1) Autokey 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 k1)

{

int i=0,j=0,flag=1;

k1=k1%n;

char c[100];

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

if(p[i]==' ')

continue;

if(flag)

{

flag=0;

if(islower(p[i]))

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

else if(isupper(p[i]))

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

}

else

{

int ind=i-1;

while(ind>0&&p[ind]==' ')

ind--;

if(islower(p[i]))

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

else if(isupper(p[i]))

c[j++]=(char)((int)(p[i]+p[ind]-65-65)%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 (k1): ");

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);

}

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

{

int i=0;

k1=k1%n;

int flag=1;

char p[100];

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

{

if(flag)

{

flag=0;

if(islower(c[i]))

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

else if(isupper(c[i]))

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

}

else

{

if(islower(p[i-1]))

{

int k=(int)(p[i-1]-97);

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 if(isupper(p[i-1]))

{

int k=(int)(p[i-1]-65);

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);

}

}

}

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 (k1): ");

scanf("%d",&k);

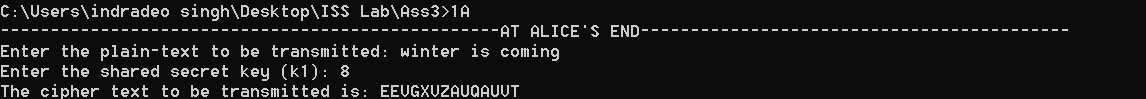
printf("The plain text decrypted is: ");

decrypt(c,k);

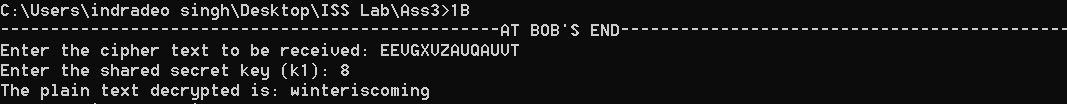
return 0;

}

Encryption Output



Decryption Output



2.)

a.) Vigenere Cipher

Encryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

#include<time.h>

const int n=26;

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

{

FILE \*fp=fopen("2A\_key.txt","r");

int key[l];

int i=0,j=0;

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

fscanf(fp,"%d",&key[i]);

i=0;

char c[100];

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

{

if(p[i]==' ')

continue;

else

{

if(islower(p[i]))

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

else if(isupper(p[i]))

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

}

}

c[j]='\0';

printf("%s",c);

fclose(fp);

return;

}

int main()

{

char p[100];

int l,i;

srand(time(0));

FILE \*fp;

fp=fopen("2A\_key.txt","w");

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

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

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

printf("\n\nEnter the size of the key stream: ");

scanf("%d",&l);

printf("\n\nGenerating pseudo random-key stream and saving it in 2A\_key.txt...\n");

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

{

int k=rand()%n;

fprintf(fp,"%d ",k);

}

fclose(fp);

printf("\n\nThe cipher text to be transmitted is: ");

encrypt(p,l);

return 0;

}

Decryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

#include<time.h>

const int n=26;

int additive\_inverse(int k){

return ((n-k)%n);

}

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

{

FILE \*fp=fopen("2A\_key.txt","r");

int key[l];

int i=0,j=0;

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

fscanf(fp,"%d",&key[i]);

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(key[i%l]))%n+65);

else if(isupper(c[i]))

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

}

p[i]='\0';

printf("%s",p);

fclose(fp);

return;

}

int main()

{

char c[100];

int l;

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

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

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

printf("\n\nEnter the size of the key stream: ");

scanf("%d",&l);

printf("\n\nThe plain text decrypted is: ");

decrypt(c,l);

return 0;

