INFORMATION SECURITY

PRACTICALS

1. **Implement the error correcting code.**

   CODE -

```
#include<iostream>

using namespace std;

int main() {
   int data[10];
   int dataatrec[10],c,c1,c2,c3,i;

   cout<<"Enter 4 bits of data one by one\n";
   cin>>data[0];
   cin>>data[1];
   cin>>data[2];
   cin>>data[4];

   //Calculation of even parity
   data[6]=data[0]^data[2]^data[4];
data[5]=data[0]^data[1]^data[4];
data[3]=data[0]^data[1]^data[2];

cout<<"\nEncoded data is\n";
for(i=0;i<7;i++)
      cout<<data[i];
```

```
cout<<"\n\nEnter received data bits one by one\n";
   for(i=0;i<7;i++)
      cin>>dataatrec[i];

   c1=dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];
c2=dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];
c3=dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0];
c=c3*4+c2*2+c1 ;

   if(c==0) {
cout<<"\nNo error while transmission of data\n";
   }
else {
cout<<"\nError on position "<<c;
cout<<"\nData sent : ";
for(i=0;i<7;i++)
       cout<<data[i];

cout<<"\nData received : ";
      for(i=0;i<7;i++)
       cout<<dataatrec[i];

cout<<"\nCorrect message is\n";
      if(dataatrec[7-c]==0)
dataatrec[7-c]=1;
```

```
    else
dataatrec[7-c]=0;
for (i=0;i<7;i++) {
cout<<dataatrec[i];
}
}
return 0;
}
```

OUTPUT -

2. Implement the error detecting code.

CODE –

2a_Error_Detect_checksum

```
#include<iostream>
#include<string.h>

using namespace std;

int main()
{
   char a[20],b[20];
   char sum[20],complement[20];
   int i;

cout<<"Enter first binary string\n";
   cin>>a;
   cout<<"Enter second binary string\n";
   cin>>b;

if(strlen(a)==strlen(b))
   {
      char carry='0';
      int length=strlen(a);

for(i=length-1;i>=0;i--)
      {
         if(a[i]=='0' && b[i]=='0' && carry=='0')
         {
            sum[i]='0';
            carry='0';
         }
         else if(a[i]=='0' && b[i]=='0' && carry=='1')
         {
            sum[i]='1';
            carry='0';
```

```
}
else if(a[i]=='0' && b[i]=='1' && carry=='0')
{
    sum[i]='1';
    carry='0';

}
else if(a[i]=='0' && b[i]=='1' && carry=='1')
{
    sum[i]='0';
    carry='1';

}
else if(a[i]=='1' && b[i]=='0' && carry=='0')
{
    sum[i]='1';
    carry='0';
     }
else if(a[i]=='1' && b[i]=='0' && carry=='1')
{
    sum[i]='0';
    carry='1';

}
else if(a[i]=='1' && b[i]=='1' && carry=='0')
{
        sum[i]='0';
        carry='1';

      }
      else if(a[i]=='1' && b[i]=='1' && carry=='1')
      {
        sum[i]='1';
        carry='1';

      }
      else
        break;
    }
    cout<<"\nSum="<<carry<<sum;

    for(i=0;i<length;i++)
    {
      if(sum[i]=='0')
        complement[i]='1';
      else
        complement[i]='0';
    }

if(carry=='1')
      carry='0';
    else
```

```
          carry='1';

   cout<<"\nChecksum="<<carry<<complement;
   }
   else
      cout<<"\nWrong input strings";

   return 0;
}
```

OUTPUT –

2b_Error_Detect_parity

CODE -

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

// Function to get parity of number n. It returns 1
// if n has odd parity, and returns 0 if n has even
// parity
bool getParity(unsigned int n)
{
    bool parity = 0;
    while (n)
    {
        parity = !parity;
        n     = n & (n - 1);
    }
    return parity;
}

/* Driver program to test getParity() */
int main()
{
    unsigned int n ;
    cout<<"enter the bits";
    cin>>n;
    cout<<"Parity of no "<<n<<" = "<<(getParity(n)? "odd": "even");

