Student Name: Khalid Farooq	USN: 1SI18CS046	Batch No: A3	Date:

Evaluation:

Write Up (10 marks)	Clarity in concepts (10 marks)	Implementation and execution of the algorithms (10 marks)	Viva (05 marks)	Total (35 marks)

Sl.No	Name of the Faculty In-Charge	Signature
1.	Sunitha N R	
2.	A H Shanthakumara	

Question No: 2

Write a program to perform the following using Playfair cipher technique

(i) Encrypt a given message M with different keys {k_1, k_2,…,k_n}. Print key and ciphertext pair

(ii) Decrypt the cipher texts obtained in (i) to get back M

Playfair Cipher:

Construct a 5 X 5 matrix using a keyword from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetical order. The letters I and J count as one letter.

Plaintext is encrypted two letters at a time, according to the following rules:

1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.
2. Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last.
3. Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last.
4. Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.

Program:

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

typedef struct
{
     int row;
     int col;
} position;

char mat[5][5];

void generatematrix (string key)
{
     int flag[26] = { 0 }; int x = 0, y = 0;
     for (int i = 0; i < key.length (); i++)
     {
          if (key[i] == 'j')
               key[i] = 'i';
          if (flag[key[i] - 'a'] == 0)
          {
               mat[x][y++] = key[i];
               flag[key[i] - 'a'] = 1;
          }
          if (y == 5)
          {
               x++;
               y = 0;
          }
     }
     for (char ch = 'a'; ch <= 'z'; ch++)
     {
          if (ch == 'j') continue;
          if (flag[ch - 'a'] == 0)
          {
               mat[x][y++] = ch;
               flag[ch - 'a'] = 1;
          }
          if (y == 5)
          {
          x++; y = 0;
          }
     }
}
```

```
string formatmessage (string msg)
{
     for (int i = 0; i < msg.length (); i++) if (msg[i] == 'j')
     msg[i] = 'i';
     for (int i = 1; i < msg.length (); i += 2) if (msg[i - 1] == msg[i])
     msg.insert (i, "x");
     if (msg.length () % 2 != 0) msg += 'x';
     return msg;
}

position getposition (char c)
{
     position p;
     for (int i = 0; i < 5; i++)
     {
          for (int j = 0; j < 5; j++)
          {
               if (c == mat[i][j])
               {
                    p ={i, j};
                    return p;
               }
          }
     }
     return p;
}

string encrypt (string message)
{
     string ctext;
     for (int i = 0; i < message.length (); i += 2)
     {
          position p1 = getposition (message[i]); position p2 =
getposition (message[i + 1]); int x1 = p1.row;
          int y1 = p1.col; int x2 = p2.row; int y2 = p2.col; if (x1 == x2)
          {
               ctext.append (1, mat[x1][(y1 + 1) % 5]);
               ctext.append (1, mat[x2][(y2 + 1) % 5]);
          }
          else if (y1 == y2)
          {
               ctext.append (1, mat[(x1 + 1) % 5][y1]);
               ctext.append (1, mat[(x2 + 1) % 5][y2]);
          }
          else
          {
               ctext.append (1, mat[x1][y2]);
               ctext.append (1, mat[x2][y1]);
          }
```

```
     }

     return ctext;
}

string decrypt (string message)
{
     string ptext;
     for (int i = 0; i < message.length (); i += 2)
     {
          position p1 = getposition (message[i]); position p2 =
getposition (message[i + 1]); int x1 = p1.row;
          int y1 = p1.col; int x2 = p2.row; int y2 = p2.col;
          if (x1 == x2)
          {
               ptext.append (1, mat[x1][--y1 < 0 ? 4 : y1]);
               ptext.append (1, mat[x2][--y2 < 0 ? 4 : y2]);
          }
          else if (y1 == y2)
          {
               ptext.append (1, mat[--x1 < 0 ? 4 : x1][y1]);
               ptext.append (1, mat[--x2 < 0 ? 4 : x2][y2]);
          }
          else{
               ptext.append (1, mat[x1][y2]);
               ptext.append (1, mat[x2][y1]);
               }
          }

     return ptext;
}

int main ()
{
     string plaintext;
     cout << "Enter message:"; cin >> plaintext;
     int n;
     cout << "Enter number of keys:"; cin >> n;
     string key[n];
     for (int i = 0; i < n; i++)
     {
          cout << "\nEnter key" << i + 1 << ":"; cin >> key[i];
          generatematrix (key[i]);
          cout << "\nKey" << i + 1 << "Matrix" << endl; for (int k = 0; k
< 5; k++)
          {
               for (int j = 0; j < 5; j++)
               {
                    cout << mat[k][j] << " ";
               }
               cout << endl;
```

```
            }
            cout << "Actual message: " << plaintext << endl; string fmsg =
formatmessage (plaintext);
            cout << "Formatted message: " << fmsg << endl; string ciphertext
= encrypt (fmsg);
            cout << "Encrypted message: " << ciphertext << endl; string
decryptmessage = decrypt (ciphertext);
            cout << "Decrypted message: " << decryptmessage << endl;
      }
      return 0;
}
```

Output:

```
wanderer@wanderer-den:~/Documents/cns lab$ g++ 1.cpp
wanderer@wanderer-den:~/Documents/cns lab$ ./a.out
Enter message : hello
Enter number of keys : 2

Enter key 1 : monarchy
Key 1 Matrix:
m o n a r
c h y b d
e f g i k
l p q s t
u v w x z
Actual Message          : hello
Formatted Message       : helxlo
Encrypted Message       : cfsupm
Decrypted Message       : helxlo

Enter key 2 : playwell
Key 2 Matrix:
p l a y w
e b c d f
g h i k m
n o q r s
t u v x z
Actual Message          : hello
Formatted Message       : helxlo
Encrypted Message       : gbyubu
Decrypted Message       : helxlo
```