SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LABORATORY (7RCSL01)

Student Name: Khalid Farooq	USN: 1SI18CS046	Batch No: A3

Evaluation:

Write Up (10 marks)	Clarity in concepts (10 marks)	Implementation and execution of the algorithms (10 marks)	Viva (05 marks)	Total (35 marks)

Sl.No	Name of the Faculty In-Charge	Signature
1.	Sunitha N R	
2.	A H Shanthakumara	

Question No: 1

Perform encryption and decryption using mono-alphabetic cipher. The program should support the following:

i. Construct an input file named plaintext.txt (consisting of 1000 alphabets, without any space or special characters)

ii. Compute key space (Permutation of set of all letters appeared in plaintext.txt: there are n! permutations of a set of n elements)

iii. Encrypt the characters of plaintext.txt using any one key from (ii) and store the corresponding ciphertext characters in ciphertext.txt

iv. Compute the frequency of occurrence of each alphabet in both plaintext.txt and ciphertext.txt and tabulate the results as follows

Frequency	Plaintext character	Ciphertext character
12.34	A	X
.	.	.
.	.	.

Monoalphabetic substitution cipher:

Select a Key randomly from 26! Key space and map from plain alphabet to cipher alphabet:

- Let us consider Plaintext P which contains every alphabets S = {a, b, c},
- There are 3! Permutations of S in a key space.
- Randomly chosen key K from key space.
- Map from plain alphabet to cipher alphabet

Program:

```
#include<bits/stdc++.h>

using namespace std ;

string uniquePlainText ;

string readPlainText(const char * name="plaintext.txt"){
    string pt ;
    ifstream fin;
    fin.open(name) ;
    fin>>pt ;
    return pt ;
}


void permute(string a, int l, int r , vector<string>& ks)
{
    if (l == r)
        ks.push_back(a) ;
    else
    {
        for (int i = l; i <= r; i++)
        {
            swap(a[l], a[i]);

            permute(a, l+1, r , ks);
```

```
            swap(a[l], a[i]);
        }
    }
}

vector<string> genKeySpace(string pt ){
    set<char> charset ;
    for(int i =0 ;i < pt.length() ; i++) charset.insert(pt[i]) ;

    uniquePlainText  = string(charset.begin() , charset.end()) ;

    vector<string> keyspace ;
    permute(uniquePlainText , 0 , uniquePlainText.length()-1 , keyspace )
;
    return keyspace ;
}


string encryptUsingKey(string uniq , string key){
    string pt = readPlainText() ;
    string ct = "" ;
    for(int i =0 ;i < pt.length() ; i++){
         char c = pt[i] ;
         ct+=(key[uniq.find(c)]) ;
    }
    cout<<"Original text = \t " << pt << endl;
    cout<< "Cipher text = \t\t " << ct <<endl;
    return ct ;
}

void saveToFile(string data , string filename ="ciphertext.txt" ){
    ofstream fout ;
    fout.open(filename.c_str()) ;
    fout<< data ;
    fout.close() ;
}


void showFrequency(string pt , string ct){
    map<char , char > mPlain ;
    map<char , char > mCipher ;

    for(int i =0 ;i < pt.length() ; i++){
         mPlain[pt[i]]++ ;
         mCipher[ct[i]]++ ;
    }
```

```
    cout<<"Frequency\t\tPlaintext Character\t\tCiphertext character"
<<endl;
    cout<<"=========\t\t==================\t\t==================="
<<endl;
    for(int i =0 ;i < pt.length() ; i++){
         cout<< (float)mPlain[pt[i]]/pt.length() << "\t\t\t" << pt[i] <<
"\t\t\t" << ct[i] << endl ;
    }

}


int main(void){
    srand(time(0)) ;
    string pt = readPlainText() ;
    cout<<"Plain text = \t " << pt << endl;
    vector<string> keyspace = genKeySpace(pt) ;
    string key = keyspace[rand()%keyspace.size()] ;

    cout<<"Unique chars = \t" << uniquePlainText <<endl;

    for(int i=0; i < keyspace.size(); i++)
         std::cout << keyspace.at(i) << ' ';
    cout<<endl;
    cout<<"Chosen key = \t" << key <<endl;
    string ct = encryptUsingKey(uniquePlainText , key) ;
    saveToFile(ct) ;
    showFrequency(pt , ct) ;
}
```

OUTPUT :

```
Plain text =     hello
Unique chars =  ehlo
ehlo ehol elho eloh eolh eohl helo heol hleo hloe hole hoel lheo lhoe leho leoh
 loeh lohe ohle ohel olhe oleh oelh oehl
Chosen key =    lhoe
Original text =          hello
Cipher text =            hlooe
Frequency               Plaintext Character            Ciphertext character
=========               ===================            ====================
0.2                     h                      h
0.2                     e                      l
0.4                     l                      o
0.4                     l                      o
0.2                     o                      e
```

```
Plain text =     Plain
Unique chars =  Pailn
Pailn Painl Palin Palni Panli Panil Pialn Pianl Pilan Pilna Pinla Pinal Plian P
lina Plain Plani Plnai Plnia Pnila Pnial Pnlia Pnlai Pnali Pnail aPiln aPinl aP
lin aPlni aPnli aPnil aiPln aiPnl ailPn ailnP ainlP ainPl aliPn alinP alPin alP
ni alnPi alniP anilP aniPl anliP anlPi anPli anPil iaPln iaPnl ialPn ialnP ianl
P ianPl iPaln iPanl iPlan iPlna iPnla iPnal ilPan ilPna ilaPn ilanP ilnaP ilnPa
 inPla inPal inlPa inlaP inalP inaPl laiPn lainP laPin laPni lanPi laniP liaPn
lianP liPan liPna linPa linaP lPian lPina lPain lPani lPnai lPnia lniPa lniaP l
nPia lnPai lnaPi lnaiP nailP naiPl naliP nalPi naPli naPil nialP niaPl nilaP ni
lPa niPla niPal nliaP nliPa nlaiP nlaPi nlPai nlPia nPila nPial nPlia nPlai nPa
li nPail
Chosen key =    ilaPn
Original text =          Plain
Cipher text =            iPlan
Frequency               Plaintext Character            Ciphertext character
=========               ===================            ====================
0.2                     P                      i
0.2                     l                      P
0.2                     a                      l
0.2                     i                      a
0.2                     n                      n
```