    getchar();
    return 0;
}
```

OUTPUT –

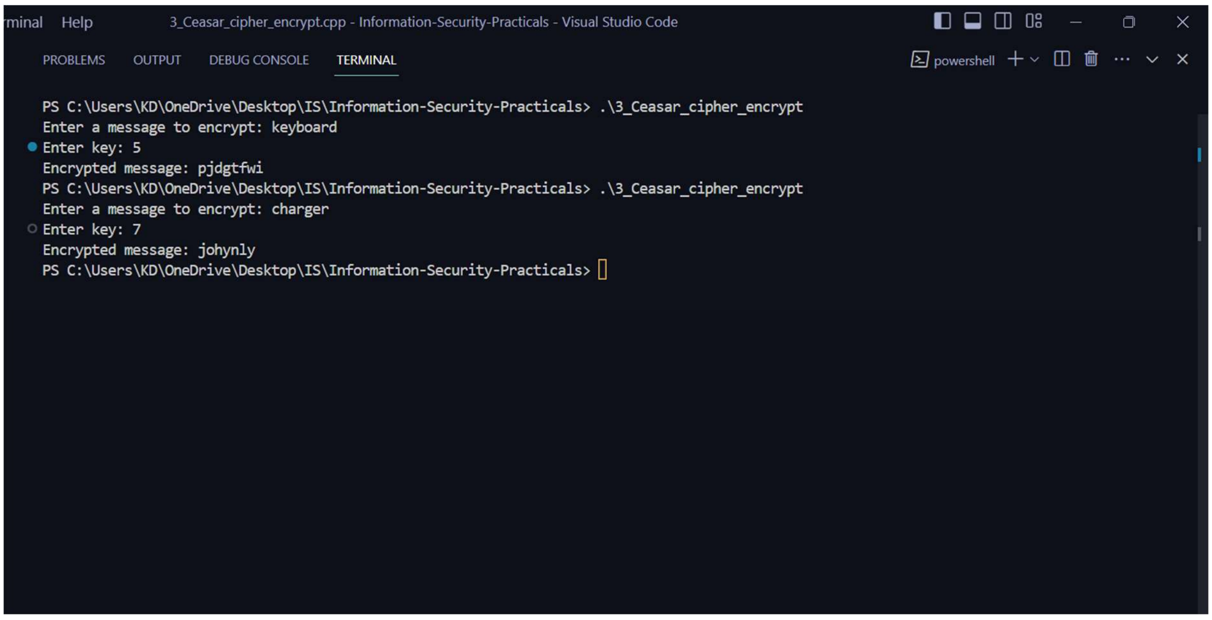
3. Implement caeser cipher substitution operation.

CODE –

3_Ceasar_cipher_encrypt

```
#include<iostream>

using namespace std;

int main()
{
char message[100], ch;
int i, key;
cout << "Enter a message to encrypt: ";
cin.getline(message, 100);
cout << "Enter key: ";
cin >> key;
for(i = 0; message[i] != '\0'; ++i){
ch = message[i];
if(ch >= 'a' && ch <= 'z'){
ch = ch + key;
if(ch > 'z'){
ch = ch - 'z' + 'a' - 1;
}
message[i] = ch;
}
else if(ch >= 'A' && ch <= 'Z'){
ch = ch + key;
if(ch > 'Z'){
ch = ch - 'Z' + 'A' - 1;
}
message[i] = ch;
}

}

cout << "Encrypted message: " << message;

return 0;

}
```

OUTPUT –



rminal Help 3_Ceasar_cipher_encrypt.cpp - Information-Security-Practicals - Visual Studio Code
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
powershell

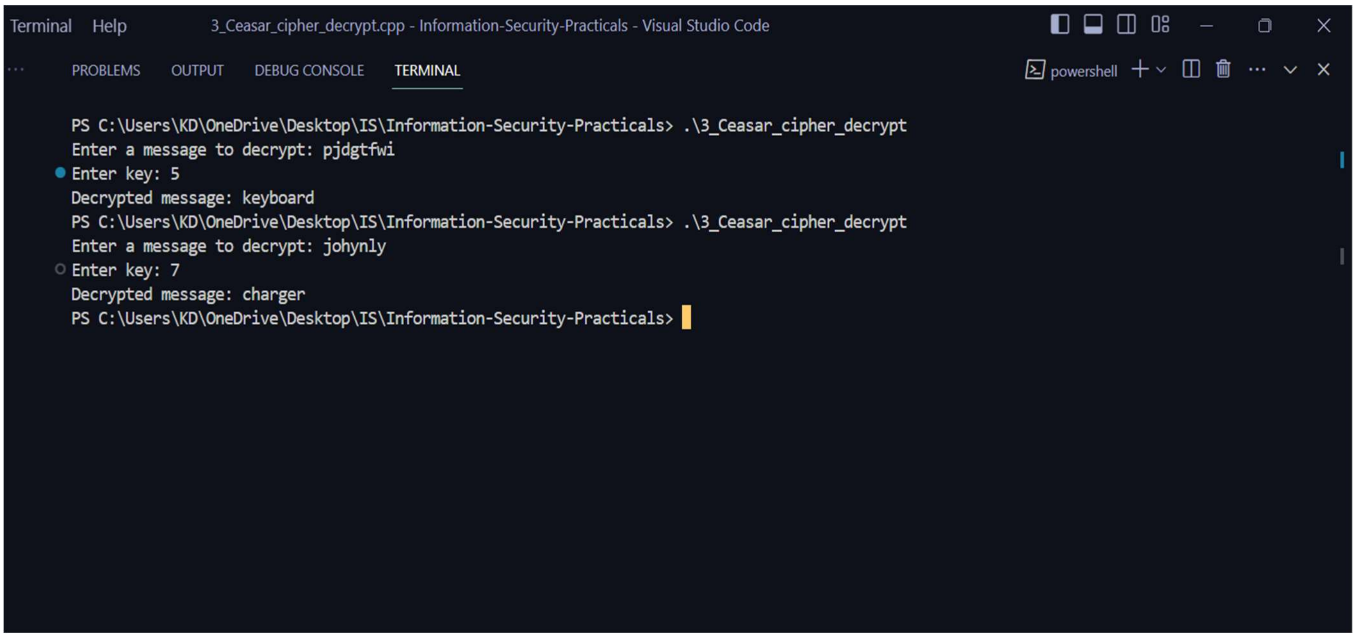
```
PS C:\Users\KD\OneDrive\Desktop\IS\Information-Security-Practicals> .\3_Ceasar_cipher_encrypt
Enter a message to encrypt: keyboard
Enter key: 5
Encrypted message: pjdgtfwi
PS C:\Users\KD\OneDrive\Desktop\IS\Information-Security-Practicals> .\3_Ceasar_cipher_encrypt
Enter a message to encrypt: charger
Enter key: 7
Encrypted message: johynly
PS C:\Users\KD\OneDrive\Desktop\IS\Information-Security-Practicals> 
```

3. Implement caeser cipher substitution operation.

3_Ceasar_cipher_decrypt

