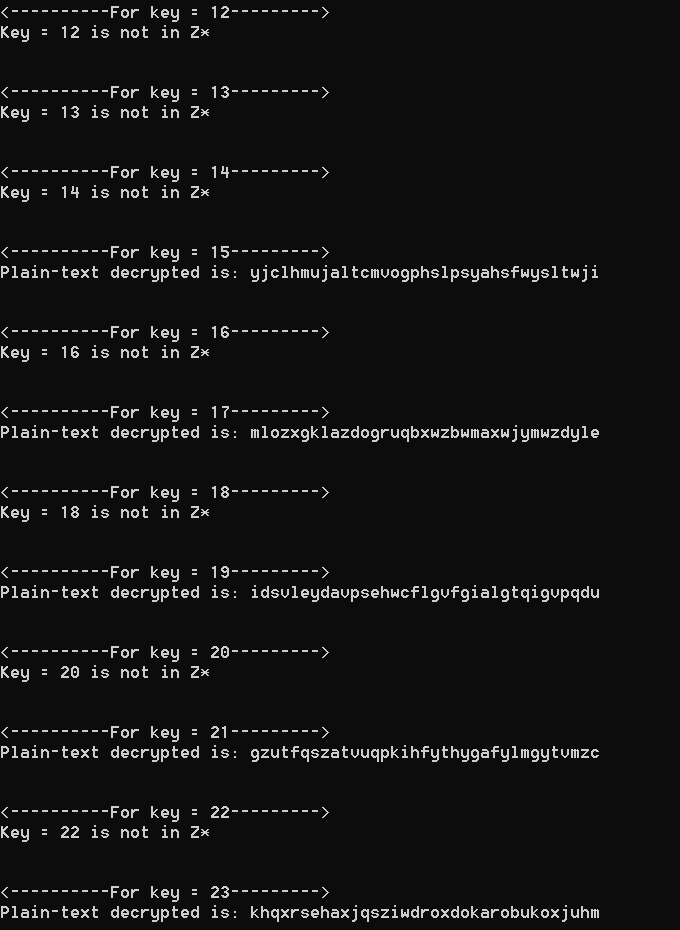
bruteForceAttack(c);

return 0;

}

Output







c.) Affine Cipher

Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

const int n=26;

int additive\_inverse(int k)

{

return ((n-k)%n);

}

int multiplicative\_inverse(int k){

int i=0;

for(i=0;i<n;i++){

if((i\*k)%n==1){

return i;

}

}

return -1;

}

void bruteForceAttack(char c[100])

{

int k1=0,i,j,k2=0;

char p[100];

char ch[100];

int flag=1;

for(k1=0;k1<n&&flag==1;k1++)

{

int k1Inv=multiplicative\_inverse(k1);

printf("Enter YES to proceed further, else N0:");

scanf("%s",ch);

if(strcmp(ch,"NO")==0)

break;

if(k1Inv==-1)

{

printf("<----------For k1 = %d--------->\n",k1);

printf("k1 = %d is not in Z\*, so not searching for all domain of k2 for it!!!!\n\n\n",k1);

}

else

{

for(k2=0;k2<n;k2++)

{

int k2Inv=additive\_inverse(k2);

for(j=0;c[j]!='\0';j++)

{

p[j]=(char)(((((int)(c[j]-'A')+k2Inv)%n)\*k1Inv)%n+97);

}

printf("<----------For k1 = %d , k2 = %d--------->\n",k1,k2);

printf("Plain-text decrypted is: %s\n\n\n",p);

}

}

}

return;

}

int main()

{

char c[100];

printf("--------------------------------------------------AT EVE'S END-------------------------------------------\n");

printf("Enter the cipher-text which is intercepted: ");

