**Practical 1**

1. **Ceaser Cipher:**
2. User will enter text file as a plain text. [PT.TXT].
3. Load the PT.TXT file and process for input message.
4. Key will be randomly generated from 1 to 26 letters which is store into KEY.TXT file.
5. Encrypt the PT.TXT file by applying the key from KEY.TXT file.
6. Display the Cipher-Text to user.
7. Decrypt the cipher text and stored the Cipher-Text & Plain-Text pair of message in separate OUTPUT.TXT file.
8. Perform the cryptanalysis by applying brute force attack over cipher-text.

**Programming:**

Functions:

1. key\_Generation()
2. encrypt\_Text()
3. decrypt\_Text()
4. Cryptanalytics()

**CODE:-**

#include<bits/stdc++.h>

using namespace std;

void key\_Generation()

{

srand((unsigned int)time(NULL));

ofstream keystore ("KEY.txt");

int num = (rand()%26);

char chr = char(num+65);

keystore << chr;

keystore.close();

}

string encrypt\_Text(string pt,char Key)

{

string cipher\_text = "";

int key = int(Key)-65;

for(int i=0;i<pt.length();i++)

{

if(pt[i]==' ')

{

cipher\_text += ' ';

continue;

}

int v;

if(pt[i]>='a')

v = pt[i]-97;

else

v = pt[i]-65;

if (key >= 0)

v = (v+key)%26;

else

v = (v+26+key)%26;

if(pt[i]>='a')

v = v+97;

else

v = v+65;

cipher\_text += char(v);

}

cout << "\nCipher Text : " << cipher\_text << endl;

return cipher\_text;

}

string decrypt\_Text(string ct,char Key)

{

string plain\_text = "";

int key = int(Key)-65;

for(int i=0;i<ct.length();i++)

{

if(ct[i]==' ')

{

plain\_text += ' ';

continue;

}

int v;

if(ct[i]>='a')

v = ct[i]-97;

else

v = ct[i]-65;

if (key >= 0)

v = (v+26-key)%26;

else

v = (v-key)%26;

if(ct[i]>='a')

v = v+97;

else

v = v+65;

plain\_text += char(v);

}

return plain\_text;

}

void Cryptanalytics(string cipher\_text)

{

ofstream possible\_output ("possible\_output.txt");

possible\_output << "Possible Original Text" << endl;

for(int j=1;j<=26;j++)

{

int key = j;

string plain\_text = "";

for(int i=0;i<cipher\_text.length();i++)

{

if(cipher\_text[i]==' ')

{

plain\_text += ' ';

continue;

}

int v;

if(cipher\_text[i]>='a')

v = cipher\_text[i]-97;

else

v = cipher\_text[i]-65;

if (key >= 0)

v = (v+26-key)%26;

else

v = (v-key)%26;

if(cipher\_text[i]>='a')

v = v+97;

else

v = v+65;

plain\_text += char(v);

}

possible\_output << j << ". " << plain\_text << endl;

}

possible\_output.close();

}

int main()

{

string input\_message;

ifstream input1 ("PT.txt");

getline(input1,input\_message);

input1.close();

key\_Generation();

string key;

ifstream input2 ("KEY.txt");

getline(input2,key);

input2.close();

string ct = encrypt\_Text(input\_message,key[0]);

string pt = decrypt\_Text(ct,key[0]);

ofstream output ("OUTPUT.txt");

output << "(";

output << pt;

output << " , ";

output << ct;

output << ")";

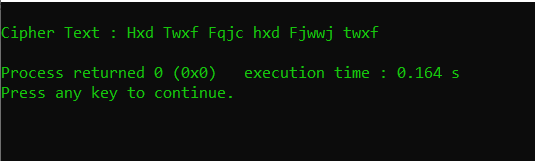
output.close();

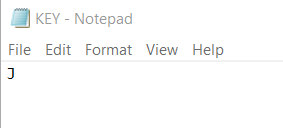
Cryptanalytics(ct);

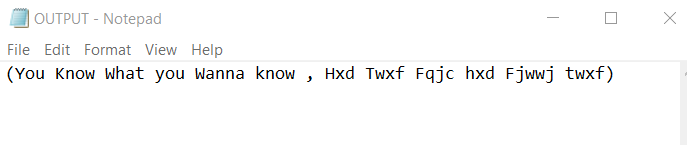
return 0;

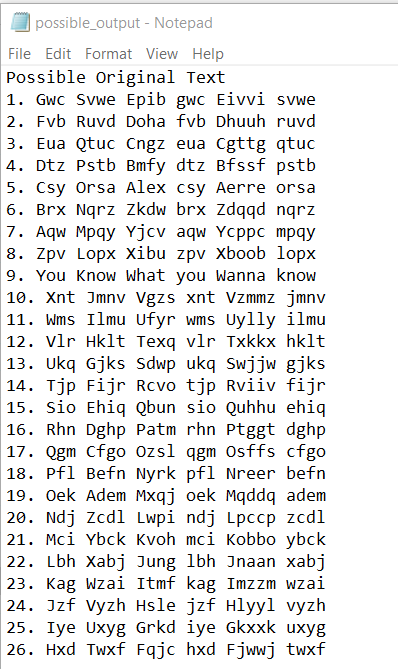
}

Output:-









1. **Playfair Cipher:**
2. User will enter text file as a plain text. [PT.TXT].
3. Load the PT.TXT file and process for input message as a group of two letters.
4. Key will be randomly generated string with minimum 6 letters and maximum 8 letters from alphabets. Display the matrix along with key and rest of alphabets.
5. Encrypt group of two letters pair wise. Show each Input pair wise letters with encrypted letters
6. Display the whole Cipher-Text to user.
7. Decrypt the cipher text and stored the Cipher-Text & Plain-Text pair of message in separate OUTPUT.TXT file.