```
#include<iostream>


using namespace std;


int main()

{

char message[100], ch;

int i, key;
cout << "Enter a message to decrypt: ";
cin.getline(message, 100);
cout << "Enter key: ";
cin >> key;
for(i = 0; message[i] != '\0'; ++i){
ch = message[i];
if(ch >= 'a' && ch <= 'z'){
ch = ch - key;
if(ch < 'a'){
ch = ch + 'z' - 'a' + 1;
}
message[i] = ch;
}
else if(ch >= 'A' && ch <= 'Z'){
ch = ch - key;
if(ch > 'a'){
ch = ch + 'Z' - 'A' + 1;
}
message[i] = ch;
}
}
cout << "Decrypted message: " << message;
return 0;
}
```

OUTPUT –


Terminal Help 3_Ceasar_cipher_decrypt.cpp - Information-Security-Practicals - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
powershell

```
PS C:\Users\KD\OneDrive\Desktop\IS\Information-Security-Practicals> .\3_Ceasar_cipher_decrypt
Enter a message to decrypt: pjdgtfwi
Enter key: 5
Decrypted message: keyboard
PS C:\Users\KD\OneDrive\Desktop\IS\Information-Security-Practicals> .\3_Ceasar_cipher_decrypt
Enter a message to decrypt: johynly
Enter key: 7
Decrypted message: charger
PS C:\Users\KD\OneDrive\Desktop\IS\Information-Security-Practicals>
```

4. Implement monoalphabetic and polyalphabetic cipher substitution operation.

Monoalphabetic_cipher

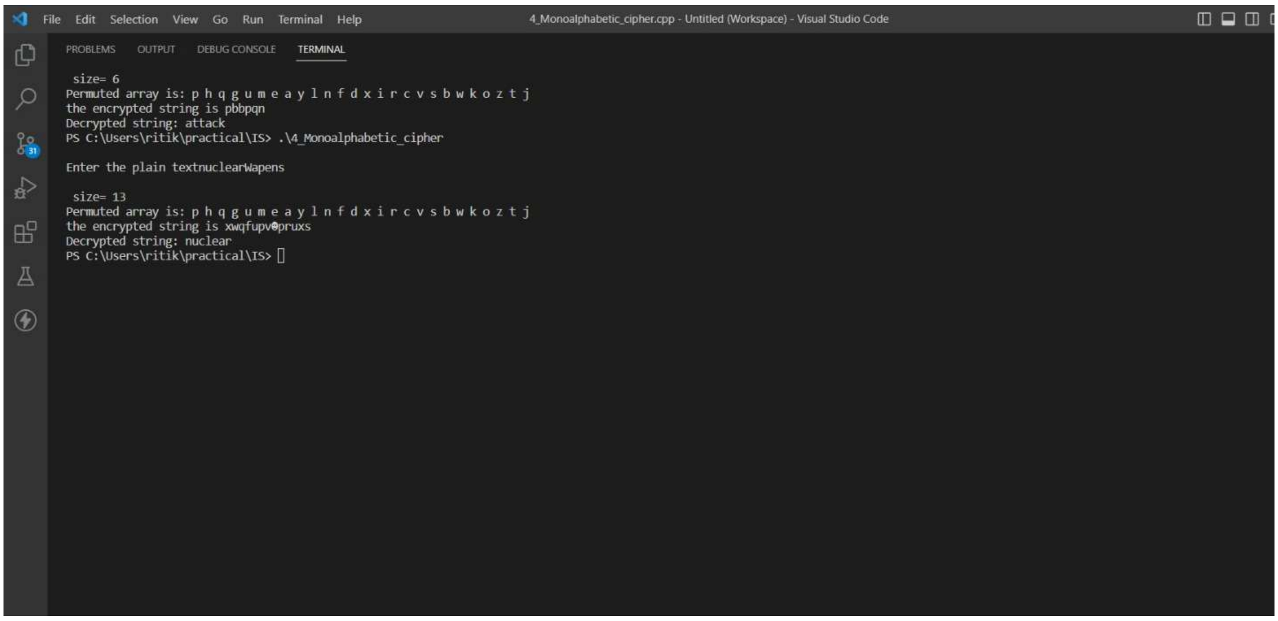
CODE-

```
#include<stdio.h>


#include<conio.h>
#include<stdlib.h>
#include<string.h>


main()
{


char plain[200],encr[200],encr1[200],decr[200];
```

```
int ch,flag[26];
char ch1;
char per[26];
int i,j,size,k=0,a=0;
char pos;

printf("\nEnter the plain text");
gets(plain);
size=strlen(plain);
printf("\n size= %d",size);
 for(i=0;i<26;i++)
  flag[i]=0;

 for(i=0;i<26;i++)
 {
   a=rand()%26;
  while(flag[a]==1)
   a=rand()%26;
  flag[a]=1;
  per[i]=(char)(a+97);
 }
printf("\nPermuted array is: ");
for(i=0;i<26;i++)
{
printf("%c ",per[i]);
}
for(j=0;j<size;j++)
{
ch=plain[j];
ch=ch-97;
ch1=per[ch];
```