}

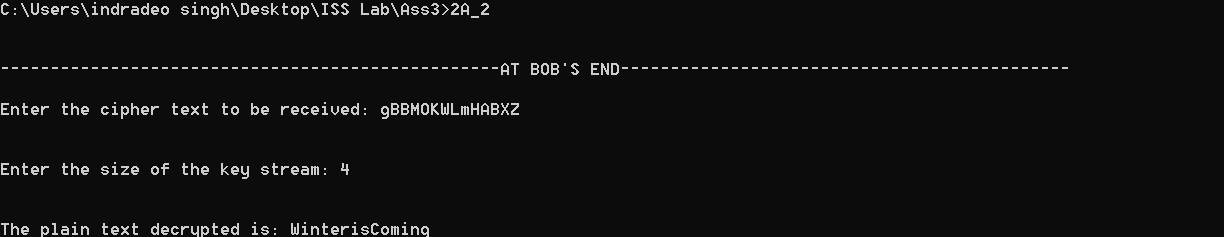
Encryption Output



Key file

10 19 14 19

Decryption Output



b.) Keyed Transposition Cipher

Encryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

#include<time.h>

const int n=26;

char modifiedP[100];

int key[100];

int getNumByFisherYatesShuffle(int n)

{

srand(time(NULL));

int ind=rand()%n;

int num=key[ind];

key[ind]=key[ind]+key[n-1];

key[n-1]=key[ind]-key[n-1];

key[ind]=key[ind]-key[n-1];

return num;

}

void generatePseudoRandomKeys(int l)

{

FILE \*fp=fopen("2B\_key.txt","w");

int i=0;

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

key[i]=i+1;

for(i=l;i>0;i--)

key[i-1]=getNumByFisherYatesShuffle(i);

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

fprintf(fp,"%d ",key[i]);

fclose(fp);

}

void preprocess(char p[100],int l)

{

int i,j=0;

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

{

if(p[i]==' ')

continue;

else

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

}

modifiedP[j]='\0';

if(strlen(modifiedP)%l!=0)

{

printf("\n\nInserting bogus character(s) to divide the plain-text into blocks of appropriate size..\n");

while(strlen(modifiedP)%l!=0)

modifiedP[strlen(modifiedP)]='x';

modifiedP[strlen(modifiedP)]='\0';

printf("\n\nThe modified plain-text is: %s\n",modifiedP);

}

return;

}

void encrypt(int l)

{

int i=0,j=0,k=0;

i=0;

char c[100];

char window[l];

int size=0;

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

{

if(size!=l)

{

window[size]=modifiedP[i];

size++;

}

else

{

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

c[j++]=window[key[k]-1];

size=0;

window[size]=modifiedP[i];

size++;

}

}

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

c[j++]=window[key[k]-1];

c[j]='\0';

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

return;

}

int main()

{

char p[100];

int l,i;

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

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

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

printf("\n\nEnter the block size of key: ");

scanf("%d",&l);

int key[l];

preprocess(p,l);

printf("\n\nGenerating pseudo random-key stream and saving it in 2B\_key.txt...\n");

generatePseudoRandomKeys(l);

printf("\n\nThe cipher text to be transmitted is: ");

encrypt(l);

return 0;

}

Decryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

#include<time.h>

const int n=26;

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

{

FILE \*fp=fopen("2B\_key.txt","r");

int key[l],inverseKey[l+1],window[l];

int i=0,j=0,size=0,k=0;

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

fscanf(fp,"%d",&key[i]);

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

inverseKey[key[i-1]]=i;

i=0;

char p[100];

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

{

if(size!=l)

{

window[size]=c[i];

size++;

}

else

{

for(k=1;k<=l;k++)

p[j++]=window[inverseKey[k]-1];

size=0;

window[size]=c[i];

size++;

}

}

for(k=1;k<=l;k++)

p[j++]=window[inverseKey[k]-1];

p[j]='\0';

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

fclose(fp);

return;

}

int main()

{

char c[100];

int l;

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

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

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

printf("\n\nEnter the block size of key: ");

scanf("%d",&l);

