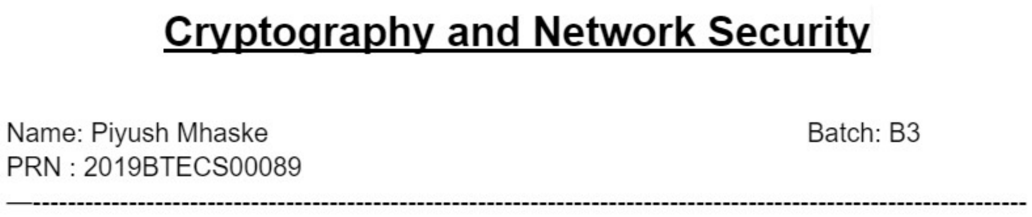


Cryptography and Network Security

Name: Piyush Mhaske Batch: B3
PRN : 2019BTECS00089

Assignment 3

Objective: Play Fair Algorithm

Theory:

The Playfair Cipher Encryption Algorithm consists of 2 steps:

1) Generate the Key Square(5x5)
The Key Square is a 5x% grid of alphabets that act as the key for encrypting the plain text. Each of the 25 alphabets must be unique and one letter of alphabet (Usually) J is omitted from the table. 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.

2) Algorithm to encrypt the plain text
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.

i) If both the letters are in same row : the letter to the right of each (going back to the leftmost position)

ii) If both the letters are in same column: take the letter below of each one (going back to the top)

iii) If neither of the above is true: form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle. Take the Row first.

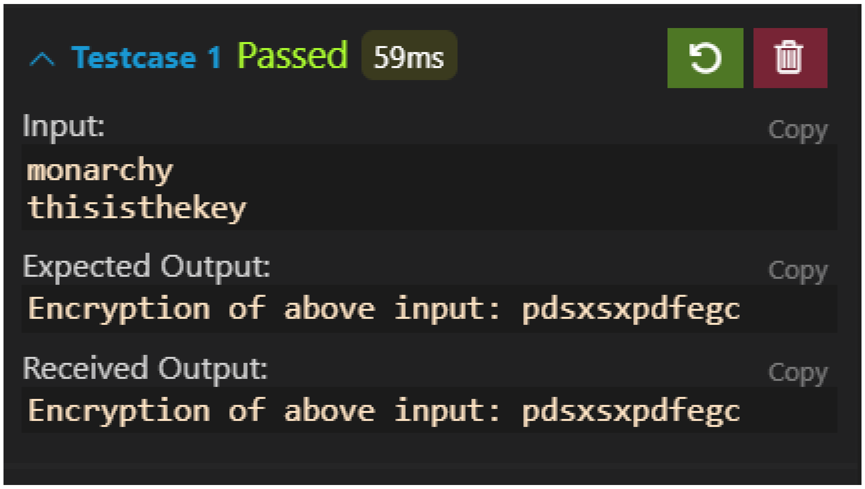
Code:

```
//code by :- Piyush Mhaske
#include <bits/stdc++.h>
#define ll long long
#define ul unsigned long long
#define pb emplace_back
#define po pop_back
#define vi vector<ll>
#define vii vector<vector<ll>>
using namespace std;
void file(){
     ios_base::sync_with_stdio(false);
     cin.tie(NULL);}
ll M = 1e9 + 7;
string search(char x, char y,  vector<vector<char>> matrix){
    int p1,q1,p2,q2;
    for(int i=0;i<5;i++){
        for(int j=0;j<5;j++){
            if(x==matrix[i][j]){
                p1=i;
                q1=j;
            }
        }
    }


    for(int i=0;i<5;i++){
        for(int j=0;j<5;j++){
            if(y==matrix[i][j]){
                p2=i;
                q2=j;
            }
        }
    }
        string ans;
    if(p1==p2){
        ans = matrix[p1][(q1+1)%5];
        ans += matrix[p2][(q2+1)%5];
    }else if(q1==q2){
        ans = matrix[(p1+1)%5][q1];
        ans += matrix[(p2+1)%5][q2];
    }else{
        ans = matrix[p1][q2];
        ans+= matrix[p2][q1];
    }

    return ans;

}
string PlayFairEnc(string key, string input){
string ans;
// Generate 5 x 5 Matrix
 vector<vector<char>> matrix(5,vector<char>(5));
```

```
vector<int> vis(27,0);
   int idx = 0;
   int bogus = 0;
   for(int i=0;i<5;i++){
       for(int j=0;j<5;j++){
           while(vis[key[idx] -'a']){
               // while(vis[bogus]){
               //     bogus = (bogus + 1)%26;
               // }
               //   matrix[i][j] = 'a' + bogus;
               //   vis[bogus]=true;
                 idx++;
           }
           if(idx < key.size()){
                matrix[i][j] = key[idx];
                vis[key[idx] -'a'] = true;
                idx++;
           }
           else{
               idx = 0;
                while(vis[idx]){
                   idx = (idx + 1)%26;
               }
               vis[idx]=true;
               matrix[i][j]  = 'a'+idx;
           }
           if(vis[8] || vis[9]){
               vis[8]=true;
               vis[9]=true;
           }

       }
   }

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

   vector<vector<char>> group;
   for(int i=0;i<input.size();i+=2){
       if( i==input.size()-1 || input[i]== input[i+1]){

           if(input[i]!='x')
           group.push_back({input[i],'x'});
           else
           group.push_back({input[i],'z'});

       }else{
            group.push_back({input[i],input[i+1]});
       }
   }
```

```
    for(auto x:group){
        cout<<x[0]<<"-"<<x[1]<<" ";
       ans = ans + search(x[0], x[1], matrix);
    }

    return ans;

}
int main()
{
    string key, input;
    cin>>key>>input;

    string ans = PlayFairEnc(key,input);
    cout<<"Encryption of above input: ";
    cout<<ans<<"\n";


    return 0;
}
```