```
encr1[j]=ch1;
}

//encr1[25]='\0';
//print

printf("\nthe encrypted string is ");
for(i=0;i<size;i++)
{
printf("%c",encr1[i]);
}



//decryption
 i=0;
 while(i<size)
 {
  //a=ct[i];
  //a=a-97;
  //pt[i]=key[a];
  a=0;
  while(a!=26)
  {
   if(per[a]==encr1[i])
   {
    decr[i]=a+97;
    break;
   }
   a++;
  }
  i++;
 }
 decr[i]='\0';
 printf("\nDecrypted string: %s\n",decr);
 getch();
}
```

OUTPUT –


File Edit Selection View Go Run Terminal Help
4_Monoalphabetic_cipher.cpp - Untitled (Workspace) - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

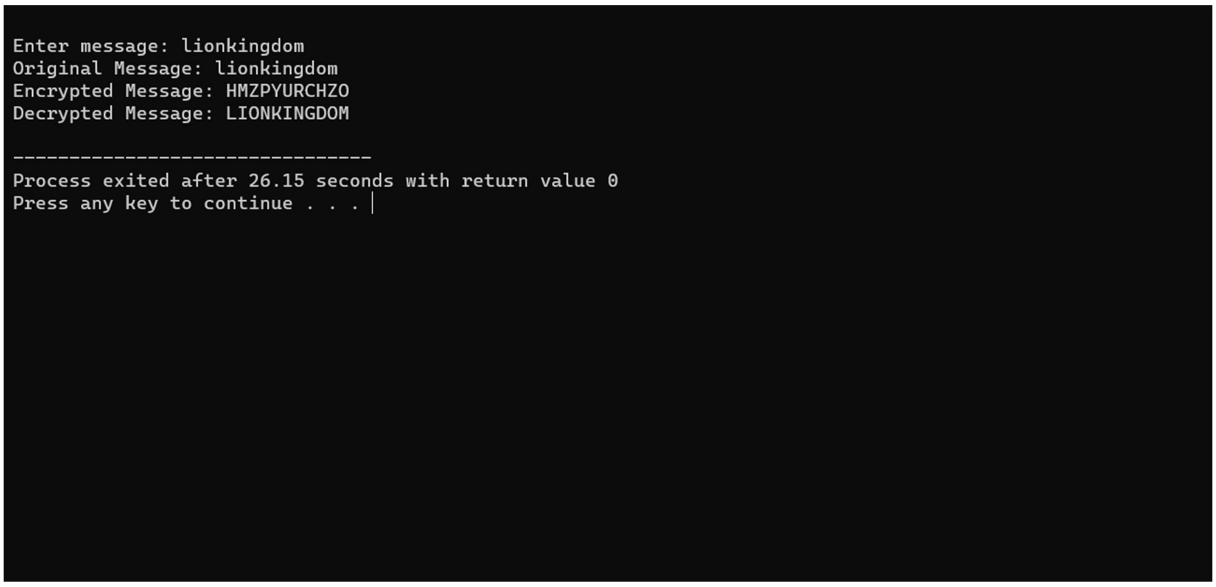
```
 size= 6
Permuted array is: p h q g u m e a y l n f d x i r c v s b w k o z t j
the encrypted string is pbbpqn
Decrypted string: attack
PS C:\Users\ritik\practical\IS> .\4_Monoalphabetic_cipher

Enter the plain textnuclearWapens

 size= 13
Permuted array is: p h q g u m e a y l n f d x i r c v s b w k o z t j
the encrypted string is xwqfupv●pruxs
Decrypted string: nuclear
PS C:\Users\ritik\practical\IS> 
```

Ployalphabetic_cipher

CODE –

```
#include <iostream>
#include <string>
using namespace std;
class Vig {
  public:
     string k;
  Vig(string k) {
     for (int i = 0; i < k.size(); ++i) {
        if (k[i] >= 'A' && k[i] <= 'Z')
           this->k += k[i];
        else if (k[i] >= 'a' && k[i] <= 'z')
           this->k += k[i] + 'A' - 'a';
     }
  }
  string encryption(string t) {
     string output;
     for (int i = 0, j = 0; i < t.length(); ++i) {
        char c = t[i];
        if (c >= 'a' && c <= 'z')
           c += 'A' - 'a';
        else if (c < 'A' || c > 'Z')
           continue;
        output += (c + k[j] - 2 * 'A') % 26 + 'A'; //added 'A' to bring it in range of ASCII alphabet [ 65-90 |
A-Z ]
        j = (j + 1) % k.length();
     }
     return output;
  }
  string decryption(string t) {
     string output;
     for (int i = 0, j = 0; i < t.length(); ++i) {
        char c = t[i];
        if (c >= 'a' && c <= 'z')
           c += 'A' - 'a';
        else if (c < 'A' || c > 'Z')
           continue;
        output += (c - k[j] + 26) % 26 + 'A';//added 'A' to bring it in range of ASCII alphabet [ 65-90 | A-Z
]
        j = (j + 1) % k.length();
     }
     return output;
  }
};
int main() {
  Vig v("WELCOME");
  cout<<"\nEnter message: ";
  string ori;
  cin>>ori;
  string encrypt = v.encryption(ori);
  string decrypt = v.decryption(encrypt);
  cout << "Original Message: "<<ori<< endl;
  cout << "Encrypted Message: " << encrypt << endl;
  cout << "Decrypted Message: " << decrypt << endl;
}
```

OUTPUT –



