

**CNS LAB**  
**Danapgol Nikhil Rajendra**  
**2019BTECS00036**

**Assignment 1**

**Aim** - To encrypt the given plain text using Caesar Cipher and then decrypt it to get plain text again.

**Theory:**

It is substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet

**Code:**

```
// Implementation of Ceasor Cypher
#include<iostream>
using namespace std;

string encrypt(string plainText,int pos){

    int n=plainText.size();
    for(int i=0;i<n;i++){
        if(plainText[i]==32)
            continue;
        if(plainText[i]>96 && plainText[i]<=122)
            plainText[i]-=32;
        plainText[i]=(plainText[i]-'A'+pos)%26+'A';
    }
    cout<<"\nCipher Text :"<<plainText<<endl;
    return plainText;
}

void decrypt(string cypherText,int pos){
    int n=cypherText.size();
```

```

for(int i=0;i<n;i++){
    if(cypherText[i]==32)
        continue;
    cypherText[i]=(cypherText[i]-'A'-pos+26)%26+'A';
}
cout<<"\nPlain Text :"<<cypherText<<endl;
}

int main(){
    int pos;
    freopen("input.txt", "r", stdin);
    freopen("caesarcipher.txt", "w", stdout);

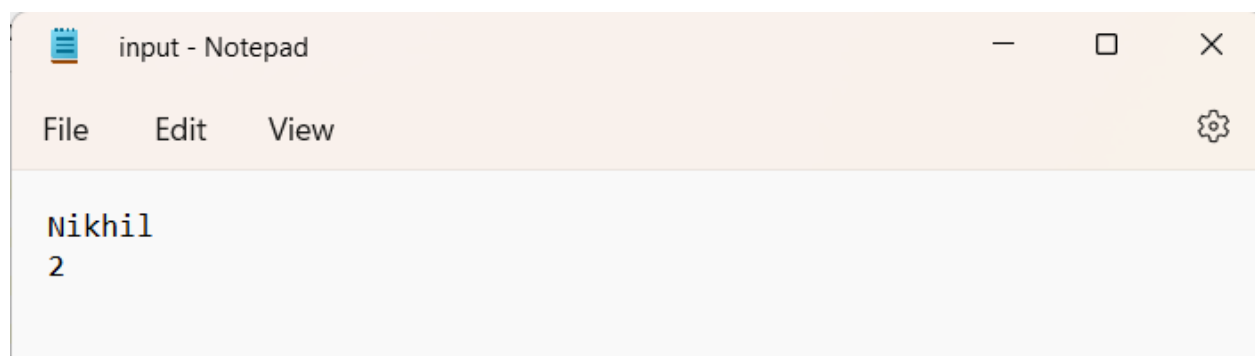
    string plain_text;
    getline(cin,plain_text);

    // cout<<"Plain text:"<<plain_text<<endl;
    cin>>pos;

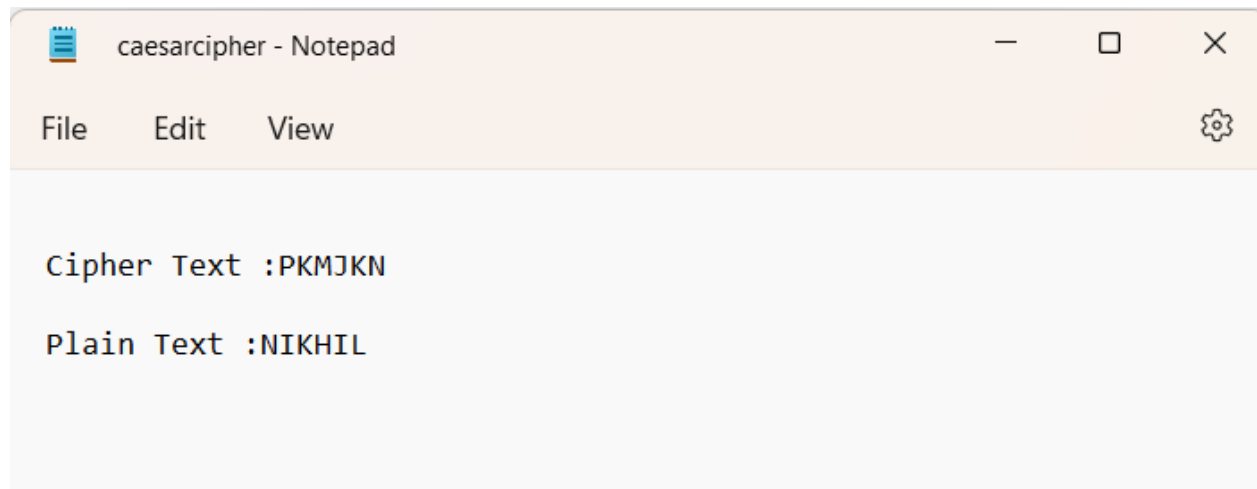
    string cypher_text=encrypt(plain_text,pos);
    decrypt(cypher_text,pos);
}