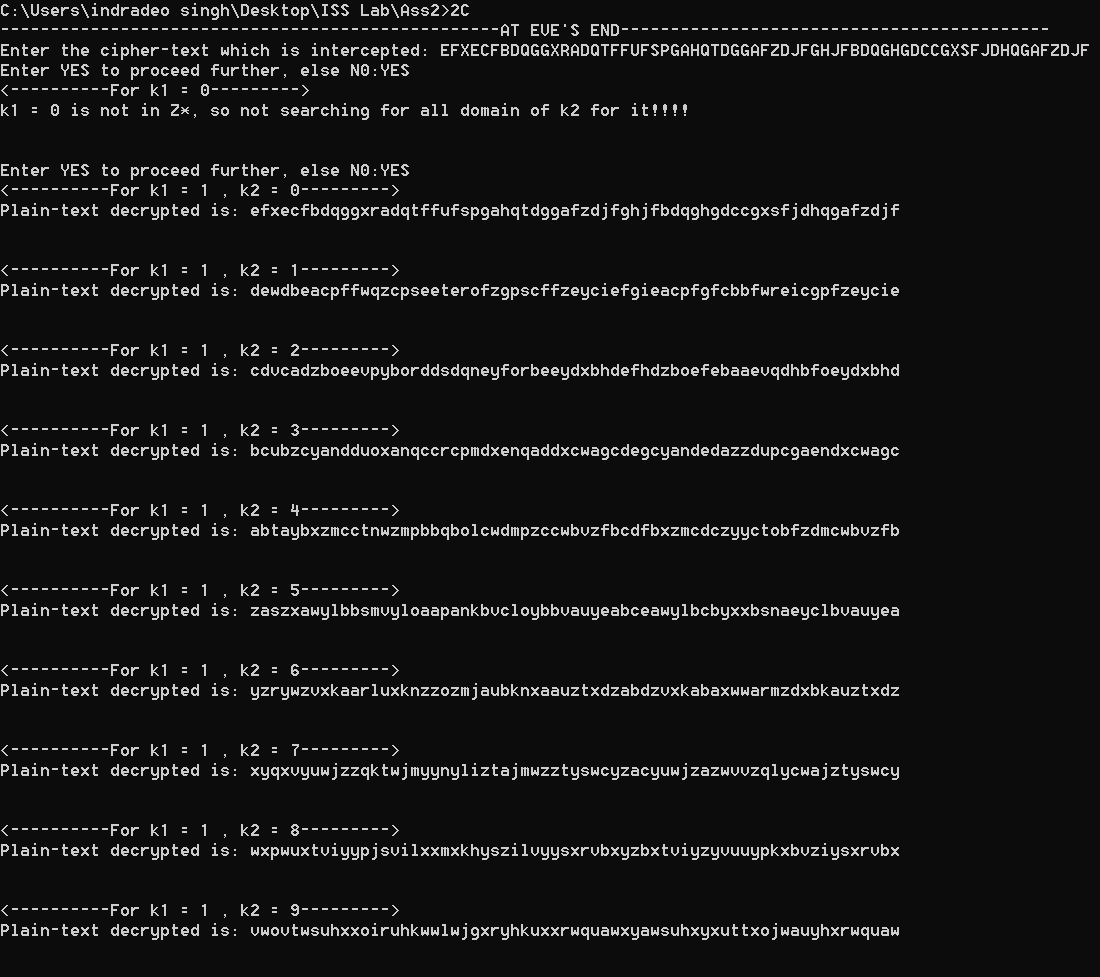
scanf("%[^\n]s",c);