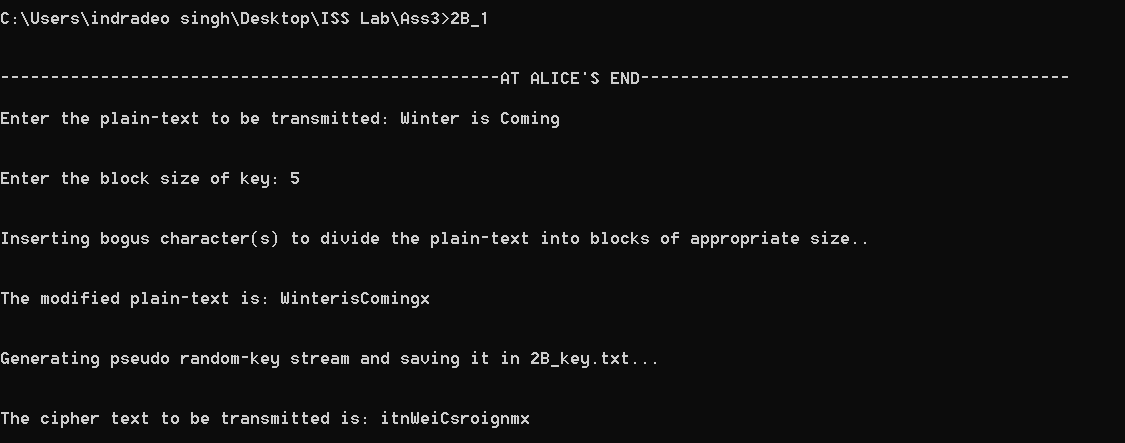
printf("\n\nThe plain text decrypted is: ");

decrypt(c,l);

return 0;

}

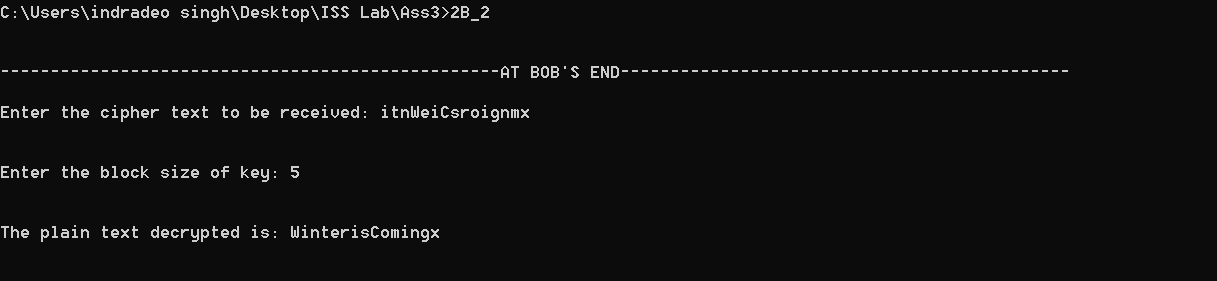
Encryption Output



Key File

2 4 3 1 5

Decryption Output



3.) Hill Cipher as a permutation cipher

Encryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

#include<time.h>

#include<string.h>

const int n=26;

int N,size=0;

int perKey[100];

char modifiedP[100];

void preprocess(char p[500])

{

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

size=0;

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

{

if(p[i]!=' ')

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

}

modifiedP[j]='\0';

if(strlen(modifiedP)%N!=0)

{

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

while(size%N!=0)

modifiedP[size]='x',size++;

modifiedP[size]='\0';

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

}

else

{

printf("\n\nNo need to add bogus character(s)!!!!!!!!!!!\n");

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

}

return;

}

void encrypt(int key[N][N])

{

int i,j,k;

char c[500];

int pmat[size/N][N];

int cmat[size/N][N];

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

{

if(isupper(modifiedP[i]))

pmat[i/size][i%size]=(int)(modifiedP[i]),cmat[i/size][i%size]=0;

else

pmat[i/size][i%size]=(int)(modifiedP[i]),cmat[i/size][i%size]=0;

}

printf("\n\nThe 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]));

}

}

printf("\n\nThe 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("\n\n\nCipher text in string format is: ");

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

{

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

{

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

}

}

c[size]='\0';

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

return;

}

int main()

{

char p[500];

int i,j;