Functions:

1. Processed\_Input/Filter\_Input
2. key\_Generation()
3. generate\_Matrix(5×5)
4. encrypt\_Text()
5. decrypt\_Text()
6. generate\_Matrix(8×8) [use Alphabets, Digits/Special Symbol]

CODE:

#include<bits/stdc++.h>

using namespace std;

string Processed\_Input(string plain\_text)

{

string plaintext = "";

char chr = 'x';

int c = 0;

for(int i=0;i<plain\_text.length()-1;i++)

{

plaintext += plain\_text[i];

if(plain\_text[i] == plain\_text[i+1])

{

plaintext += chr;

c++;

}

else

{

plaintext += plain\_text[i+1];

i++;

}

}

if((plain\_text.length()+c)%2)

{

plaintext += plain\_text[plain\_text.length()-1];

plaintext += chr;

}

return plaintext;

}

void key\_Generation()

{

string s = "";

srand((unsigned int)time(NULL));

int rot=(rand()%3)+6;

srand((unsigned int)time(NULL));

for(int i=0;i<rot;i++)

{

char chr = char((rand()%26)+97);

s += chr;

}

ofstream keystore ("KEY.txt");

keystore << s;

keystore.close();

}

void generatekeyMatrix(char a[5][5],string s)

{

int arr[26] = {0};

int k1=0,k2=0;

for(int i=0;i<s.length();i++)

{

if(!(arr[int(s[i])-97]))

{

arr[int(s[i])-97] = 1;

a[k1][k2] = s[i];

k2++;

if(k2==5)

{

k1++;

k2=0;

}

}

if(arr[int('i')-97] == 1 || arr[int('j')-97] == 1)

{

arr[int('i')-97] = arr[int('j')-97] = 1;

}

}

int index = 0;

for(int i=0;i<5;i++)

{

for(int j=0;j<5;j++)

{

if(a[i][j]==NULL)

{

while(arr[index])

{

index++;

}

arr[index] = 1;

a[i][j] = char(index+97);

}

if(arr[int('i')-97] == 1 || arr[int('j')-97] == 1)

{

arr[int('i')-97] = arr[int('j')-97] = 1;

}

}

}

}

void findchar(char a[5][5], char s,int \*p1,int \*p2)

{

for(int i=0;i<5;i++)

{

for(int j=0;j<5;j++)

{

if(s==a[i][j] || (s=='i' && a[i][j]=='j') || (s=='j' && a[i][j]=='i'))

{

\*p1 = i;

\*p2 = j;

break;

}

}

}

}

string encrypt\_Text(char a[5][5],string s)

{

string ans = "";

for(int i=0;i<s.length();i+=2)

{

char a1 = s[i];

char a2 = s[i+1];

int x1,y1,x2,y2;

findchar(a,a1,&x1,&y1);

findchar(a,a2,&x2,&y2);

if(x1==x2)

{

y1 = (y1+1)%5;

y2 = (y2+1)%5;

}

else if(y1==y2)

{

x1 = (x1+1)%5;

x2 = (x2+1)%5;

}

else

{

int temp = y2;

y2 = y1;

y1 = temp;

}

ans += a[x1][y1];

ans += a[x2][y2];

}

return ans;

}

string decrypt\_Text(char a[5][5],string s)

{

string ans = "";

for(int i=0;i<s.length();i+=2)

{

char a1 = s[i];

char a2 = s[i+1];

int x1,y1,x2,y2;

findchar(a,a1,&x1,&y1);

findchar(a,a2,&x2,&y2);

if(x1==x2)

{

if(y1==0)

{

y1 = 4;

}

else

{

y1 = (y1-1)%5;

}

if(y2==0)

{

y2 = 4;

}

else

{

y2 = (y2-1)%5;

}

}

else if(y1==y2)

{

if(x1==0)

{

x1 = 4;

}

else

{

x1 = (x1-1)%5;

}

if(x2==0)

{

x2 = 4;

}

else

{

x2 = (x2-1)%5;

}

}

else

{

int temp = y2;

y2 = y1;

y1 = temp;

}

ans += a[x1][y1];

ans += a[x2][y2];

}

return ans;

}

int main()

{

string input\_message;

ifstream input1 ("PT.txt");

getline(input1,input\_message);

input1.close();

input\_message = Processed\_Input(input\_message);

key\_Generation();

string key;

ifstream input2 ("KEY.txt");

getline(input2,key);

input2.close();

char keyMatrix[5][5] = {NULL};

generatekeyMatrix(keyMatrix,key);

cout << "\nKey Matrix\n";

for(int i=0;i<5;i++)

{

for(int j=0;j<5;j++)

{

cout << keyMatrix[i][j] << " ";

}

cout << endl;

}

string ct = encrypt\_Text(keyMatrix,input\_message);

cout << "\nCipher Text : " << ct << endl;

string pt = decrypt\_Text(keyMatrix,ct);

//cout << pt << endl;

ofstream output ("OUTPUT.txt");

output << "(";

output << pt;

output << " , ";

output << ct;

output << ")";

output.close();

return 0;

}

Output:-