```
Enter message: lionkingdom
Original Message: lionkingdom
Encrypted Message: HMZPYURCHZO
Decrypted Message: LIONKINGDOM

--------------------------------
Process exited after 26.15 seconds with return value 0
Press any key to continue . . .
```

5. Implement playfair cipher substitution operation.

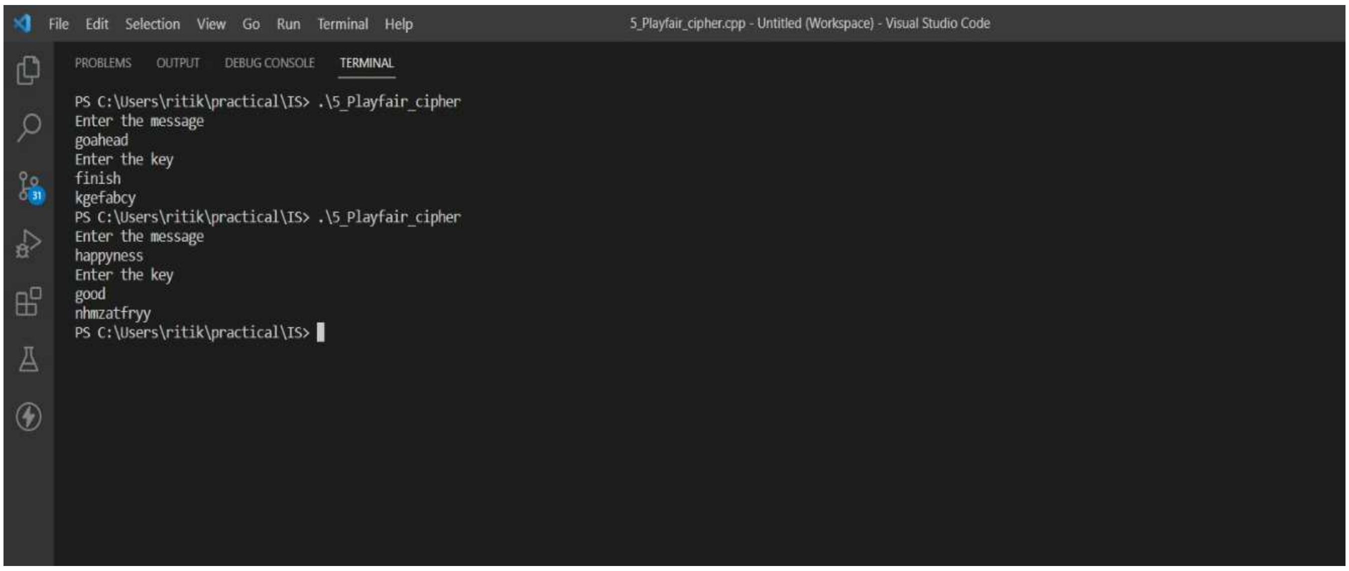
CODE –

```
#include<iostream>
#include<string>
#include<vector>
#include<map>
using namespace std;
int main(){
  int i,j,k,n;
  cout<<"Enter the message"<<endl;
  string s,origin;
  getline(cin,origin);
  cout<<"Enter the key"<<endl;
  string key;
  cin>>key;
  for(i=0;i<origin.size();i++){
    if(origin[i]!=' ')
      s+= origin[i];
  }
  vector<vector<char> > a(5,vector<char>(5,' '));
  n=5;
  map<char,int> mp;
  k=0;
  int pi,pj;
  for(i=0;i<n;i++){
    for(j=0;j<n;j++){
      while(mp[key[k]]>0&&k<key.size()){
        k++;
      }
      if(k<key.size()){
        a[i][j]=key[k];
        mp[key[k]]++;
        pi=i;
```

```
            pj=j;
        }
        if(k==key.size())
        break;
    }
    if(k==key.size())
        break;
}
k=0;
for(;i<n;i++){
    for(;j<n;j++){
        while(mp[char(k+'a')]>0&&k<26){
            k++;
        }
        if(char(k+'a')=='j'){
            j--;
            k++;
            continue;
        }
        if(k<26){
            a[i][j]=char(k+'a');
            mp[char(k+'a')]++;
        }
    }
    j=0;
}
string ans;
if(s.size()%2==1)
    s+="x";
for(i=0;i<s.size()-1;i++){
    if(s[i]==s[i+1])
```