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

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

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

printf("\nEnter the key size: ");

scanf("%d",&N);

printf("\nEnter the permutation key: ");

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

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

int key[N][N];

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

{

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

{

if(i==perKey[j]-1)

key[i][j]=1;

else

key[i][j]=0;

}

}

printf("\n\nKey matrix to be used in this case: \n");

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

{

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

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

printf("\n");

}

preprocess(p);

encrypt(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 perKey[100];

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

{

int i,j,k;

char p[500];

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

size++;

int pmat[size/N][N];

int cmat[size/N][N];

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

{

if(isupper(c[i]))

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

else

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

}

printf("\n\nThe 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]\*key[k][j]));

}

}

printf("\n\nThe 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("\n\nPlain 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]);

}

}

p[size]='\0';

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

return;

}

int main()

{

char c[100];

int i,j;

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

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

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

printf("\nEnter the key size: ");

scanf("%d",&N);

printf("\nEnter the permutation key: ");

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

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

int key[N][N];

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

{

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

{

if(i==perKey[j]-1)

key[j][i]=1;

else

key[j][i]=0;

}

}

printf("Key matrix to be used in this case: \n");

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

{

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

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

printf("\n");

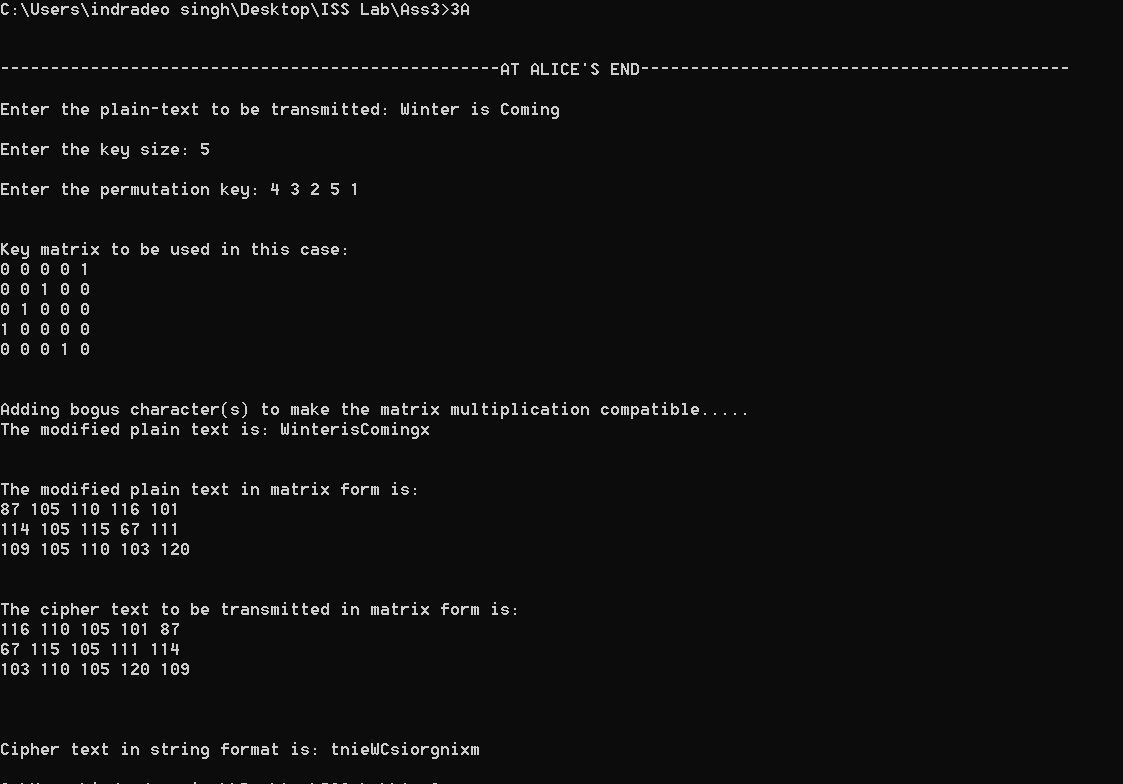
}

decrypt(c,key);