```

**Input:**



## Output:



**CNS LAB**  
**Danapgol Nikhil Rajendra**  
**2019BTECS00036**

**Assignment 2**

**Aim :**

Given a Cipher text , encrypted caesar using , Using Crypt analysis find the plain text

**Theory:**

Caesar Cipher It is a substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet We will decrypt using all the possible key , and find the most relative plain text

**Code :**

```
set<string> dict;
dict.insert("i");
dict.insert("am");
dict.insert("in");
dict.insert("cns");
dict.insert("lab");
dict.insert("for");
dict.insert("LA");
string s, org;
cout << "Enter Cipher text" << endl;
getline(cin, s);
string x;
int k = 0;
cout << "\nCipher text is: " << s << endl << endl;
org = s;
for (int k = 0; k < 26; k++)
{
```

```

cout << "Keep Key as: " << k << endl;
s = org;
string word = "";
int flg = 0;
for (int i = 0; i < s.length(); i++)
{
    if (s[i] == ' ')
    {
        if (dict.find(word) == dict.end())
        {
            flg = 1;
            break;
        }
        word = "";
        continue;
    }
    int val = s[i] - 'a';
    val = (val - k + 26) % 26;
    char ch = 'a' + val;
    word += ch;
    s[i] = ch;
}
if (dict.find(word) == dict.end())
{
    flg = 1;
}
if (flg == 0)
cout << s << endl << endl;
}

```

Input/Output:

C:\CNSLAB\Assignment\_1\Cryptanalysis\cryptanalysis.exe

Enter Cipher text

bn

Cipher text is: bn

Keep Key as: 0

Keep Key as: 1

am

Keep Key as: 2

Keep Key as: 3

Keep Key as: 4

Keep Key as: 5

Keep Key as: 6

Keep Key as: 7

Keep Key as: 8

Keep Key as: 9

Keep Key as: 10

Keep Key as: 11

C:\CNSLAB\Assignment\_1\Cryptanalysis\cryptanalysis.exe

Enter Cipher text

dot

Cipher text is: dot

Keep Key as: 0

Keep Key as: 1

cns

Keep Key as: 2

Keep Key as: 3

Keep Key as: 4

Keep Key as: 5

Keep Key as: 6

Keep Key as: 7

Keep Key as: 8

Keep Key as: 9

Keep Key as: 10

Keep Key as: 11

Keep Key as: 12

Keep Key as: 13

Keep Key as: 14

Keep Key as: 15

Keep Key as: 16

Keep Key as: 17

**CNS LAB**  
**Danapgol Nikhil Rajendra**  
**2019BTECS00036**

**Assignment 3**

**Aim :**

Given the plain text, encrypt it using Playfair Encryption Algorithm

**Theory :**

Playfair Encryption Algorithm:

- The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.
- The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.

The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter. Pair cannot be made with same letter. Break the letter in single and add a bogus letter to the previous letter. If both the letters are in the same column: Take the letter below each one. If both the letters are in the same row: Take the letter to the right of each one. If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

**Code:**

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



```
map<char,pair<int,int>> charPos;
```

```
// Capitalize the character
```

```
void capitalize(string &str){
```

```
    for(char &c:str){
```

```
        if(c>=97 && c<=122)
```

```
            c-=32;
```

```
    }
```

```
}
```

```
// add char pairs to the set
```

```
void addToSet(vector<pair<char,char>> &v,char a,char b){
```

```
    v.push_back({a,b});
```

```
}
```

```
// build matrix
```

```
void buildMatrix(vector<vector<char>> &mat,string key){
```

```
    vector<bool>vis(26,false);
```

```
    int i=0,j=0;
```

```
    for(int k=0;k<key.length();k++){
```

```
        if(!vis[key[k]-65]){
```

```
            mat[i][j]=key[k];
```

```
            if(key[k]=='I'||key[k]=='J'){
```

```
                vis['I'-65]=true;
```

```
                vis['J'-65]=true;
```

```
            }
```

```
            else vis[key[k]-65]=true;
```

```
            j++;
```

```
            if(j==5){
```

```
                j%=5;
```

```
                i++;
```

```

    }
}

}
for(int k=0;k<26;k++){

    if(!vis[k]){
        if(k+'A'=='I' || k+'A'=='J'){
            vis['I'-65]=true;
            vis['J'-65]=true;
        }
        mat[i][j]=k+'A';
        j++;
        if(j==5){
            j%=5;
            i++;
        }
    }

}

// cout<<"\n \nMatrix:\n";
for(int i=0;i<5;i++){
    for(int j=0;j<5;j++){
        cout<<mat[i][j]<<" ";
    }
    cout<<"\n";
}

}

void encrypt(vector<vector<char>> &mat,vector<pair<char,char>>
&pairSet){

    // storing position of chars in the matrix
    for(int i=0;i<5;i++){
        for(int j=0;j<5;j++)

