

Name: Nikita Shivchandra Khot

PRN No: 2020BTECS00041

Assignment No 3

Study and implementation of Playfair Technique

Encryption:

Code:

```
// Playfair Cipher
#include <bits/stdc++.h>
using namespace std;
#define SIZE 30

// Function to convert the string to lowercase
void toLowerCase(char plain[], int ps)
{
    int i;
    for (i = 0; i < ps; i++)
    {
        if (plain[i] > 64 && plain[i] < 91)
            plain[i] += 32;
    }
}

// Function to remove all spaces in a string
int removeSpaces(char* plain, int ps)
{
    int i, count = 0;
    for (i = 0; i < ps; i++)
        if (plain[i] != ' ')
            plain[count++] = plain[i];
    plain[count] = '\0';
    return count;
}

// Function to generate the 5x5 key square
void generateKeyTable(char key[], int ks, char keyT[5][5])
{
    int i, j, k;
    // a 26 character hashmap
    // to store count of the alphabet
    int dicty[26] = { 0 };
    for (i = 0; i < ks; i++)
    {
```

```

        if (key[i] != 'j')
            dicty[key[i] - 97] = 2;
    }
    dicty['j' - 97] = 1;
    i = 0;
    j = 0;
    for (k = 0; k < ks; k++)
    {
        if (dicty[key[k] - 97] == 2)
        {
            dicty[key[k] - 97] -= 1;
            keyT[i][j] = key[k];
            j++;
            if (j == 5)
            {
                i++;
                j = 0;
            }
        }
    }
    for (k = 0; k < 26; k++)
    {
        if (dicty[k] == 0)
        {
            keyT[i][j] = (char)(k + 97);
            j++;
            if (j == 5)
            {
                i++;
                j = 0;
            }
        }
    }
}

// Function to search for the characters of a digraph
// in the key square and return their position
void search(char keyT[5][5], char a, char b, int arr[])
{
    int i, j;
    if (a == 'j')
        a = 'i';
    else if (b == 'j')
        b = 'i';
    for (i = 0; i < 5; i++)
    {

```

```

        for (j = 0; j < 5; j++)
        {
            if (keyT[i][j] == a)
            {
                arr[0] = i;
                arr[1] = j;
            }
            else if (keyT[i][j] == b)
            {
                arr[2] = i;
                arr[3] = j;
            }
        }
    }
}

// Function to find the modulus with 5
int mod5(int a) { return (a % 5); }

// Function to make the plain text length to be even
int prepare(char str[], int ptrs)
{
    if (ptrs % 2 != 0)
    {
        str[ptrs++] = 'z';
        str[ptrs] = '\0';
    }
    return ptrs;
}

// Function for performing the encryption
void encrypt(char str[], char keyT[5][5], int ps)
{
    int i, a[4];
    for (i = 0; i < ps; i += 2)
    {
        search(keyT, str[i], str[i + 1], a);
        if (a[0] == a[2])
        {
            str[i] = keyT[a[0]][mod5(a[1] + 1)];
            str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];
        }
        else if (a[1] == a[3])
        {
            str[i] = keyT[mod5(a[0] + 1)][a[1]];
            str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];
        }
    }
}

```

```

    }
    else
    {
        str[i] = keyT[a[0]][a[3]];
        str[i + 1] = keyT[a[2]][a[1]];
    }
}

// Function to encrypt using Playfair Cipher
void encryptByPlayfairCipher(char str[], char key[])
{
    char ps, ks, keyT[5][5];
    // Key
    ks = strlen(key);
    ks = removeSpaces(key, ks);
    toLowerCase(key, ks);
    // Plaintext
    ps = strlen(str);
    toLowerCase(str, ps);
    ps = removeSpaces(str, ps);
    ps = prepare(str, ps);
    generateKeyTable(key, ks, keyT);
    encrypt(str, keyT, ps);
}

// Driver code
int main()
{
    char str[30], key[30];
    strcpy(key, "Monarchy");
    cout << "Key text: " << key << "\n";
    strcpy(str, "nikitakhot");
    cout << "Plain text: " << str << "\n";
    encryptByPlayfairCipher(str, key);
    cout << "\n(Encrypted) Cipher text: " << str << "\n";
    return 0;
}

