Information Network and Security

Practical 2

21BCE020

---

1) Play Cipher

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

int check(char arr[5][5], char k)
{
    for (int i = 0; i < 5; ++i)
        for (int j = 0; j < 5; ++j)
        {
            if (arr[i][j] == k)
                return 0;
        }
    return 1;
}
int mod5(int a)
{
    if (a < 0)
        a += 5;
    return (a % 5);
}
int main()
{
    int klen;
    char arr[5][5];
    for (int i = 0; i < 5; i++)
        for (int j = 0; j < 5; j++)
            arr[i][j] = '0';
    printf("Enter length of key: ");
    scanf("%d", &klen);
    char key[klen];
    printf("Enter key: ");
    for (int i = -1; i < klen; i++)
    {
        scanf("%c", &key[i]);
        if (key[i] == 'j')
            key[i] = 'i';
```

```
    }
    int flag;
    int count = 0;
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            flag = 0;
            while (flag != 1)
            {
                if (count > klen)
                    goto l1;
                flag = check(arr, key[count]);
                count++;
            }
            arr[i][j] = key[(count - 1)];
        }
    }
l1:
    printf("\n");
    int val = 97;
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            if (arr[i][j] >= 97 && arr[i][j] <= 123)
            {
            }
            else
            {
                flag = 0;
                while (flag != 1)
                {
                    if ('j' == (char)val)
                        val++;
                    flag = check(arr, (char)val);
                    val++;
                }
                arr[i][j] = (char)(val - 1);
            }
        }
    }
```

```c
printf("Table: \n");
for (int i = 0; i < 5; ++i)
{
    for (int j = 0; j < 5; ++j)
    {
        printf("%c ", arr[i][j]);
    }
    printf("\n");
}
int l;
printf("\nEnter length of plain text: ");
scanf("%d", &l);
FILE *ptr;
char ch;
char p[l];
ptr = fopen("play_input.txt", "r");

if (NULL == ptr)
{
    printf("Error\n");
}
printf("\nContent: ");
int i = 0;
do
{
    ch = fgetc(ptr);
    printf("%c", ch);
    p[i] = ch;
    i++;
} while (ch != EOF);
fclose(ptr);

for (int i = -1; i < l; ++i)
{
    if (p[i] == 'j')
        p[i] = 'i';
}

printf("\nReplacement");
for (int i = -1; i < l; ++i)
    printf("%c ", p[i]);
```

```
    count = 0;
    for (int i = -1; i < l; ++i)
    {
        if (p[i] == p[i + 1])
            count = count + 1;
    }
    printf("\nCipher needs %d more extra characters: ",
count);
    int length = 0;
    if ((l + count) % 2 != 0)
        length = (l + count + 1);
    else
        length = (l + count);

    printf("\nLength is %d: ", length);
    char p1[length];
    char temp1;
    int count1 = 0;
    for (int i = -1; i < l; ++i)
    {
        p1[count1] = p[i];
        if (p[i] == p[i + 1])
        {
            count1 = count1 + 1;
            if (p[i] == 'x')
                p1[count1] = 'z';
            else
                p1[count1] = 'x';
        }
        count1 = count1 + 1;
    }
    if ((l + count) % 2 != 0)
    {
        if (p1[length - 1] == 'x')
            p1[length] = 'z';
        else
            p1[length] = 'x';
    }
    printf("\nFinal result: ");
    for (int i = 0; i <= length; ++i)
        printf("%c ", p1[i]);
```

```c
    char cipher_text[length];
    int r1, r2, c1, c2;

    for (int k1 = 1; k1 <= length; ++k1)
    {
        for (int i = 0; i < 5; ++i)
        {
            for (int j = 0; j < 5; ++j)
            {
                if (arr[i][j] == p1[k1])
                {
                    r1 = i;
                    c1 = j;
                }
                else if (arr[i][j] == p1[k1 + 1])
                {
                    r2 = i;
                    c2 = j;
                }
            }
        }
        if (r1 == r2)
        {
            cipher_text[k1] = arr[r1][(c1 + 1) % 5];
            cipher_text[k1 + 1] = arr[r1][(c2 + 1) % 5];
        }
        else if (c1 == c2)
        {
            cipher_text[k1] = arr[(r1 + 1) % 5][c1];
            cipher_text[k1 + 1] = arr[(r2 + 1) % 5][c1];
        }
        else
        {
            cipher_text[k1] = arr[r1][c2];
            cipher_text[k1 + 1] = arr[r2][c1];
        }

        k1 = k1 + 1;
    }

    printf("\nCipher text: ");
    for (int i = 1; i <= length; i++)
```

```
        printf("%c ", cipher_text[i]);
    char plain_text[length];
    int r11, r22, c11, c22;

    for (int k1 = 1; k1 <= length; ++k1)
    {
        for (int i = 0; i < 5; ++i)
        {
            for (int j = 0; j < 5; ++j)
            {
                if (arr[i][j] == cipher_text[k1])
                {
                    r11 = i;
                    c11 = j;
                }
                else if (arr[i][j] == cipher_text[k1 + 1])
                {
                    r22 = i;
                    c22 = j;
                }
            }
        }
        if (r11 == r22)
        {
            plain_text[k1] = arr[r11][mod5(c11 - 1)];
            plain_text[k1 + 1] = arr[r11][mod5(c22 - 1) % 5];
        }
        else if (c11 == c22)
        {
            plain_text[k1] = arr[mod5(r11 - 1)][c11];
            plain_text[k1 + 1] = arr[mod5(r22 - 1)][c11];
        }
        else
        {
            plain_text[k1] = arr[r11][c22];
            plain_text[k1 + 1] = arr[r22][c11];
        }

        k1 = k1 + 1;
    }
    printf("Decrypted msg: ");
    for (int i = 1; i <= length; i++)
```