```

```

    if(mat[i][j]=='I' || mat[i][j]=='J')
        charPos['J']=charPos['I']=make_pair(i,j);
    else
        charPos[mat[i][j]]=make_pair(i,j);
}

// print
cout<<"\n";
// cout<<"Plain Text:";
for(auto it:pairSet)
    cout<<it.first<<it.second<<" ";
cout<<endl;

// cout<<"Cypher Text: ";
for(auto &it:pairSet){
    int i1=charPos[it.first].first;
    int j1=charPos[it.first].second;
    int i2=charPos[it.second].first;
    int j2=charPos[it.second].second;

    if(i1==i2){
        it.first=mat[i1][(j1+1)%5];
        it.second=mat[i1][(j2+1)%5];
        cout<<mat[i1][(j1+1)%5]<<mat[i1][(j2+1)%5]<<" ";
    }
    else if(j1==j2){
        it.first=mat[(i1+1)%5][j1];
        it.second=mat[(i2+1)%5][j2];
        cout<<mat[(i1+1)%5][j1]<<mat[(i2+1)%5][j2]<<" ";
    }
    else{
        it.first=mat[i1][j2];
        it.second=mat[i2][j1];
        cout<<mat[i1][j2]<<mat[i2][j1]<<" ";
    }
}

```

```

    }
    cout<<"\n\n";
}

```

```

void decrypt(vector<vector<char>> &mat,vector<pair<char,char>>
&pairSet){

```

```

    // print
    // for(auto it:charPos)
    //  cout<<it.first<<" "<<it.second.first<<" "<<it.second.second<<endl;
    // cout<<endl;

```

```

    // cout<<"Plain Text: ";
    for(auto it:pairSet){
        int i1=charPos[it.first].first;
        int j1=charPos[it.first].second;
        int i2=charPos[it.second].first;
        int j2=charPos[it.second].second;

        // cout<<i1<<j1<<" "<<i2<<j2<<endl;

```

```

        if(i1==i2){
            cout<<mat[i1][((j1-1)+5)%5]<<mat[i1][((j2-1)+5)%5]<<" ";
        }
        else if(j1==j2){
            cout<<mat[((i1-1)+5)%5][j1]<<mat[((i2-1)+5)%5][j2]<<" ";
        }
        else{
            cout<<mat[i1][j2]<<mat[i2][j1]<<" ";
        }

```

```

    }
}

```

```

int main(){

    freopen("playfairinput.txt", "r", stdin);
    freopen("playfaircipher.txt", "w", stdout);


    string tmp,key,plainText;
    vector<pair<char,char>> pairSet;


    getline(cin,tmp);

    for(char c:tmp){
        if(c!=32)
            plainText.push_back(c);
    }

    // cout<<plainText<<endl;
    capitalize(plainText);

    getline(cin,key);

    capitalize(key);

    // cout<<key<<endl;

    int n=plainText.size();
    for(int i=0;i<n;){
        char a=plainText[i];
        if(i==n-1 || plainText[i+1]==a){
            addToSet(pairSet,a,'X'); // added if x as dummy for same chars
            i++;
        }
        else{
            addToSet(pairSet,a,plainText[i+1]);
            i+=2;
        }
    }
}

