Description:

1. **Perform encryption, decryption using the following substitution techniques**

a) Ceaser cipher

Discussion: The Caesar cipher is one of the earliest known and simplest ciphers. It is a type of substitution cipher in which each letter in the plaintext is 'shifted' a certain number of places down the alphabet. For example, with a shift of 1, A would be replaced by B, B would become C, and so on. The method is named after Julius Caesar, who apparently used it to communicate with his generals. More complex encryption schemes such as the Vigenere employ the Caesar cipher as one element of the encryption process. The widely known ROT13 'encryption' is simply a Caesar cipher with an offset of 13. The Caesar cipher offers essentially no communication security, and it will be shown that it can be easily broken even by hand.

To pass an encrypted message from one person to another, it is first necessary that both parties have the 'key' for the cipher, so that the sender may encrypt it and the receiver may decrypt it. For the caesar cipher, the key is the number of characters to shift the cipher alphabet.

First we translate all of our characters to numbers, 'a'=0, 'b'=1, 'c'=2, ... , 'z'=25. We can now represent the caesar cipher encryption function, e(x), where x is the character we are encrypting, as:

$$e(x) = (x + k)(mod\ 26)$$

Where k is the key (the shift) applied to each letter. After applying this function the result is a number which must then be translated back into a letter. The decryption function is :

$$e(x) = (x - k)(mod\ 26)$$

METHODOLOGY FOLLOWED:

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

int main(){
```

```
ifstream fin;
fin.open("input.txt") ;
string st;

string keyst;
getline(fin,keyst);
cout<<keyst<<"\n";

 int key;

if(keyst.size()==1){
    key=keyst[0]-'0';
}
else{
    key= (keyst[0]-'0')*10 + (keyst[1]-'0');
}

cout<<"\nThe CAESERKEY between 0 and 25 is:  "<<key<<"\n";
cout<<"\n*** The PLAIN TEXT for Encryption is :\n\n" ;

ofstream fout;

fout.open("output.txt");

while(getline(fin,st)){

  cout<<st<<"\n";
  int n = st.size();

   for(int i=0;i<n;i++){
     if(st[i]!=' '){
     st[i] ='a'+ (st[i]-'a'+key)%26;
     }
   }

  fout<<st<<"\n";

}

 fout.close();
 fin.close();

string st2;
ifstream fin2;
fin2.open("output.txt");

 ofstream fout2;
```

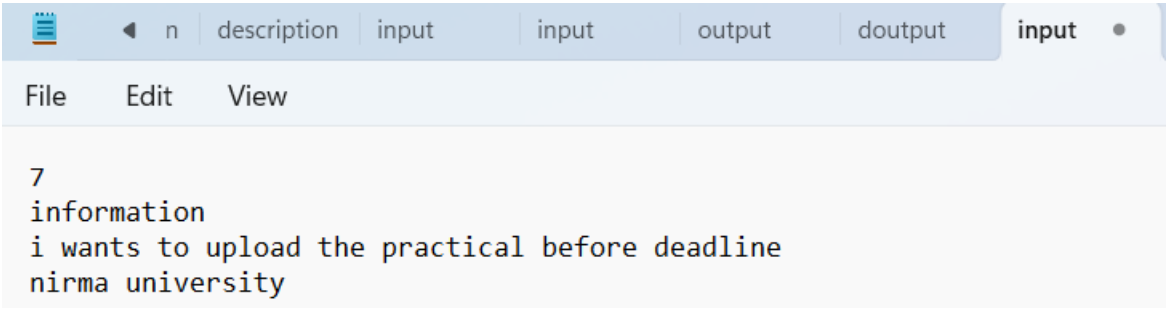
```
    fout2.open("doutput.txt");

   cout<<"\n*** The Encrypted text is :\n\n" ;

    while(getline(fin2,st2)){

     int n = st2.size();
     cout<<st2<<"\n";

      for(int i=0;i<n;i++){
        if(st2[i]!=' '){
        int a=st2[i]-'a'-key;

         if(a<0){
         st2[i] ='a'+ ((a % 26) + 26) % 26;
         }
         else{
             st2[i] ='a'+ a%26;
         }