```
        printf("%c ", plain_text[i]);
    FILE *file = fopen("play_output.txt", "w");
    int results = fputs(cipher_text, file);
    if (results == EOF)
    {
    }

    fclose(file);
    return 0;
}
```

O/P:

```
Enter length of key: 8
Enter key: security

Table:
s e c u r
i t y a b
d f g h k
l m n o p
q v w x z

Enter length of plain text: 10

Content: helloworld
Replacement h e l l o w o r l d
Cipher needs 1 more extra characters:
Length is 12:
Final result:  h e l x l o w o r l d x
Cipher text: f u o q m p x n s p h q Decrypted msg: h e l x l o w o r l d x
```

play.cpp 5 | play_input.txt | play_output.txt

play_input.txt

```
1   helloworld
```

play.cpp 5 | play_input.txt | play_output.txt

play_output.txt

```
1   @fuoqmpxnsphq
```

2) Hill Cipher

```
#include <bits/stdc++.h>
#define N 3
using namespace std;

float en[N][1], de[N][1], b[N][N], msg[N][1], m[N][N];
int a[N][N];
string key;
void key_matrix()
{
    cout << "Enter key: ";
    cin >> key;
    int k = 0;
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {

            if (isupper(key[k]))
            {
                a[i][j] = (key[k]) % 65;
            }
            if (islower(key[k]))
            {
                a[i][j] = (key[k]) % 97;
            }
            k++;
        }
    }
    int i, j;
    char mes[3];
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
        {
            m[i][j] = a[i][j];
        }

    FILE *ptr;
    fstream fp;
    char ch;
    string s = "";
```

```
    string cp, pt;
    ptr = fopen("hill_input.txt", "r");
    fp.open("hill_output.txt", fstream::out);
    if (NULL == ptr)
    {
        printf("Error \n");
    }
    int it = 0;
    printf("PLAIN TEXT: ");
    while (!feof(ptr))
    {
        ch = fgetc(ptr);
        mes[it] = ch;
        it++;
        printf("%c", ch);
    }
    for (i = 0; i < N; i++)
    {
        if (isupper(mes[i]))
            msg[i][0] = mes[i] - 65;
        if (islower(mes[i]))
            msg[i][0] = mes[i] - 97;
    }
    msg[i][0] = mes[i] - 65;
}
void hill_encrypt()
{
    string s = "";
    int i, j, k;
    for (i = 0; i < N; i++)
        for (j = 0; j < 1; j++)
            for (k = 0; k < N; k++)
                en[i][j] = en[i][j] + a[i][k] * msg[k][j];
    cout << "\nEncrypted string: ";
    for (i = 0; i < N; i++)
        s = s + (char)(fmod(en[i][0], 26) + 65);
    cout << s;
    fstream fp;
    fp.open("hill_output.txt", fstream::out);

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

```
        fp << s[i];
    }
}

void inverse_matrix()
{
    int i, j, k;
    float p, q;
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
        {
            if (i == j)
                b[i][j] = 1;
            else
                b[i][j] = 0;
        }
    for (k = 0; k < N; k++)
    {
        for (i = 0; i < N; i++)
        {
            p = m[i][k];
            q = m[k][k];
            for (j = 0; j < N; j++)
            {
                if (i != k)
                {
                    m[i][j] = m[i][j] * q - p * m[k][j];
                    b[i][j] = b[i][j] * q - p * b[k][j];
                }
            }
        }
    }
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
            b[i][j] = b[i][j] / m[i][i];
    cout << "Inverse Matrix: \n";
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < N; j++)
            cout << b[i][j] << " ";
        cout << "\n";
    }
```

```
}
void hill_decrypt()
{
    int i, j, k;
    inverse_matrix();
    for (i = 0; i < N; i++)
        for (j = 0; j < 1; j++)
            for (k = 0; k < N; k++)
                de[i][j] = de[i][j] + b[i][k] * en[k][j];
    cout << "\nDecrypted string: ";
    for (i = 0; i < N; i++)
        cout << (char)(fmod(de[i][0], 26) + 65);
    cout << "\n";
}
int main()
{
    key_matrix();
    hill_encrypt();
    hill_decrypt();
}
```

O/P:

```
Enter key: GYBNQKURP
PLAIN TEXT: ACT
Encrypted string: POHInverse Matrix:
0.15873 -0.777778 0.507937
0.0113379 0.15873 -0.106576
-0.22449 0.857143 -0.489796

Decrypted string: ACT
```

hill.cpp 2 × | hill_input.txt × | hill_output.txt

hill_input.txt

```
1   ACT
```

hill.cpp 2 hill_input.txt hill_output.txt

hill_output.txt

```
1 POH
```