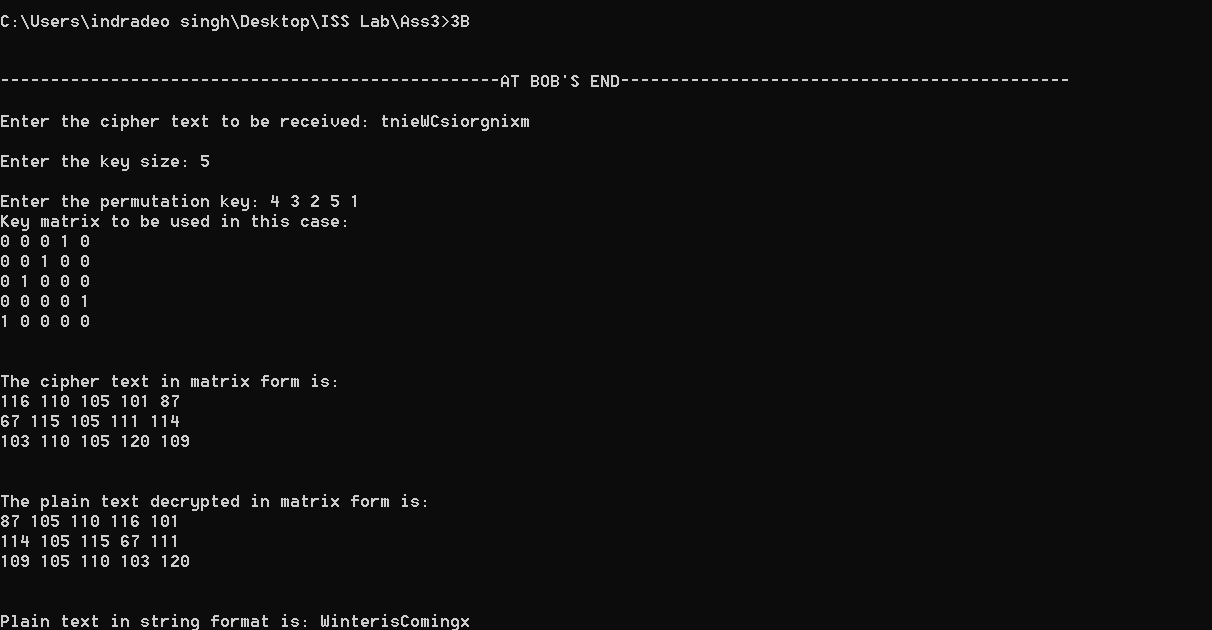
return 0;