```
            s[i+1]='x';
    }
    map<char,pair<int,int> > mp2;
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            mp2[a[i][j]] = make_pair(i,j);
        }
    }

    for(i=0;i<s.size()-1;i+=2){
        int y1 = mp2[s[i]].first;
        int x1 = mp2[s[i]].second;
        int y2 = mp2[s[i+1]].first;
        int x2 = mp2[s[i+1]].second;
        if(y1==y2){
            ans+=a[y1][(x1+1)%5];
            ans+=a[y1][(x2+1)%5];
        }
        else if(x1==x2){
            ans+=a[(y1+1)%5][x1];
            ans+=a[(y2+1)%5][x2];
        }
        else {
            ans+=a[y1][x2];
            ans+=a[y2][x1];
        }
    }
    cout<<ans<<'\n';
    return 0;
}
```

OUTPUT –


File Edit Selection View Go Run Terminal Help
5_Playfair_cipher.cpp - Untitled (Workspace) - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\ritik\practical\IS> .\5_Playfair_cipher
Enter the message
goahead
Enter the key
finish
kgefabcy
PS C:\Users\ritik\practical\IS> .\5_Playfair_cipher
Enter the message
happyness
Enter the key
good
nhmzatfryy
PS C:\Users\ritik\practical\IS>
```

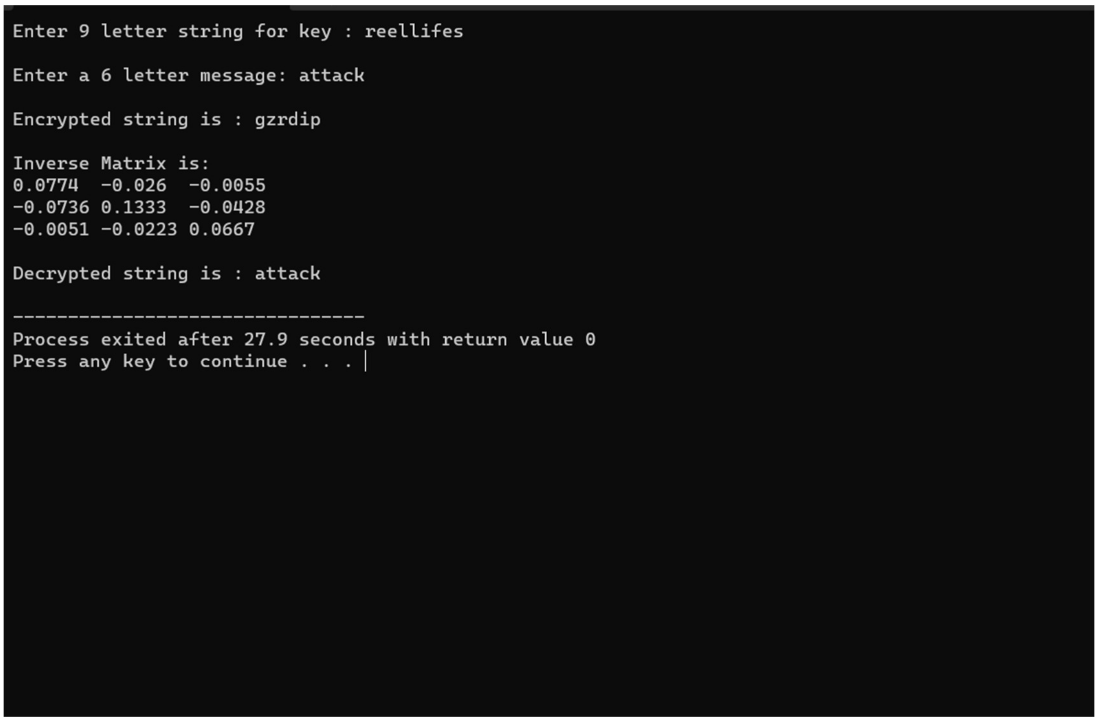
6. Implement hill cipher substitution operation.

CODE-

```
#include<iostream>

#include<cmath>

using namespace std;

float encrypt[3][2], decrypt[3][2], a[3][3], b[3][3],

mes[3][2], c[3][3];

void encryption(); //encrypts the message

void decryption(); //decrypts the message

void getKeyMessage(); //gets key and message from user

void inverse(); //finds inverse of key matrix

int main()

{

 getKeyMessage();

 encryption();

 decryption();


 return 0;
```

```
}
void encryption()
{
 int i, j, k;
 for(i = 0; i < 3; i++)
 for(j = 0; j < 2; j++)
 for(k = 0; k < 3; k++)
 encrypt[i][j] = encrypt[i][j] + a[i][k] *
 mes[k][j];
 cout << "\nEncrypted string is : ";
 for(i = 0; i < 3; i++)
 for(j = 0; j<2; j++)
 cout << (char)(fmod(encrypt[i][j], 26) + 97);
}
void decryption()
{
 int i, j, k;
 inverse();
 for(i = 0; i < 3; i++)
 for(j = 0; j < 2; j++)
 for(k = 0; k < 3; k++)
 decrypt[i][j] = decrypt[i][j] + b[i][k] *
 encrypt[k][j];
 cout << "\nDecrypted string is : ";
 for(i = 0; i < 3; i++)
 for(j = 0; j<2; j++)
 cout << (char)(fmod(decrypt[i][j], 26) + 97);
 cout << "\n";
}
void getKeyMessage()
{
 int i, j;
 char msg[6], y;
 cout << "Enter 9 letter string for key : ";
 for(i = 0; i < 3; i++)
 for(j = 0; j < 3; j++)
 {
    cin >> y;
    a[i][j] = y - 'a';
    c[i][j] = a[i][j];
 }
 cout << "\nEnter a 6 letter message: ";
 cin >> msg;
 for(i = 0; i < 3; i++)
    for(j = 0; j<2; j++)
        mes[i][j] = msg[i*2 + j] - 97;
}
void inverse()
{
 int i, j, k;
 float p, q;
 for(i = 0; i < 3; i++)
     for(j = 0; j < 3; j++)
     {
         if(i == j)
            b[i][j] = 1;
         else
            b[i][j] = 0;
     }
 for(k = 0; k < 3; k++)
 {
     for(i = 0; i < 3; i++)
 {
 p = c[i][k];
 q = c[k][k];
 for(j = 0; j < 3; j++)
 {
 if(i != k)
 {
 c[i][j] = c[i][j] * q - p * c[k][j];
 b[i][j] = b[i][j] * q - p * b[k][j];
 }
 }
 }
 }
 for(i = 0; i < 3; i++)
    for(j = 0; j < 3; j++)
        b[i][j] = ceil((b[i][j] / c[i][i])*10000)/10000.0;

 cout << "\n\nInverse Matrix is:\n";
 for(i = 0; i < 3; i++)
 {
 for(j = 0; j < 3; j++)
 cout << b[i][j] << "\t";
 cout << "\n";
 }
}
```