        }
      }

     fout2<<st2<<"\n";

   }

    fin2.close();
    fout2.close();


    cout<<"\n*** Decrypted  text is : \n\n";

    ifstream fin3;
    fin3.open("doutput.txt");
    string st3;


   while(getline(fin3,st3)){
       cout<<st3<<"\n";
   }

   fin3.close();


    return 0;
}
```

- **INPUT:**

  - Here program gets Input from input.txt file
  - Input Formate:

    key – first line of an input file

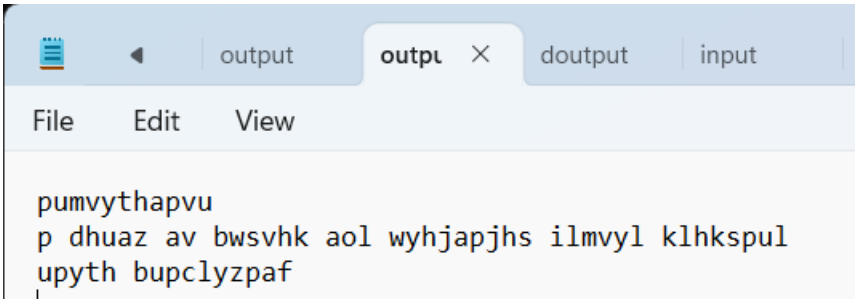
    Plain text – from 2nd line to last line of input file.



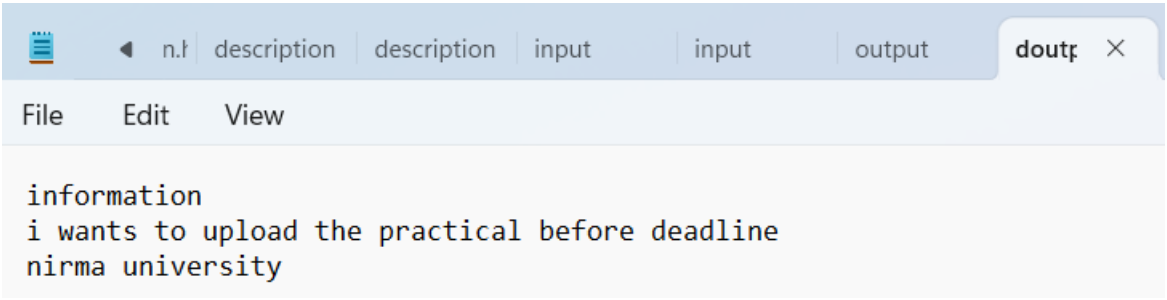
- Output:

```
7

The CAESERKEY between 0 and 25 is:  7

*** The PLAIN TEXT for Encryption is :

information
i wants to upload the practical before deadline
nirma university

*** The Encrypted text is :

pumvythapvu
p dhuaz av bwsvhk aol wyhjapjhs ilmvyl klhkspul
upyth bupclyzpaf

*** Decrypted  text is :

information
i wants to upload the practical before deadline
nirma university
```

- After execution of the program, Encrypted message write in output.txt file



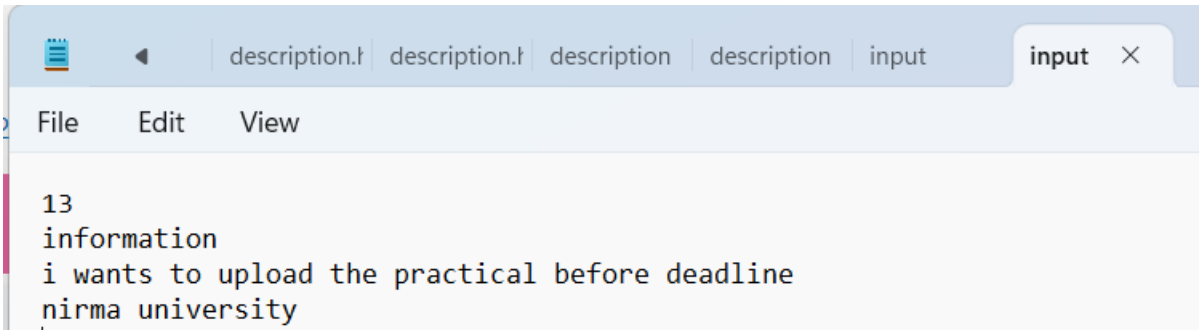
- For decryption ,
  Input from (encrypted message) - output.txt file
  Output (decrypted message) - doutput.txt



Note: this program done both the task -> (1) encryption and (2) decryption.

b) ROT – 13: -we provide key = 13 in program (a)

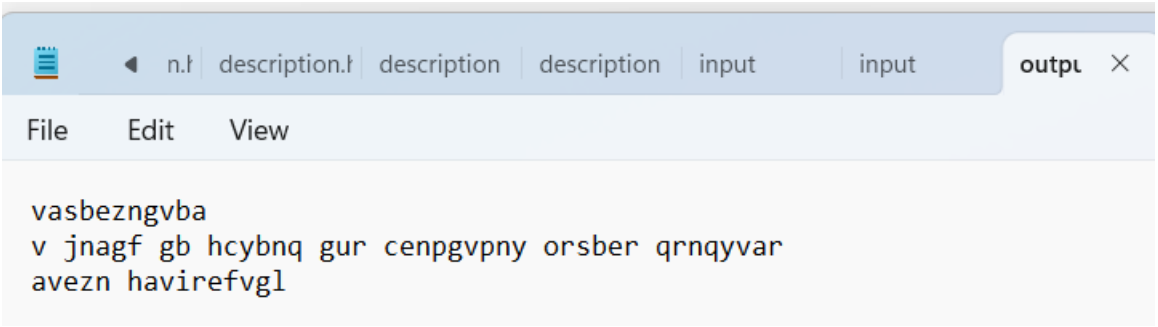
- **METHODOLOGY FOLLOWED:** Same as (a) Ceaser cipher but key = 13.
- **INPUT:**



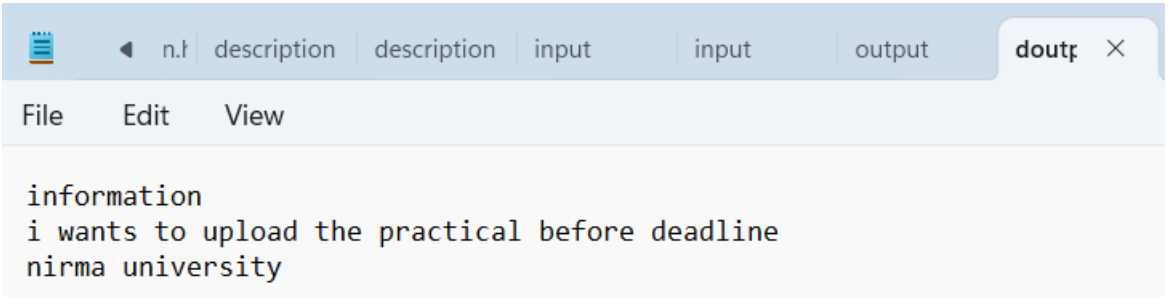
- Output:

```
13

The CAESERKEY between 0 and 25 is:  13

*** The PLAIN TEXT for Encryption is :

information
i wants to upload the practical before deadline
nirma university

*** The Encrypted text is :

vasbezngvba
v jnagf gb hcybnq gur cenpgvpny orsber qrnqyvar
avezn havirefvgl

*** Decrypted  text is :

information
i wants to upload the practical before deadline
nirma university

Process returned 0 (0x0)   execution time : 0.046 s
Press any key to continue.
```

- After execution of the program, Encrypted message write in output.txt file



- For decryption,
  Input from (encrypted message) - output.txt file
  Output (decrypted message) - doutput.txt



Note: this program done both the task -> (1) encryption and (2) decryption.

Question:

1.Crack the following plaintext TRVJRI TZGYVIJ RIV HLZKV VRJP KF TIRTB