}

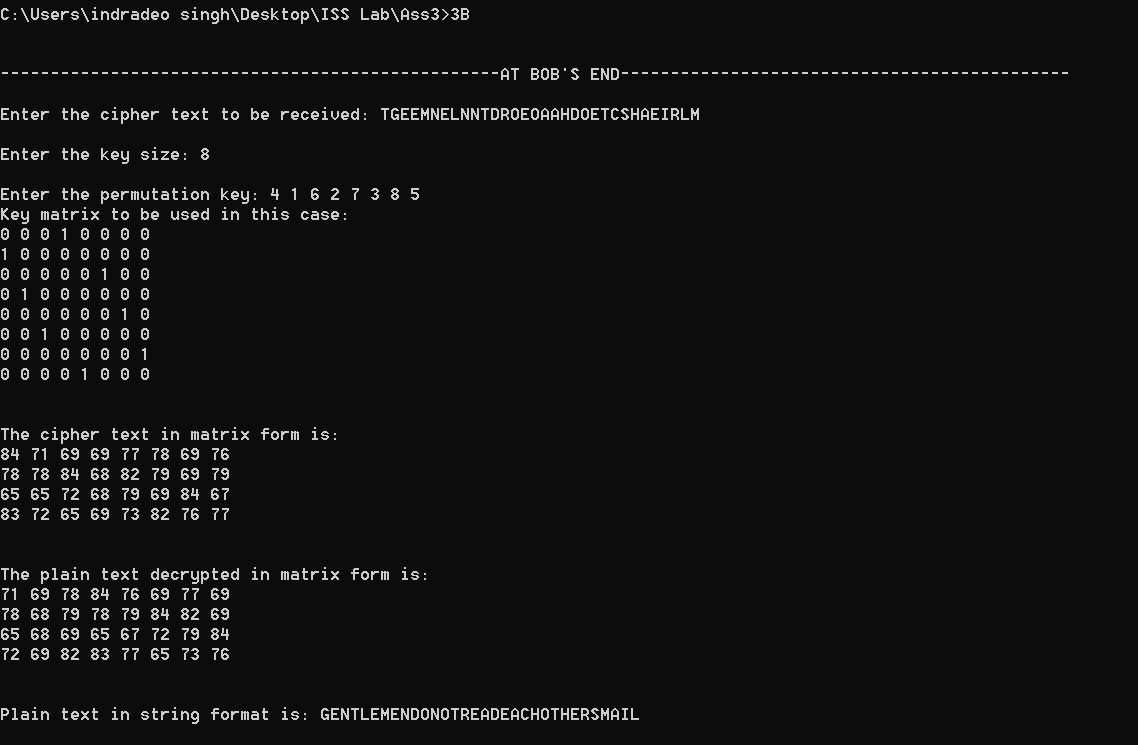
Encryption Output (Sample)



Decryption Output (Sample)



Decryption Output (A/Q)



4.) a.) One Time Pad

Encryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

#include<time.h>

const int n=26;

char modifiedP[100];

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

{

FILE \*fp=fopen("4A\_key.txt","w");

int i=0,j=0;

i=0;

char c[100];

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

{

if(p[i]==' ')

continue;

else

{

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

}

}

modifiedP[j]='\0';

printf("\n\nThe modified plain-text is: %s\n",modifiedP);

int key[j];

printf("\nGenerating pseudo random-key stream and saving it in 4A\_key.txt...\n");

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

{

key[i]=rand()%2;

fprintf(fp,"%d ",key[i]);

}

j=0;

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

{

c[i]=(char)(((int)modifiedP[i]-48)^key[i]+48);

}

printf("\n\nThe cipher text to be transmitted is: ");

c[i]='\0';

printf("%s",c);

fclose(fp);

return;

}

int main()

{

char p[100];

int l,i;

srand(time(0));

printf("\n\n--------------------------------------------------ONE TIME PAD---------------------------------------------\n");

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

printf("\n\nEnter the plain-text(binary string) to be transmitted: ");

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

encrypt(p,l);

return 0;

}

Decryption Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

#include<time.h>

#include<string.h>

const int n=26;

void decrypt(char c[100])

{

FILE \*fp=fopen("4A\_key.txt","r");

int key[strlen(c)];

int i=0,j=0;

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

fscanf(fp,"%d",&key[i]);

i=0;

char p[100];

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

{

p[i]=(char)(((int)c[i]-48)^key[i]+48);

}

p[i]='\0';

printf("%s",p);

fclose(fp);

return;

}

int main()

{

char c[100];

printf("\n\n--------------------------------------------------ONE TIME PAD---------------------------------------------\n");

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

printf("\nEnter the cipher text(binary string) to be received: ");