```

```

    }
}
// breaking the plain text into 2chars
// for(auto it:pairSet)
//  cout<<it.first<<it.second<<" ";

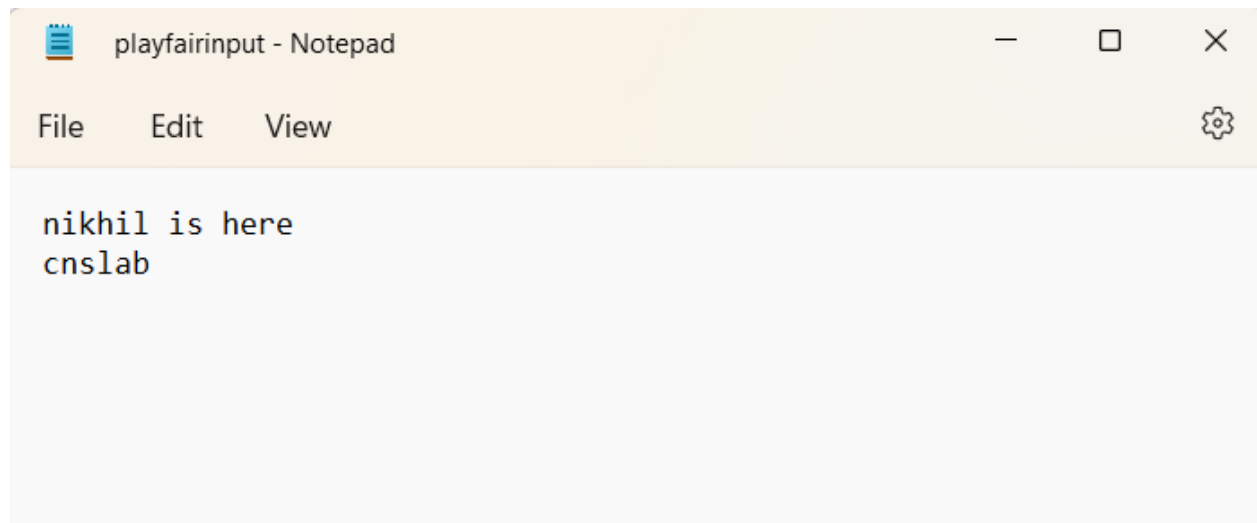
// build matrix
vector<vector<char>> mat(5,vector<char>(5));
buildMatrix(mat,key);

// Encrypt the text
// cout<<"Encryption\n";
encrypt(mat,pairSet);

// Decrypt the Cypher Text
// cout<<"Decryption\n";
decrypt(mat,pairSet);
}

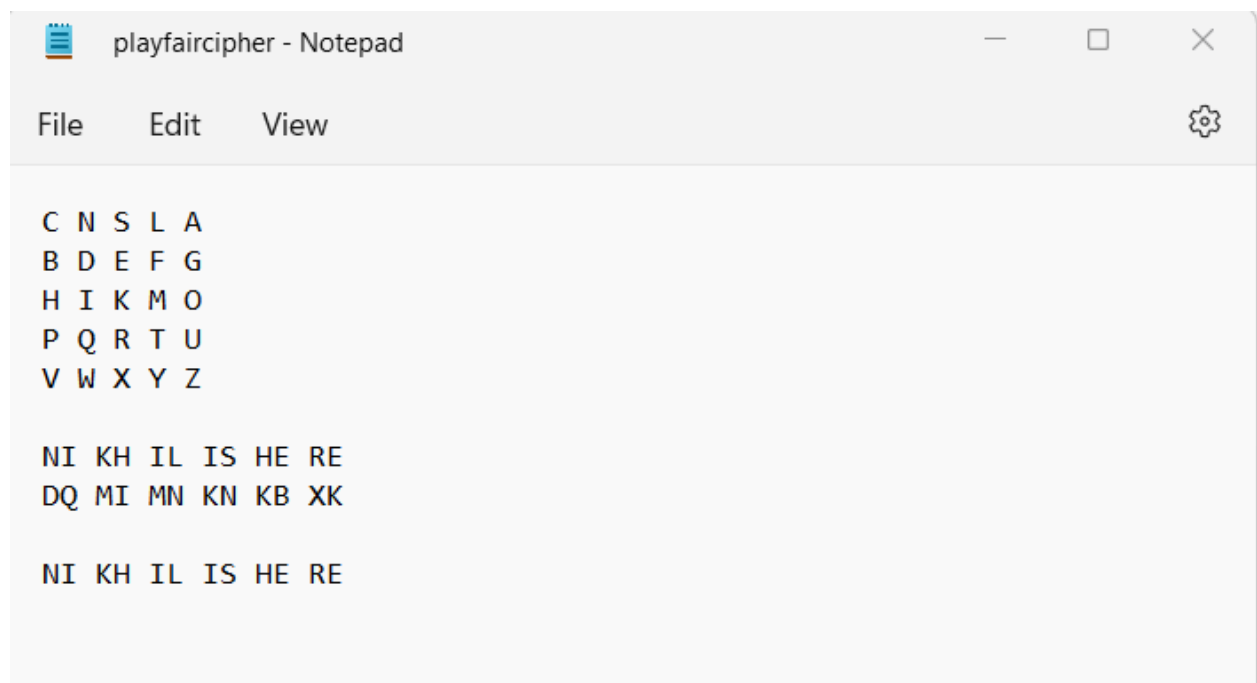
```