OUTPUT –



```
Enter 9 letter string for key : reellifes

Enter a 6 letter message: attack

Encrypted string is : gzrdip

Inverse Matrix is:
0.0774  -0.026  -0.0055
-0.0736 0.1333  -0.0428
-0.0051 -0.0223 0.0667

Decrypted string is : attack

--------------------------------
Process exited after 27.9 seconds with return value 0
Press any key to continue . . .
```

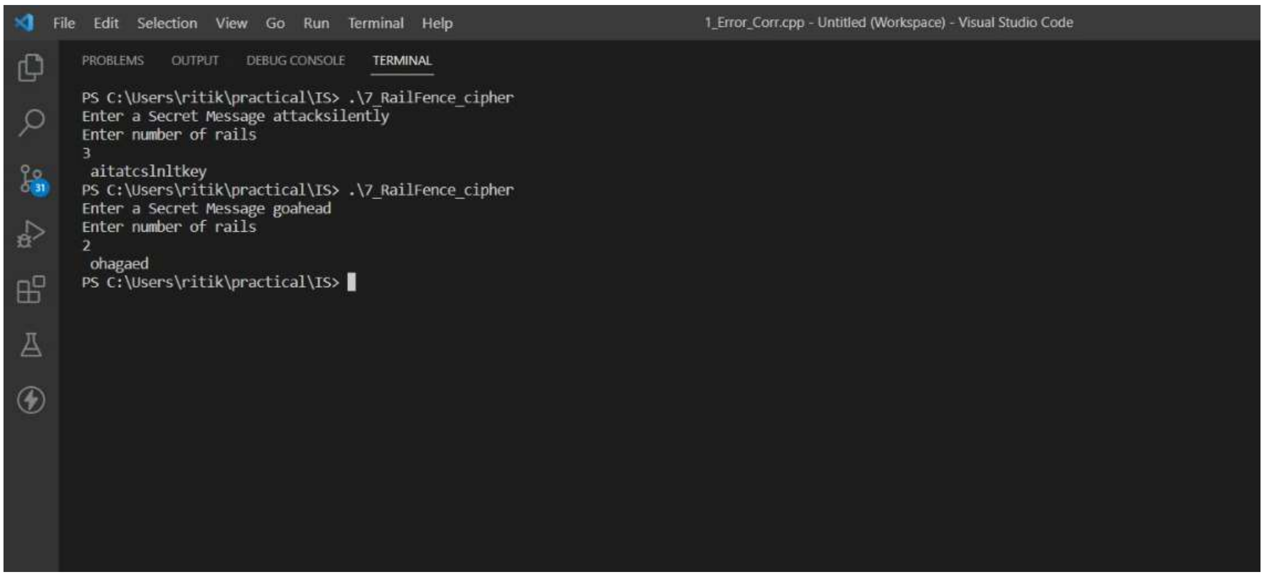
7. Implement rail fence cipher transposition operation.

CODE –

```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>

main()

{

   int i,j,len,rails,count,code[100][1000];

   char str[1000];

   printf("Enter a Secret Message");

   gets(str);

   len=strlen(str);

   printf("Enter number of rails\n");

   scanf("%d",&rails);

   for(i=0;i<rails;i++)
```

```
    {
        for(j=0;j<len;j++)
        {
            code[i][j]=0;
        }
    }
count=0;
j=0;
while(j<len)
{
if(count%2==0)
{
    for(i=0;i<rails;i++)
    {
        //strcpy(code[i][j],str[j]);
        code[i][j]=(int)str[j];
        j++;
    }

}
else
{

    for(i=rails-2;i>0;i--)
    {
        code[i][j]=(int)str[j];
    j++;
    }
}

count++;
}

for(i=0;i<rails;i++)
{
   for(j=0;j<len;j++)
   {
      if(code[i][j]!=0)
      printf("%c",code[i][j]);
   }

}
}
```

OUTPUT –


File Edit Selection View Go Run Terminal Help
1_Error_Corr.cpp - Untitled (Workspace) - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\ritik\practical\IS> .\7_RailFence_cipher
Enter a Secret Message attacksilently
Enter number of rails
3
 aitatcslnltkey
PS C:\Users\ritik\practical\IS> .\7_RailFence_cipher
Enter a Secret Message goahead
Enter number of rails
2
 ohagaed
PS C:\Users\ritik\practical\IS> 
```

8. Implement row transposition cipher transposition operation.

CODE -