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

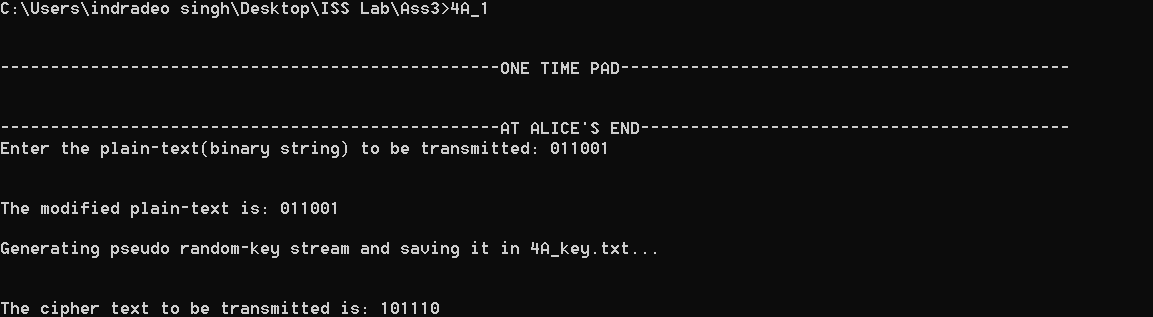
printf("\n\nThe plain text(binary string) decrypted is: ");

decrypt(c);

return 0;

}

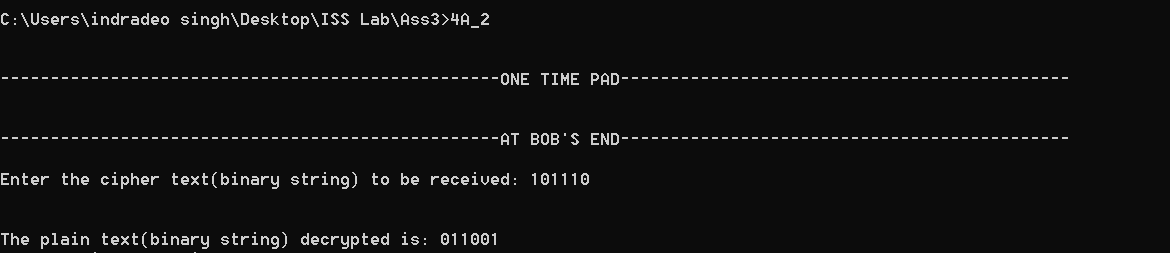
Encryption Output



Key File

1 1 0 1 1 1

Decryption Output



b.) Cryptanalysis of OTP

Code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

#include<time.h>

#include<string.h>

const int n=26;

int key[100];

char p[100];

char c[100];

char favourableP[100];

int additive\_inverse(int k)

{

int temp=n;

while(temp>=k)

temp+=n;

return ((temp-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;

}

int binaryToDecimal(int l,int r)

{

int dec=0,temp=1,i;

for(i=r;i>=l;i--)

{

dec+=key[i]\*temp;

temp\*=2;

}

return dec;

}

void startCryptanalysis()

{

int i=0,j=0,s0,a,b,s1,s2;

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

{

if(p[i]==' ')

continue;

else if(j==15)

break;

else

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

}

favourableP[j]='\0';

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

key[i]=((int)(favourableP[i])-48)^((int)c[i]-48);

s0=binaryToDecimal(0,4);

s1=binaryToDecimal(5,9);

s2=binaryToDecimal(10,14);

int mulInv=multiplicative\_inverse(s1+additive\_inverse(s0));

if(mulInv==-1)

printf("\n\n Cryptanalysis can't be done as the multiplicative inverse of (s1-s0) can't be found!!!!");

else

{

a=((mulInv%n)\*((s2+additive\_inverse(s1))%n))%n;

b=((mulInv%n)\*(((s1\*s1+additive\_inverse(s0\*s2))%n)%n))%n;

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*Cryptanalysis is successful with the following results\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\nS0 is: %d",s0);

printf("\na is: %d",a);

printf("\nb is: %d",b);

}

}

int main()

{

printf("\n\n--------------------------------------ONE TIME PAD WITH GIVEN PRNG FUNCTION--------------------------------\n");

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

printf("\nEnter the atleast 15 bits of intercepted cipher text: ");

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

fflush(stdin);

printf("\nEnter the 15 bits of the intercepted plain text: ");

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

if(strlen(c)<15||strlen(p)<15)

{

printf("\n\nWarning, Please enter the appropriate number of bits as required!!!!!!");

}

else

{

startCryptanalysis();

}

return 0;

}

Output

