

CNS LAB

Name: Sadaf Najeem Mulla

PRN: 2019BTECS00038

Assignment 3

Aim - Given the plain text, encrypt it using Playfair Encryption Algorithm

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:

```
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;
```

```

string k;
cout << "Enter key" << endl;
cin >> k;

char mat[5][5];
int row = 0, col = 0;
map<char, int> m;
for (int i = 0; i < k.size(); i++)
{
    if (m.find(k[i]) != m.end() || k[i] == 'j') continue;
    mat[row][col] = k[i];
    m[k[i]] = 1;
    col++;
    if (col == 5)
    {
        col = 0;
        row++;
    }
}

for (int i = 0; i < 26; i++)
{
    char ch = 'a' + i;

    if (ch == 'j')
        continue;

    if (m.find(ch) != m.end())
        continue;
    m[ch] = 1;
    mat[row][col] = ch;
    col++;
    if (col == 5)
    {
        col = 0;
        row++;
    }
}

map<char, pair<int, int>> loc; cout << endl;

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

```

```

cout << mat[i][j] << " ";
loc[mat[i][j]] = {i, j};
}
cout << endl;
}

```

```

x = "";

```

```

string pos = "";

```

```

for (int i = 0; i < s.length(); i++) {
if (i == (s.length() - 1))
{
x += s[i];
x += 'x';
pos += '*';
pos += '#';
}
else
{
x += s[i];
pos += '*';
if (s[i] == s[i + 1])
{

x += 'x';
pos += '#';
}
else
{
x += s[i + 1];
i++;
pos += '*';
}
}
}
}

```

```

s = x;

```

```

cout << "\nPlain text is: " << s << endl; cout << "Key is: " << k << endl; for (int i = 0; i <
s.length(); i += 2) {
char ft = s[i];
int ftR = loc[ft].first;
int ftC = loc[ft].second;

```

```

char sd = s[i + 1];
int sdR = loc[sd].first;
int sdC = loc[sd].second;

if (ftR == sdR)
{
s[i] = (mat[ftR][(ftC + 1) % 5]);
s[i + 1] = (mat[ftR][(sdC + 1) % 5]); continue;
}

if (ftC == sdC)
{
s[i] = (mat[(ftR + 1) % 5][ftC]);
s[i + 1] = (mat[(sdR + 1) % 5][sdC]); continue;
}

s[i] = mat[ftR][sdC];
s[i + 1] = mat[sdR][ftC];

}
string cip = s;
transform(cip.begin(), cip.end(), cip.begin(), ::toupper); cout << "Cipher text is: " << cip;

```

```

for (int i = 0; i < s.length(); i += 2)
{
char ft = s[i];
int ftR = loc[ft].first;
int ftC = loc[ft].second;

char sd = s[i + 1];
int sdR = loc[sd].first;
int sdC = loc[sd].second;

if (ftR == sdR)
{
s[i] = (mat[ftR][(ftC - 1 + 5) % 5]);
s[i + 1] = (mat[ftR][(sdC - 1 + 5) % 5]);
continue;
}

if (ftC == sdC)
{

```

```
s[i] = (mat[(ftR - 1 + 5) % 5][ftC]);
s[i + 1] = (mat[(sdR - 1 + 5) % 5][sdC]);
continue;
}
```

```
s[i] = mat[ftR][sdC];
s[i + 1] = mat[sdR][ftC];
```

```
}
```

```
string ans = "";
for (int i = 0; i < s.length(); i++)
{
    if (pos[i] == '*')
        ans += s[i];
}
s = ans;
cout << "\n\nPlain text after decryption is: " << s;
```

TestCases:

D:\4th_year\CNS\Ass3.exe

Enter plain text

my name is sadaf

Enter key

best

b e s t a

c d f g h

i k l m n

o p q r u

v w x y z

Plain text is: mynameissadafx

Key is: best

Cipher text is: RTUHKTLBTBHELS

Plain text after decription is: mynameissadaf

Process exited after 101 seconds with return value 0

Press any key to continue . . .