Output:



Input:

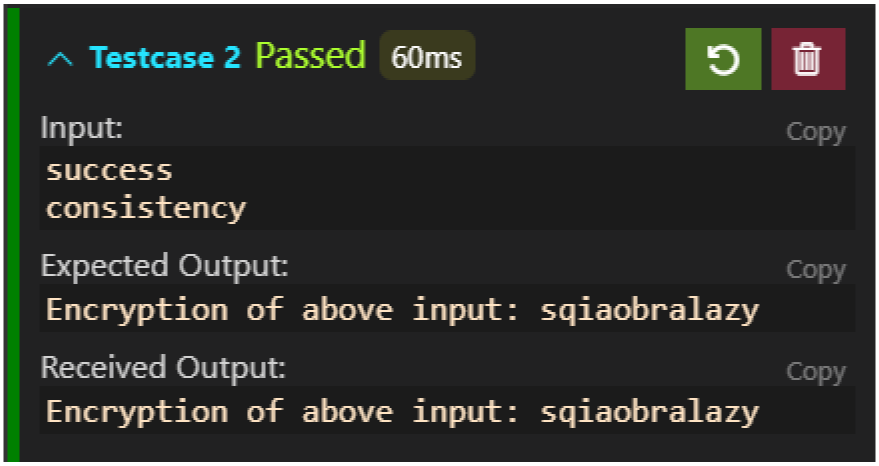
```
monarchy
thisisthekey
```

Expected Output:

```
Encryption of above input: pdsxsxpdfegc
```

Received Output:

```
Encryption of above input: pdsxsxpdfegc
```



Input:

```
success
consistency
```

Expected Output:

```
Encryption of above input: sqiaobralazy
```

Received Output:

```
Encryption of above input: sqiaobralazy
```

Conclusion:

Play Fair is better algorithm than Caesar-cipher in security. But the PlayFair can be decrypted if the key is known.