**Input :**



```
playfairinput - Notepad
File Edit View
nikhil is here
cnslab
```

## Output:

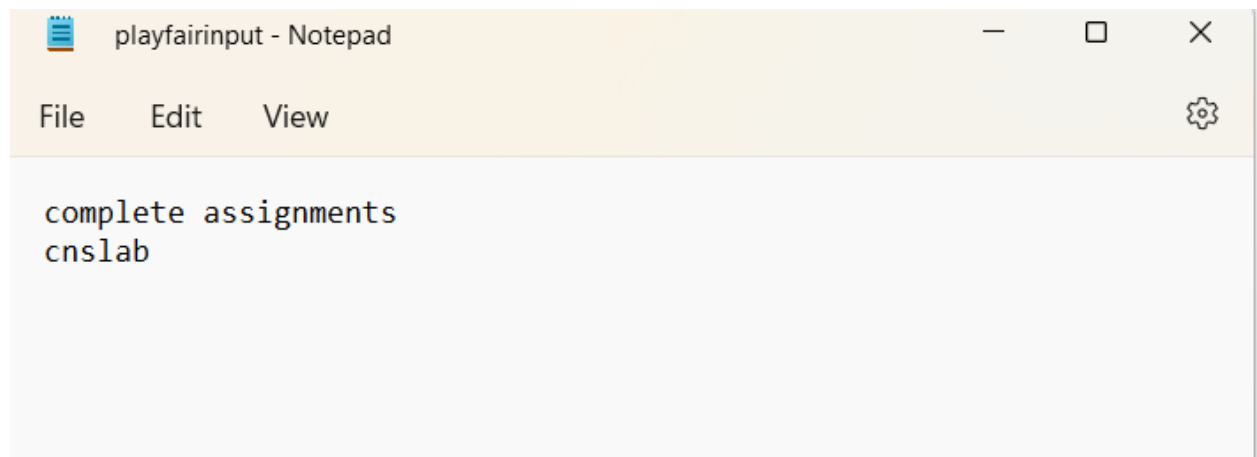


```
playfaircipher - Notepad
File Edit View
C N S L A
B D E F G
H I K M O
P Q R T U
V W X Y Z

NI KH IL IS HE RE
DQ MI MN KN KB XK

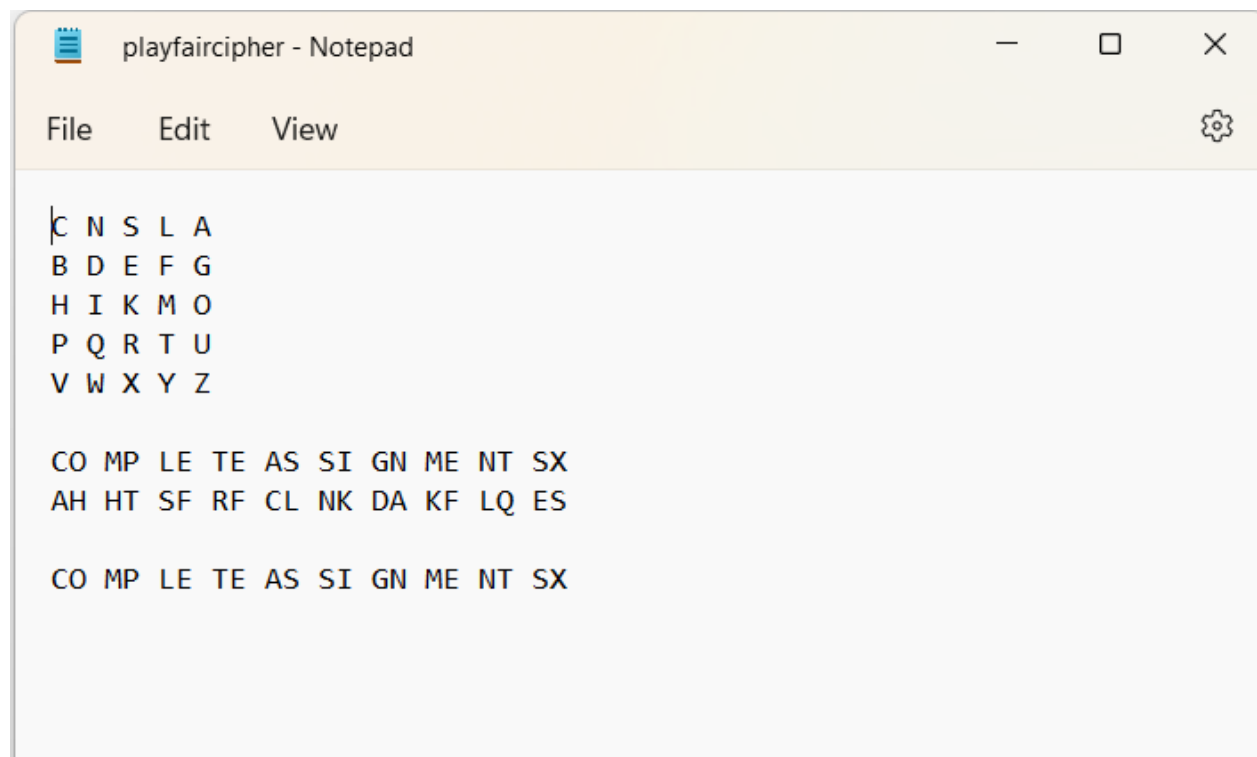
NI KH IL IS HE RE
```

## Input :



```
complete assignments
cnslab
```

## Output :



```
C N S L A
B D E F G
H I K M O
P Q R T U
V W X Y Z

CO MP LE TE AS SI GN ME NT SX
AH HT SF RF CL NK DA KF LQ ES

CO MP LE TE AS SI GN ME NT SX
```



**CNS LAB**  
**Danapgol Nikhil Rajendra**  
**2019BTECS00036**

**Assignment 4**

**Aim :**

**Given the plain text, encrypt it using Vigenere Encryption Algorithm.**

**Theory:**

**Vigenere Cipher Encryption Algorithm :**

It uses a simple form of polyalphabetic cipher In this cipher we add the respective character of a key in the plain text and substitute the character.