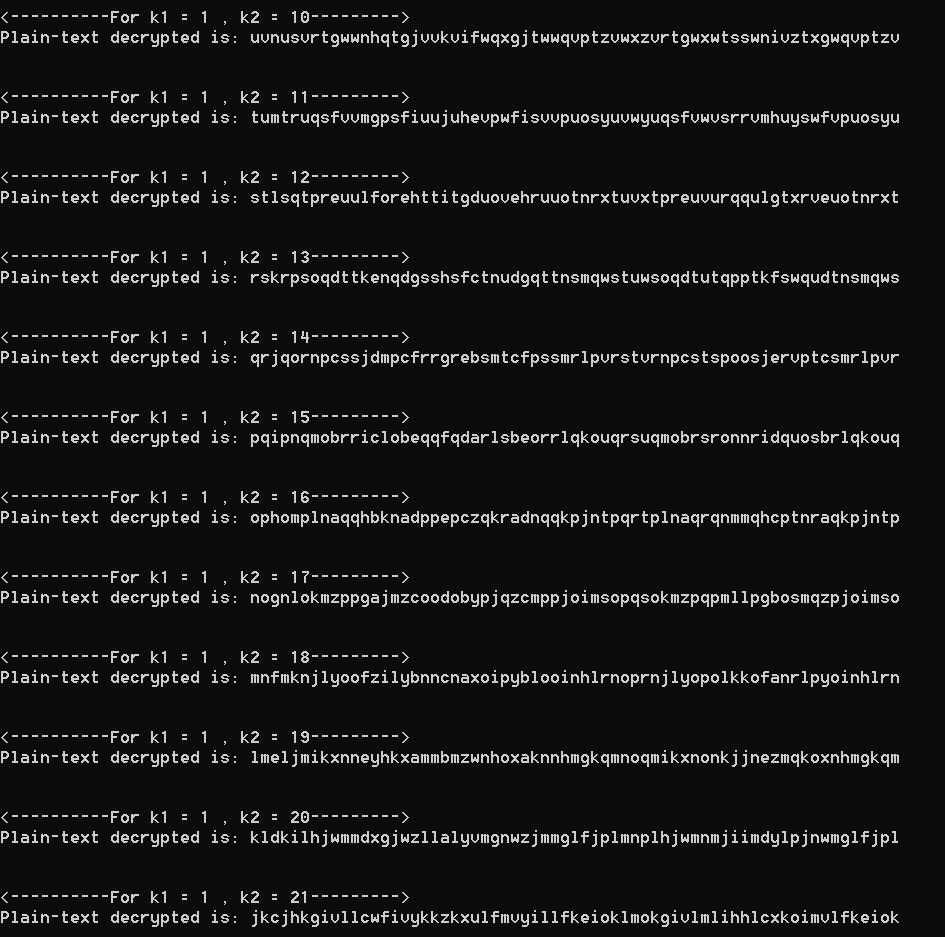
bruteForceAttack(c);

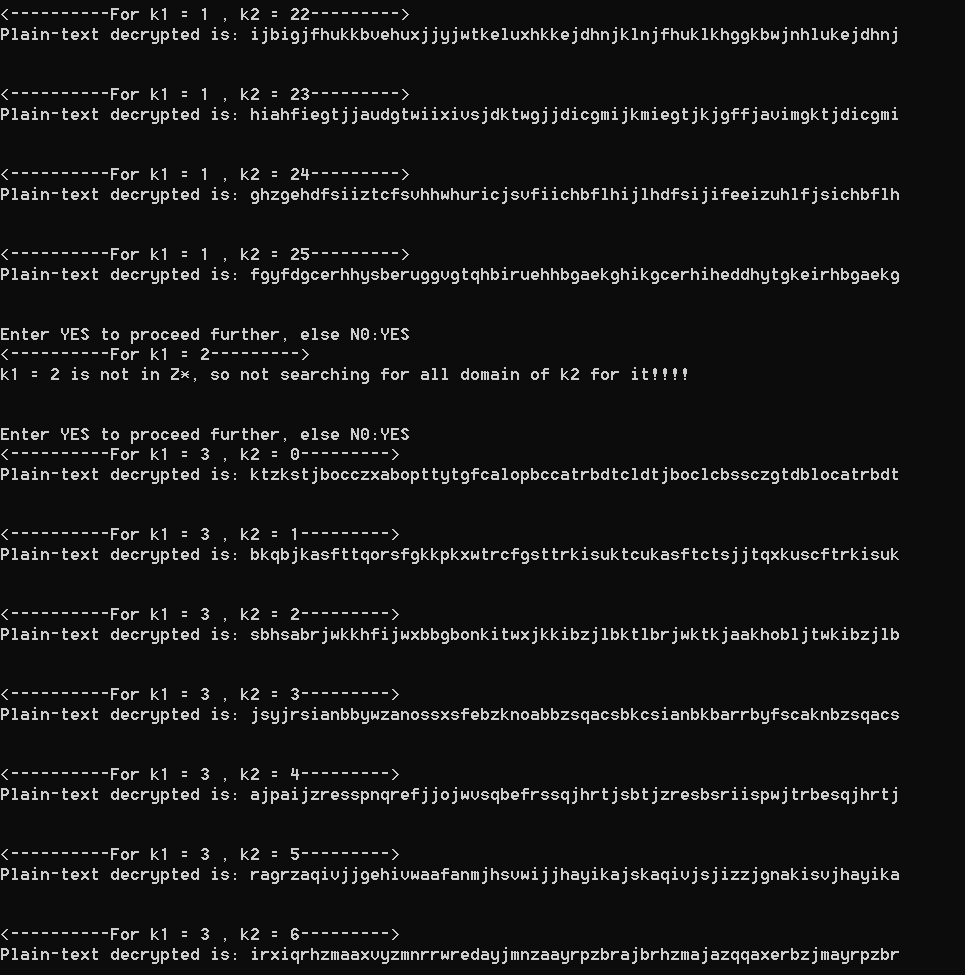
return 0;