```
#include<stdio.h>
```

```
#include<string.h>


void cipher(int i, int c);

int findMin();

void makeArray(int, int);


char arr[22][22], darr[22][22], emessage[111], retmessage[111], key[55];

char temp[55], temp2[55];

int k = 0;


int main()

{

    char *message;


    int i, j, klen, emlen, flag = 0;

    int r, c, index, rows;


    printf("Enter the key\n");

    fflush(stdin);

    gets(key);


    printf("\nEnter message to be ciphered\n");

    fflush(stdin);

    gets(message);


    strcpy(temp, key);

    klen = strlen(key);


    k = 0;

    for (i = 0;; i++)

    {
```

```
    if (flag == 1)
        break;

    for (j = 0; key[j] != NULL; j++)
    {
        if (message[k] == NULL)
        {
            flag = 1;
            arr[i][j] = '-';
        }
        else
        {
            arr[i][j] = message[k++];
        }
    }
}
r = i;
c = j;

for (i = 0; i < r; i++)
{
    for (j = 0; j < c; j++)
    {
        printf("%c ", arr[i][j]);
    }
    printf("\n");
}

k = 0;

for (i = 0; i < klen; i++)
```

```
{
    index = findMin();
    cipher(index, r);
}

emessage[k] = '\0';
printf("\nEncrypted message is\n");
for (i = 0; emessage[i] != NULL; i++)
    printf("%c", emessage[i]);

printf("\n\n");
//deciphering

emlen = strlen(emessage);
//emlen is length of encrypted message

strcpy(temp, key);

rows = emlen / klen;
//rows is no of row of the array to made from ciphered message

j = 0;

for (i = 0, k = 1; emessage[i] != NULL; i++, k++)
{
    //printf("\nEmlen=%d",emlen);
    temp2[j++] = emessage[i];
    if ((k % rows) == 0)
    {
        temp2[j] = '\0';
        index = findMin();
```

```
            makeArray(index, rows);
            j = 0;
        }
    }

    printf("\nArray Retrieved is\n");

    k = 0;
    for (i = 0; i < r; i++)
    {
        for (j = 0; j < c; j++)
        {
            printf("%c ", darr[i][j]);
            //retrieving message
            retmessage[k++] = darr[i][j];

        }
        printf("\n");
    }
    retmessage[k] = '\0';

    printf("\nMessage retrieved is\n");

    for (i = 0; retmessage[i] != NULL; i++)
        printf("%c", retmessage[i]);

    return (0);
}

void cipher(int i, int r)
{
    int j;
    for (j = 0; j < r; j++)
    {
        {
            emessage[k++] = arr[j][i];
        }
    }
    // emessage[k]='\0';
}

void makeArray(int col, int row)
{
    int i, j;

    for (i = 0; i < row; i++)
    {
        darr[i][col] = temp2[i];
    }
}

int findMin()
{
    int i, j, min, index;

    min = temp[0];
    index = 0;
    for (j = 0; temp[j] != NULL; j++)
    {
        if (temp[j] < min)
        {
            min = temp[j];
            index = j;
        }
    }

    temp[index] = 123;
    return (index);
}
```

OUTPUT –

9. Implement product cipher transposition operation.

10. Illustrate the Ciphertext only and Known Plaintext attacks.

ANSWER -

CipherText-only Attack

In cryptography, a ciphertext-only attack (COA) or known ciphertext attack is an attack model for cryptanalysis where the attacker is assumed to have access only to a set of ciphertexts.The attack is completely successful if the corresponding plaintexts can be deduced, or even better, the key. The ability to obtain any information at all about the underlying plaintext is still considered a success. For example, if an adversary is sending ciphertext continuously to maintain traffic-flow security, it would be very useful to be able to distinguish real messages from nulls. Even making an informed guess of the existence of real messages would facilitate traffic analysis. Every modern cipher attempts to provide protection against ciphertextonly attacks. The vetting process for a new cipher design standard usually takes many years and includes exhaustive testing of large quantities of ciphertext for any statistical departure from random noise. Encryption Standard process. Also, the field of steganography evolved, in part, to develop methods like mimic functions that allow one piece of data to adopt the statistical profile of another. Nonetheless poor cipherusage or reliance on home-grown proprietary algorithms that have notbeen subject to thorough scrutiny has resulted in many computer-age encryption systems that are still subject to ciphertext-only attack.

Known Plaintext Attack

The known-plaintext attack (KPA) or crib is an attack model for cryptanalysis where the attacker has samples of both the plaintext and its encrypted version (ciphertext), and is at liberty to make use of them to reveal further secret information such as secret keys and code books. The term "crib" originated at Bletchley Park, the British World War II decryption operation. Classical ciphers are typically vulnerable to known-plaintext attack. For example, a Caesar cipher can be solved using a single letter of corresponding plaintext and ciphertext to decrypt entirely. A general monoalphabetic substitution cipher needs several character pairs and some guessing if there are fewer than 26 distinct pairs. Modern ciphers such as Advanced Encryption Standard are not susceptible to known-plaintext attacks.

11. Implement a stream cipher technique.