**Code :**

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

// Capitalize the character
void capitalize(string &str){
    for(char &c:str){
        if(c>=97 && c<=122)
            c-=32;
    }
}

string encrypt(string &plainText,string &key){
    int n=key.size();
    int i=0;
```

```

for(char &c:plainText){
    if(c>=65 && c<=90){
        int a=c-65;
        int b=key[i%n]-65;
        c=((a+b)%26+65);
        i++;
    }
}
return plainText;
}

```

```

string decrypt(string &cypherText,string &key){

```

```

    int n=key.size();
    int i=0;
    for(char &c:cypherText){
        if(c>=65 && c<=90){
            int a=c-65;
            int b=key[i%n]-65;
            c=(a-b+26)%26+65;
            i++;
        }
    }
    return cypherText;
}

```

```

int main(){

```

```

    freopen("vigenereinput.txt", "r", stdin);
    freopen("vigenereoutput.txt", "w", stdout);

```

```

    string key,plainText;
    getline(cin,plainText);

```

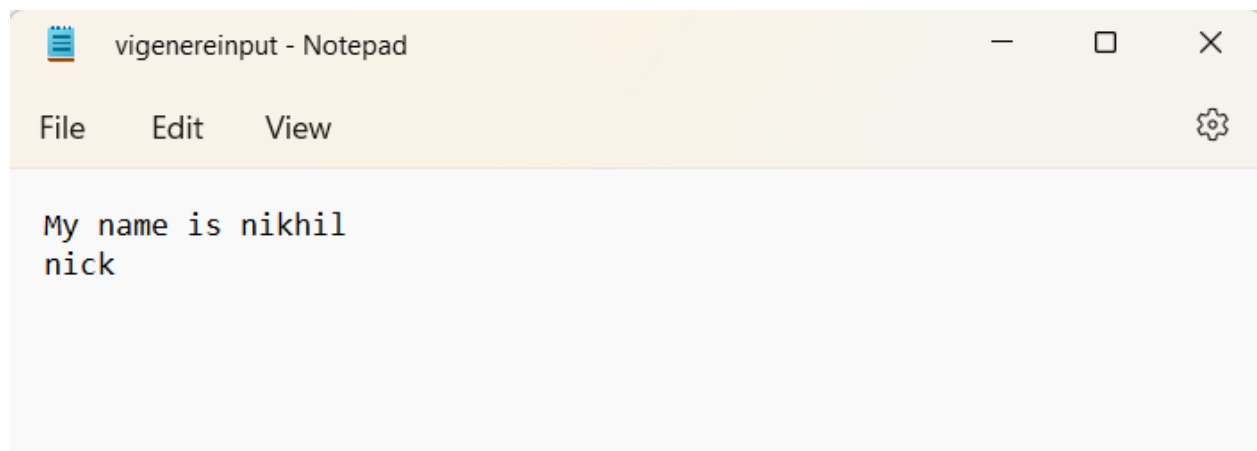
```
// cout<<plainText<<endl;
capitalize(plainText);
getline(cin,key);
capitalize(key);

string CypherText=encrypt(plainText,key);
cout<<"Cypher Text:"<<CypherText<<"\n\n";

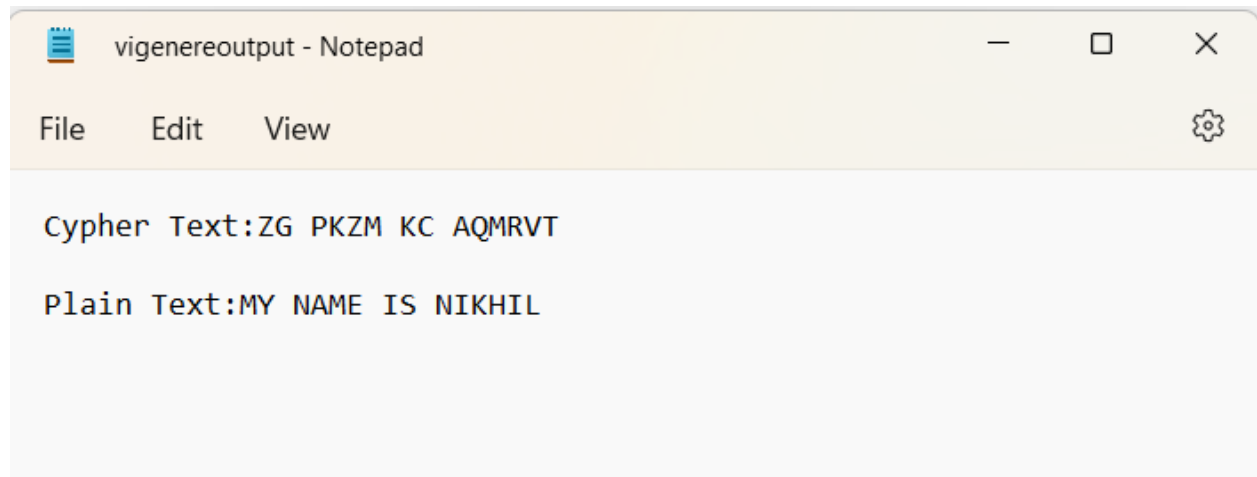
plainText=decrypt(CypherText,key);