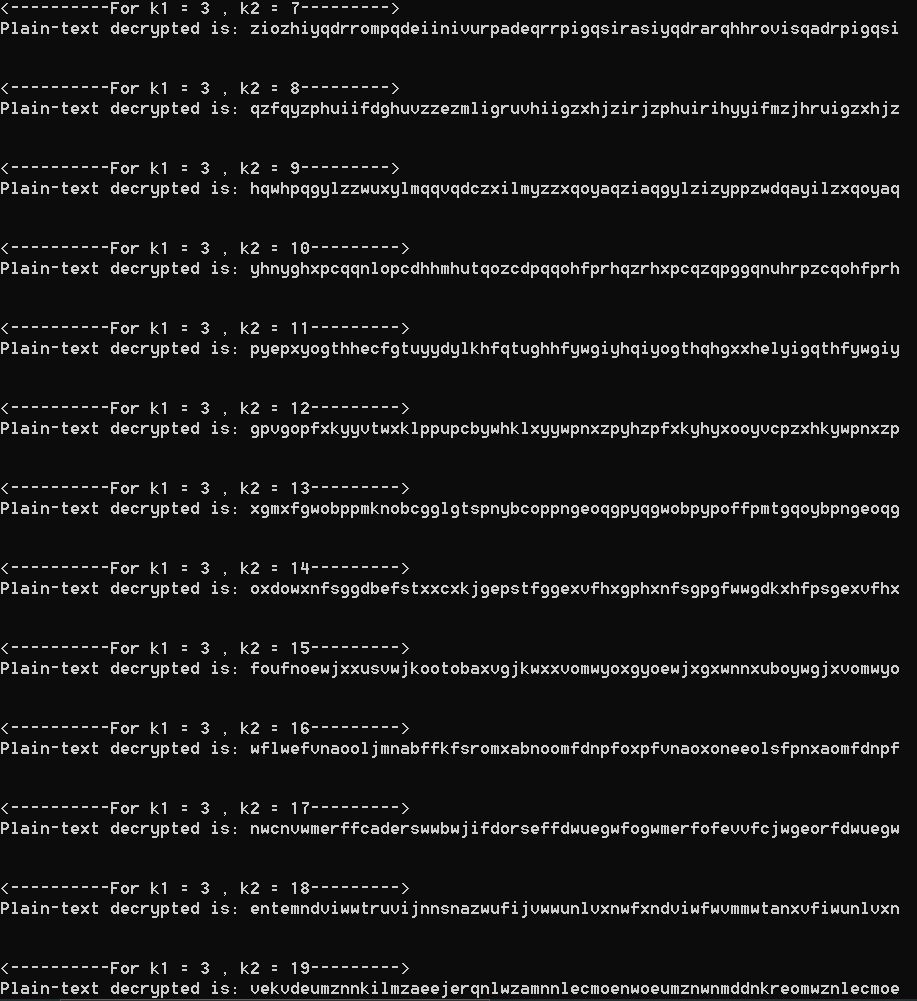
}

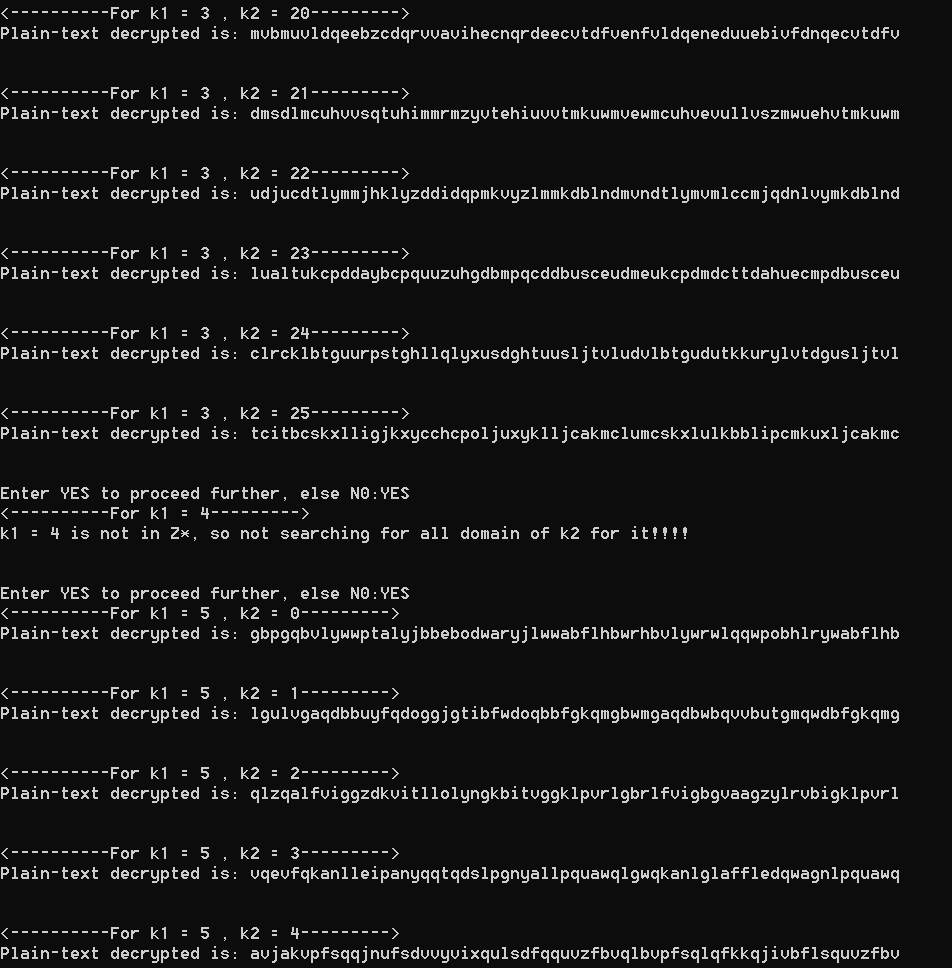
Output