```

Output:

```

Key text: Monarchy
Plain text: nikitakhot

(Encrypted) Cipher text: ageksrfdrrp

```

Example:

Play Fair					key = monarchy
m	o	n	a	r	plain text = nikitakhot
c	h	y	b	d	
e	f	g	i/j	k	<u>ni</u> <u>ki</u> <u>ta</u> <u>kh</u> <u>ot</u>
l	p	q	s	t	
u	v	w	x	z	ag ek sr fd rp : encrypted
					fd rp

Decryption:

Code:

```
#include <bits/stdc++.h>
using namespace std;
#define SIZE 30

void toLowerCase(char plain[], int ps)
{
    int i;
    for (i = 0; i < ps; i++) {
        if (plain[i] > 64 && plain[i] < 91)
            plain[i] += 32;
    }
}

int removeSpaces(char* plain, int ps)
{
    int i, count = 0;
    for (i = 0; i < ps; i++)
        if (plain[i] != ' ')
            plain[count++] = plain[i];
    plain[count] = '\0';
    return count;
}

// generates the 5x5 key square
void generateKeyTable(char key[], int ks, char keyT[5][5])
{
    int i, j, k, *dicty;

    dicty = (int*)calloc(26, sizeof(int));
```

```

    for (i = 0; i < ks; i++) {
        if (key[i] != 'j')
            dicty[key[i] - 97] = 2;
    }
    dicty['j' - 97] = 1;

    i = 0;
    j = 0;
    for (k = 0; k < ks; k++) {
        if (dicty[key[k] - 97] == 2) {
            dicty[key[k] - 97] -= 1;
            keyT[i][j] = key[k];
            j++;
            if (j == 5) {
                i++;
                j = 0;
            }
        }
    }
    for (k = 0; k < 26; k++) {
        if (dicty[k] == 0) {
            keyT[i][j] = (char)(k + 97);
            j++;
            if (j == 5) {
                i++;
                j = 0;
            }
        }
    }
}

void search(char keyT[5][5], char a, char b, int arr[])
{
    int i, j;

    if (a == 'j')
        a = 'i';
    else if (b == 'j')
        b = 'i';

    for (i = 0; i < 5; i++) {
        for (j = 0; j < 5; j++) {
            if (keyT[i][j] == a) {
                arr[0] = i;
                arr[1] = j;
            }
        }
    }
}

```

```

        }
        else if (keyT[i][j] == b) {
            arr[2] = i;
            arr[3] = j;
        }
    }
}

int mod5(int a)
{
    if (a < 0)
        a += 5;
    return (a % 5);
}

void decrypt(char str[], char keyT[5][5], int ps)
{
    int i, a[4];
    for (i = 0; i < ps; i += 2) {
        search(keyT, str[i], str[i + 1], a);
        if (a[0] == a[2]) {
            str[i] = keyT[a[0]][mod5(a[1] - 1)];
            str[i + 1] = keyT[a[0]][mod5(a[3] - 1)];
        }
        else if (a[1] == a[3]) {
            str[i] = keyT[mod5(a[0] - 1)][a[1]];
            str[i + 1] = keyT[mod5(a[2] - 1)][a[1]];
        }
        else {
            str[i] = keyT[a[0]][a[3]];
            str[i + 1] = keyT[a[2]][a[1]];
        }
    }
}

void decryptByPlayfairCipher(char str[], char key[])
{
    char ps, ks, keyT[5][5];

    // Key
    ks = strlen(key);
    ks = removeSpaces(key, ks);
    toLowerCase(key, ks);
    // ciphertext
    ps = strlen(str);

```

```

        toLowerCase(str, ps);
        ps = removeSpaces(str, ps);
        generateKeyTable(key, ks, keyT);
        decrypt(str, keyT, ps);
    }

int main()
{
    char str[SIZE], key[SIZE];
    strcpy(key, "Monarchy");
    cout << "Key Text: " << key << endl;
    strcpy(str, "ageksrfdrp");
    cout << "Plain text: " << str << endl;
    decryptByPlayfairCipher(str, key);
    cout << "Deciphered text: " << str << endl;
    return 0;
}

```

Output:

```

Key Text: monarchy
Plain text: ageksrfdrp
decrypted text: nikitakhot

```