cout<<"Plain Text:"<<plainText<<endl;
return 0;
}
```

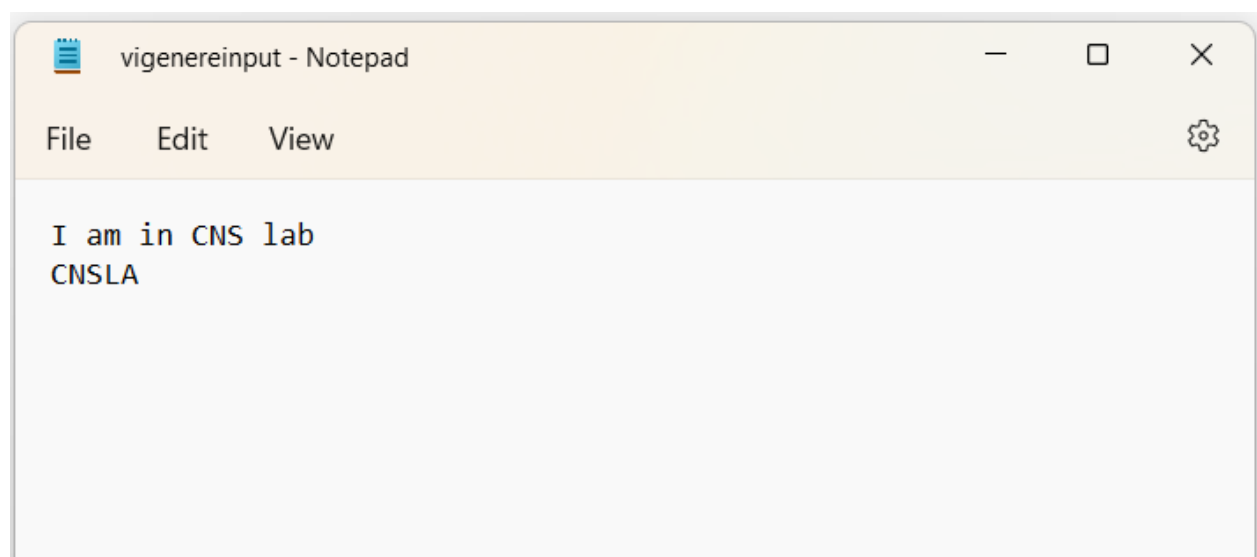
**Input :**



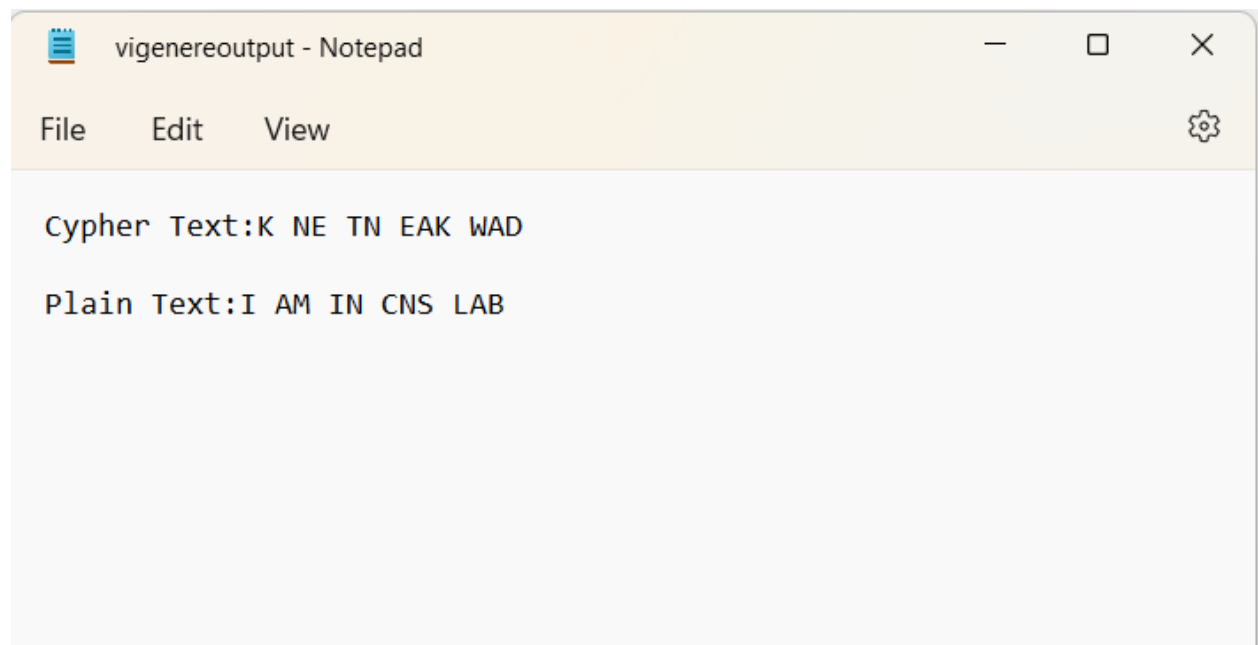
**Output :**



**Input :**



**Output :**



**CNS LAB**  
**Danapgol Nikhil Rajendra**  
**2019BTECS00036**

**Assignment 5**

**Aim :**

Given the plain text, encrypt it using Rail fence Encryption Algorithm

**Theory:**

**Rail fence Cipher Encryption Algorithm :**

- In the rail fence cipher, the plain-text is written downwards and diagonally on successive rails of an imaginary fence.
- When we reach the bottom rail, we traverse upwards moving diagonally, after reaching the top rail, the direction is changed again. Thus the alphabets of the message are written in a zig-zag manner.
- After each alphabet has been written, the individual rows are combined to obtain the cipher-text

**Code:**