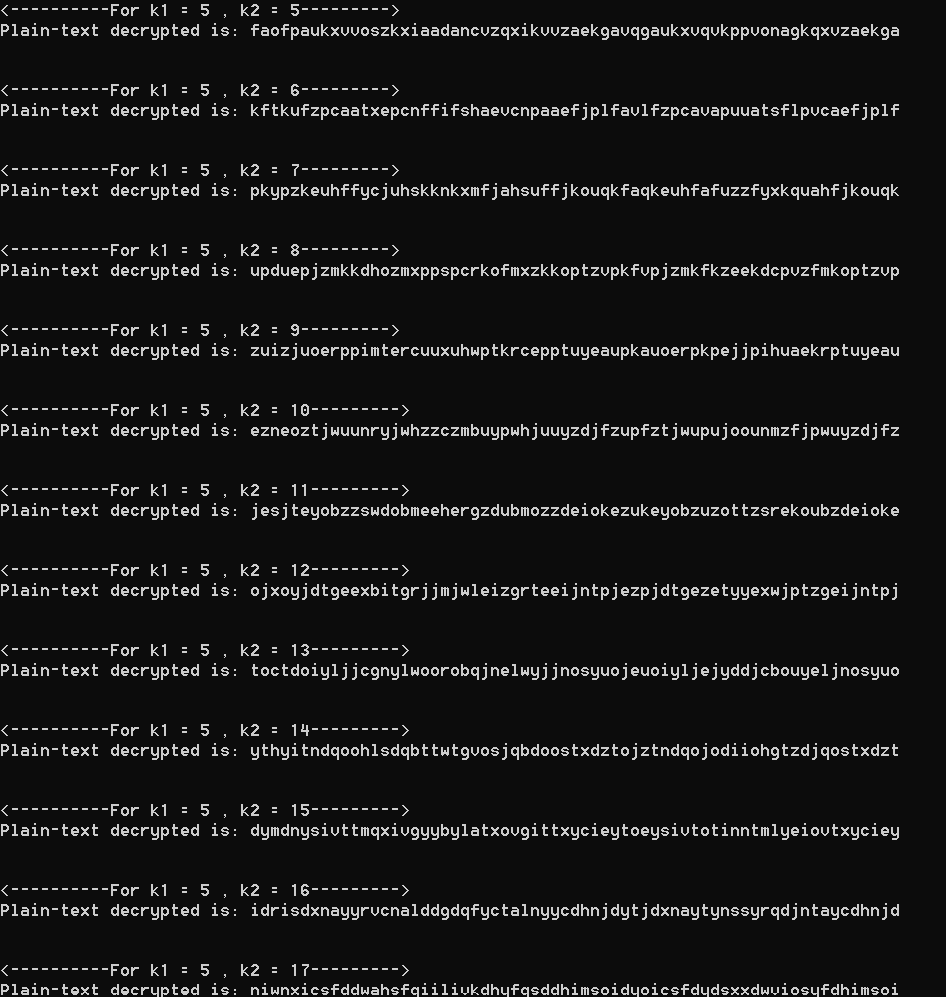


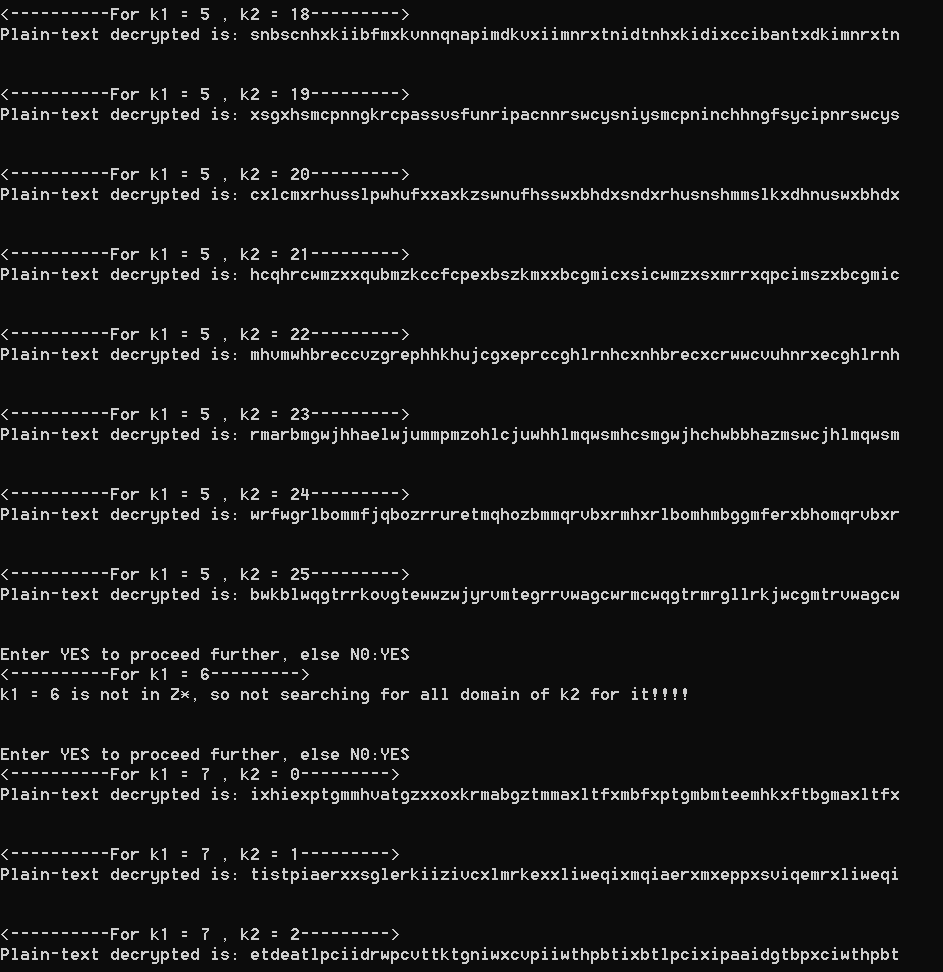














------------------------------------ THE END ------------------------------------------