```
KEY = 0 and decrypted text : TRVJRI TZGYVIJ RIV HLZKV VRJP KF TIRTB
KEY = 1 and decrypted text : SQUIQH SYFXUHI QHU GKYJU UQIO JE SHQSA
KEY = 2 and decrypted text : RPTHPG RXEWTGH PGT FJXIT TPHN ID RGPRZ
KEY = 3 and decrypted text : QOSGOF QWDVSFG OFS EIWHS SOGM HC QFOQY
KEY = 4 and decrypted text : PNRFNE PVCUREF NER DHVGR RNFL GB PENPX
KEY = 5 and decrypted text : OMQEMD OUBTQDE MDQ CGUFQ QMEK FA ODMOW
KEY = 6 and decrypted text : NLPDLC NTASPCD LCP BFTEP PLDJ EZ NCLNV
KEY = 7 and decrypted text : MKOCKB MSZROBC KBO AESDO OKCI DY MBKMU
KEY = 8 and decrypted text : LJNBJA LRYQNAB JAN ZDRCN NJBH CX LAJLT
KEY = 9 and decrypted text : KIMAIZ KQXPMZA IZM YCQBM MIAG BW KZIKS
KEY = 10 and decrypted text : JHLZHY JPWOLYZ HYL XBPAL LHZF AV JYHJR
KEY = 11 and decrypted text : IGKYGX IOVNKXY GXK WAOZK KGYE ZU IXGIQ
KEY = 12 and decrypted text : HFJXFW HNUMJWX FWJ VZNYJ JFXD YT HWFHP
KEY = 13 and decrypted text : GEIWEV GMTLIVW EVI UYMXI IEWC XS GVEGO
KEY = 14 and decrypted text : FDHVDU FLSKHUV DUH TXLWH HDVB WR FUDFN
KEY = 15 and decrypted text : ECGUCT EKRJGTU CTG SWKVG GCUA VQ ETCEM
KEY = 16 and decrypted text : DBFTBS DJQIFST BSF RVJUF FBTZ UP DSBDL
KEY = 17 and decrypted text : CAESAR CIPHERS ARE QUITE EASY TO CRACK
KEY = 18 and decrypted text : BZDRZQ BHOGDQR ZQD PTHSD DZRX SN BQZBJ
KEY = 19 and decrypted text : AYCQYP AGNFCPQ YPC OSGRC CYQW RM APYAI
KEY = 20 and decrypted text : ZXBPXO ZFMEBOP XOB NRFQB BXPV QL ZOXZH
KEY = 21 and decrypted text : YWAOWN YELDANO WNA MQEPA AWOU PK YNWYG
KEY = 22 and decrypted text : XVZNVM XDKCZMN VMZ LPDOZ ZVNT OJ XMVXF
KEY = 23 and decrypted text : WUYMUL WCJBYLM ULY KOCNY YUMS NI WLUWE
KEY = 24 and decrypted text : VTXLTK VBIAXKL TKX JNBMX XTLR MH VKTVD
KEY = 25 and decrypted text : USWKSJ UAHZWJK SJW IMALW WSKQ LG UJSUC
```

ANS: KEY = 17 , MESSAGE : CAESAR CIPHERS ARE QUITE EASY TO CRACK

2. **What key do we need to make "CAESAR" become "MKOCKB"?**

   ANS: KEY = 10

3. **What key do we need to make "CIPHER" become "SYFXUH"?**
   ANS: KEY = 16

4. **Use the Caesar cipher to encrypt your first name**
   ANS :
   Name: Kartik
   Key: 21
   Encrypted message: fvmodf

5. **How can we find the decryption key from the encryption key?**
   Ans: (decryption key = encryption key) in Caesar cipher

3) Hill cipher: Hill cipher is a polygraphic substitution cipher based on linear algebra.Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1, ..., Z = 25 is

used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.

The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).

Input : Plaintext: ACT

Key: GYBNQKURP

Output : Ciphertext: POH

We have to encrypt the message 'ACT' (n=3).

The key is 'GYBNQKURP' which can be written as the nxn matrix

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}$$

The message 'ACT' is written as vector:

$$\begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix}$$

The enciphered vector is given as: which corresponds to ciphertext of 'POH'

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} = \begin{bmatrix} 67 \\ 222 \\ 319 \end{bmatrix} \equiv \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} \pmod{26}$$

Decryption :

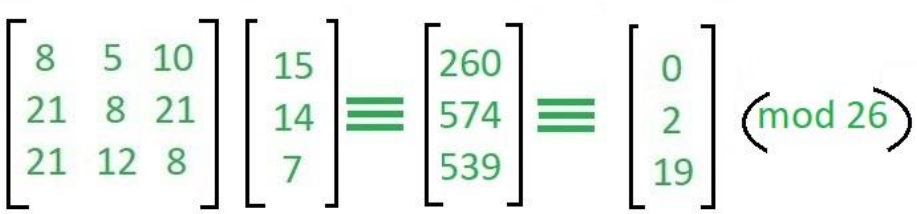
To decrypt the message, we turn the ciphertext back into a vector, then simply multiply by the inverse matrix of the key matrix (IFKVIVVMI in letters).The inverse of the matrix used in the

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}^{-1} \equiv \begin{bmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{bmatrix} \pmod{26}$$

previous example is:

For the previous Ciphertext 'POH': which gives us back 'ACT'.

Assume that all the alphabets are in upper case.



METHODOLOGY FOLLOWED:

NOTE: IN THIS CODE I USE MOD 256 (CHAR SIZE = 1 BYTE , TOTAL 256 DIFFERENT CHARACTERS WE HAVE IT WORKS FOR ALL CHARECTERS)

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

int determinant(int m[3][3])
{
  int d = 0;
  for (int i = 0; i < 3; i++)
  {
    int t[2][2];
    int p = 0;
    int q = 0;
    for (int j = 1; j < 3; j++)
    {

      for (int k = 0; k < 3; k++)
      {
        if (k == i)
          continue;
        t[p][q] = m[j][k];
        q++;
      }
      q = 0;
      p++;
    }

    if (i % 2 == 0)
    {
      d += m[0][i] * (t[0][0] * t[1][1] - t[0][1] * t[1][0]);
    }
    else
    {
      d -= m[0][i] * (t[0][0] * t[1][1] - t[0][1] * t[1][0]);
```

```
    }
  }

  return d;
}

int minor(int matrix[3][3], int r, int c)
{

  int t[2][2];
  int p = 0;
  int q = 0;
  for (int j = 0; j < 3; j++)
  {
    if (j == r)
      continue;
    for (int k = 0; k < 3; k++)
    {
      if (k == c)
        continue;
      t[p][q] = matrix[j][k];
      q++;
    }
    q = 0;
    p++;
  }

  if ((r + c) % 2 == 0)
  {
    return (t[0][0] * t[1][1] - t[0][1] * t[1][0]);
  }
  else
  {
    return -(t[0][0] * t[1][1] - t[0][1] * t[1][0]);
  }
}

void adjacent(int mat[3][3])
{

  int ad[3][3] = {};

  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      // transpose - a[j][i] -> minor(mat,i , j)
      ad[j][i] = minor(mat, i, j);
```

```
    }
  }

  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      mat[i][j] = ad[i][j];
    }
  }
}

int multiplicativeInverse(int D)
{
  // D=D%26;

  for (int i = 1; i <= 256; i++)
  {
    if ((i * D) % 256 == 1)
    {
      return i;
    }
  }
}

void matrix_mod(int m[3][3], int mod)
{
  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      m[i][j] = m[i][j] % mod;
    }
  }
}

// this inverse function not help because it gives inverse
// but type in float hence we missing the information
// hence we not use this inverse function .
void inverse(int m[3][3])
{

  int D = determinant(m);
  adjacent(m);

  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
```

```
    {
      m[i][j] /= D;
    }
  }
}

int mod(int a, int mod)
{
  if (a < 0)
  {
    return ((a % mod) + mod) % mod;
  }
  else
  {
    return a % mod;
  }
}

string decription(int km[3][3], int em[3])
{

  int temp[3][3];

  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      temp[i][j] = km[i][j];
    }
  }

  int D = determinant(temp);
  adjacent(temp);

  // cout<<D<<"\n";
  // cout<<D%26<<"\n";

  D = multiplicativeInverse(D);

  // cout<<" adjacent matrix.....\n";
  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      temp[i][j] = (D * temp[i][j]) % 256;
    }
  }
```

```
  // inverse of temp = 1/D * adjacent matrix of temp;

  // here our task is to maintain   conguerance modulo relation
  // if  a conguerant b ( mod m ) ,means  (a-b)%m =0;
  // if a conguerant b ( mod m )  then  a*k conguerant b*k ( mod m )

  int dm[3] = {};

  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      dm[i] += temp[i][j] * em[j];
    }
  }

  for (int i = 0; i < 3; i++)
  {
    dm[i] = mod(dm[i], 256);
  }

  // cout<<"\n\nDecrypted message is : ";

  string st = "";
  for (int i = 0; i < 3; i++)
  {
    st.push_back(dm[i]);
  }

  return st;
}

string encription(int km[3][3], int tm[3], int em[3])
{

  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      em[i] += km[i][j] * tm[j];
    }
  }

  for (int i = 0; i < 3; i++)
  {
    em[i] %= 256;
  }
```