```
string s;  
cout << "Enter plain text" << endl;  
getline(cin, s);  
string x;  
for (int i = 0; i < s.length(); i++)  
    if (s[i] != ' ')  
        x += s[i];  
s = x;  
int k;  
cout << "Enter key" << endl;  
cin >> k;
```

```

cout << "\nPlain text is: " << s << endl;
cout << "Key is: " << k << endl;
int n = s.length();
vector<vector<char>> mat(k);
int row = 0;
int flg = 1;
for (int i = 0; i < s.length(); i++)
{
    mat[row].push_back(s[i]);
    row += flg;
    if (row == (k - 1))
    {
        flg = -1;
    }
    if (row == 0)
        flg = 1;
}
string cip = "";
for (int i = 0; i < k; i++)
{
    for (int j = 0; j < mat[i].size(); j++)
        cip += mat[i][j];
}
s = cip;
transform(cip.begin(), cip.end(), cip.begin(), ::toupper);
cout << "\nCipher text is: " << cip;
int tp = 1;
vector<vector<int>> matd(k);
row = 0;
flg = 1;
for (int i = 1; i <= n; i++)
{
    matd[row].push_back(i);
    row += flg;
    if (row == (k - 1))

```

```

{
    flg = -1;
}
if (row == 0)
    flg = 1;
}
vector<int> dd;
for (int i = 0; i < k; i++)
{
    for (int j = 0; j < mat[i].size(); j++)
        dd.push_back(matd[i][j]);
}
cout << endl;
map<int, char> m;
for (int i = 0; i < n; i++)
    m[dd[i]] = s[i];
string plain = "";
for (int i = 1; i <= n; i++)
    plain += m[i];
cout << "\n\nPlain text after decryption is: " << plain;

```

**Input/Output:**



```
"C:\CNSLAB\Assignment_1\Transposition Cipher\railfence.exe"
Enter plain text
nikhil danapol this side
Enter key
3

Plain text is: nikhildanapolthisside
Key is: 3

Cipher text is: NINOIDIHLAAGLHSIEKDPTS

Plain text after decryption is: nikhildanapolthisside
Process returned 0 (0x0)   execution time : 9.509 s
Press any key to continue.
```

```
"C:\CNSLAB\Assignment_1\Transposition Cipher\railfence.exe"
Enter plain text
I am in CNS LAB
Enter key
5

Plain text is: IaminCNSLAB
Key is: 5

Cipher text is: ILASAMNBICN

Plain text after decryption is: IaminCNSLAB
Process returned 0 (0x0)   execution time : 7.649 s
Press any key to continue.
```

**CNS LAB**  
**Danapgol Nikhil Rajendra**  
**2019BTECS00036**

**Assignment 6**

**Aim :**

Given the plain text, encrypt it using Columnar Encryption Algorithm

**Theory:**

**Columnar Cipher Encryption Algorithm:**

In a transposition cipher, the order of the alphabets is re-arranged to obtain the cipher-text.

1. The message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order.
2. Width of the rows and the permutation of the columns are usually defined by a keyword.
3. For example, the word HACK is of length 4 (so the rows are of length 4), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "3 1 2 4".
4. Any spare spaces are filled with nulls or left blank or placed by a character (Example: \_).
5. Finally, the message is read off in columns, in the order specified by the keyword

**Code :**

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

int main()
{
```

```

string s;
cout << "Enter plain text" << endl;
getline(cin, s);
string x;
for (int i = 0; i < s.length(); i++)
if (s[i] != ' ')
x += s[i];
s = x;
int kSize;
cout << "Enter key size" << endl;
cin >> kSize;
vector<int> k(kSize);
int n = s.size();
for (int i = 0; i < kSize; i++)
cin >> k[i];
cout << "\nPlain text is: " << s << endl;
vector<vector<char>> mat(kSize + 1);
int row = 0;
for (int i = 0; i < s.length(); i++)
{
mat[k[row++]].push_back(s[i]);
row = row % kSize;
}
string cipher = "";
for (int i = 0; i <= kSize; i++)
for (int j = 0; j < mat[i].size(); j++)
cipher += mat[i][j];
cout << "\nCipher text is: " << cipher;
return 0;

}

```

## Input/Output:

```
Select "C:\CNSLAB\Assignment_1\Columnar Transposition\columnar.exe"
Enter plain text
nikhil ddanapgol is here
Enter key size
5
4
5
3
2
1

Plain text is: nikhilddanapgolishere

Cipher text is: inlrhaoekdghnlaieidps
Process returned 0 (0x0)   execution time : 14.332 s
Press any key to continue.
```

```
"C:\CNSLAB\Assignment_1\Columnar Transposition\columnar.exe"
Enter plain text
I am in CNS lab
Enter key size
3
3
1
2

Plain text is: IaminCNSlab

Cipher text is: anSbmCIIiNa
Process returned 0 (0x0)   execution time : 13.041 s
Press any key to continue.
```