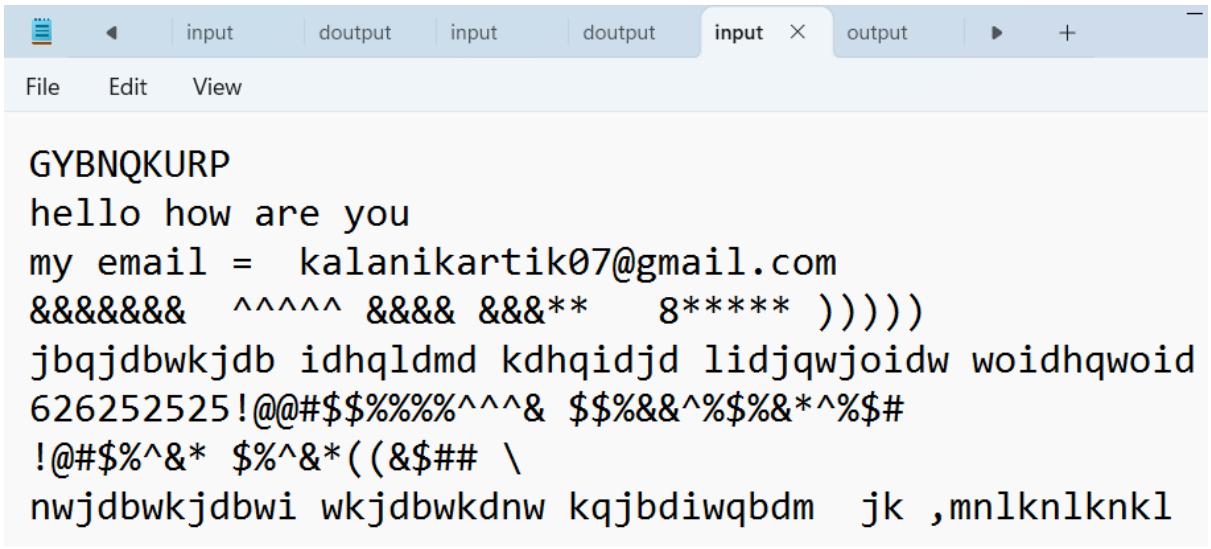
```
  // int d = determinantOfMatrix(matrix,n);

  //  cout<<"\nEncrypted message is : ";

  string st;
  for (int i = 0; i < 3; i++)
  {
    st.push_back(em[i]);
  }

  return st;
}

/* ofstream fout;
 fout.open("output.txt");
 fout.close();
*/

int main()
{

  ifstream fin;
  fin.open("input.txt");

  string keyst;
  getline(fin, keyst);

  cout << "Key: " << keyst << "\n\n";

  int n = 3;
  int km[3][3]; // km - key matrix
  int c = 1;

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

      km[i][j] = keyst[n * i + j];
    }
  }

  string st;

  ofstream fout;
  fout.open("output.txt");

  ofstream fout2;
```

```
  fout2.open("doutput.txt");

  while (getline(fin, st))
  {
    int sz = st.length();
    if (sz % 3 == 1)
    {
      st.append("  ");
      sz += 2;
    }
    else if (sz % 3 == 2)
    {
      st.append(" ");
      sz += 1;
    }

    for (int j = 0; j < sz; j += 3)
    {

      int tm[3]; // tm - text matrix
      for (int i = 0; i < 3; i++)
      {
        tm[i] = st[j + i];
      }

      // matrix multiplication key_matrix*text_matrix;

      int em[3] = {}; // em - encrypted matrix
      string est = encription(km, tm, em);

      fout << est;
      cout << est << " ";

      string s = decription(km, em);

      fout2 << s;
      cout << s << " ";

      cout << "\n";
    }
    fout2 << "\n";
    fout << "\n";
  }

  fout2.close();

  return 0;
}
```

INTPUT: form input.txt file

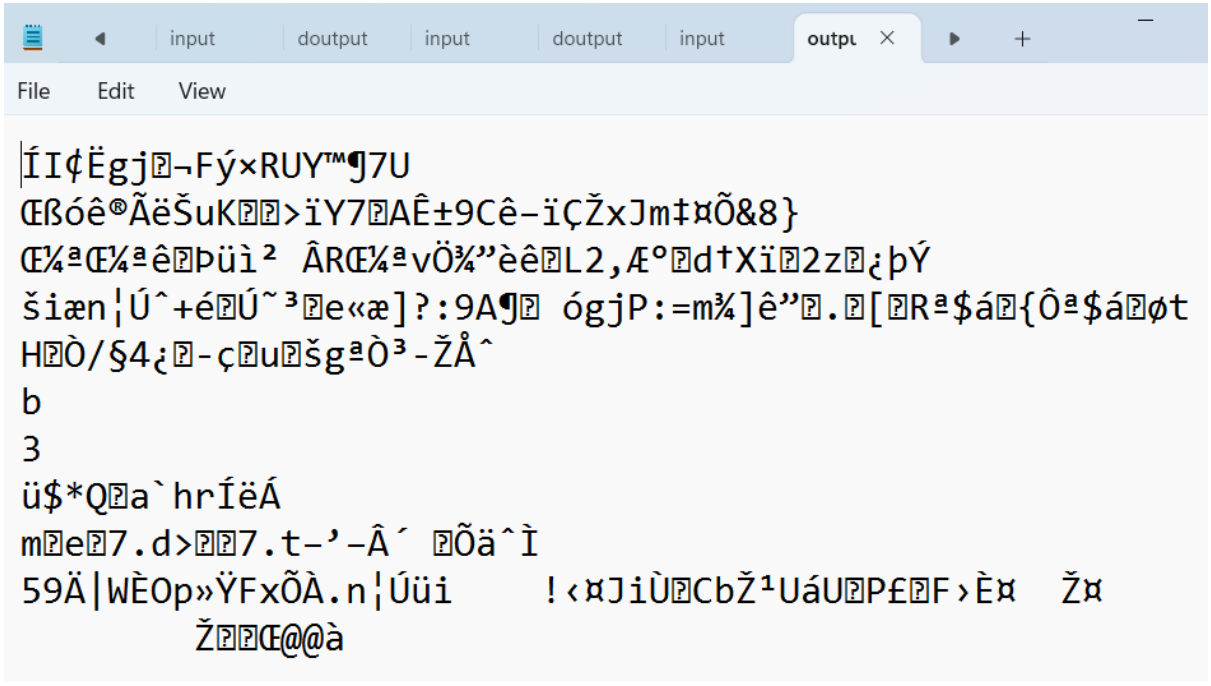
KEY – FIRST LINE OF INPUT.TXT

PLAINTTEX: FROM 2nd to end of file



Here key = GNBNQKURP

OUTPUT1(ENCRYPTION , ENCRYPTED TEXT) : OUTPUT.TXT



DECRYPTION : DOUTPUT.TXT

-WE GET OUR MESSAGE BACK THROUGH DECRYPT THE ENCRIPTED TEXT.