

EX NO:02

DATE:21.02.24

Write a program for demonstration of encrypting and decrypting the messages by Playfair Substitution technique.

Aim: To write a c program to perform encryption and decryption of a message using Playfair substitution technique.

Algorithm:

1. Include necessary header files (<stdio.h> and <string.h>).
2. Declare character arrays for the original message (msg), encryption key (key), new key (newKey), encrypted message (encryptedMsg), and decrypted message (decryptedMsg).
3. Declare integer variables msgLen, keyLen, i, and j for storing lengths and loop indices.
4. Initialize msg and key with the original message and encryption key.
5. Calculate msgLen and keyLen using strlen.
6. Use a loop to generate the new key (newKey) based on the original key (key).
7. Initialize i and j to 0.
8. Use a loop to iterate over each character in the original message (msg).
9. Combine it with the corresponding character from the new key (newKey) using modular arithmetic.
10. Add a null terminator at the end of the decrypted message.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define SIZE 5

char playfair[SIZE][SIZE];

void preparePlayfairKey(char *key) {
    char *ptr, *temp;
    int i, j, k, l, flag = 0;
    char alphabet[26] = {0};

    ptr = key;
    temp = key;

    while (*ptr != '\0') {
        if (*ptr >= 'a' && *ptr <= 'z') {
            *ptr = *ptr - 32;
        }
        ptr++;
    }
}
```

```

while (*temp != '\0') {
    if (*temp == 'J') {
        *temp = 'I';
    }
    if (alphabet[*temp - 65] == 0) {
        alphabet[*temp - 65] = 1;
        playfair[flag / SIZE][flag % SIZE] = *temp;
        flag++;
    }
    temp++;
}

// Fill the remaining characters
for (i = 0; i < 26; i++) {
    if (alphabet[i] == 0) {
        playfair[flag / SIZE][flag % SIZE] = (char) (i + 65);
        flag++;
    }
}
}

```

```

void constructPlayfairTable(char *key) {
    int i, j;

    preparePlayfairKey(key);

    printf("\nPlayfair Key Matrix:\n");
    for (i = 0; i < SIZE; i++) {
        for (j = 0; j < SIZE; j++) {
            printf("%c ", playfair[i][j]);
        }
        printf("\n");
    }
}

```

```

void encryptPlayfair(char *text, char *key) {
    constructPlayfairTable(key);

    int i, j, a, b, m, n;
    char p1, p2;

    for (i = 0; i < strlen(text); i += 2) {
        p1 = text[i];
        p2 = text[i + 1];
        for (j = 0; j < SIZE; j++) {
            for (m = 0; m < SIZE; m++) {
                if (playfair[j][m] == p1) {

```

```

        a = j;
        b = m;
    } else if (playfair[j][m] == p2) {
        a = j;
        b = m;
    }
}
}
if (a == 0) {
    p1 = playfair[0][b];
    p2 = playfair[SIZE - 1][b];
} else if (b == 0) {
    p1 = playfair[a][0];
    p2 = playfair[a][SIZE - 1];
} else {
    p1 = playfair[a][b - 1];
    p2 = playfair[a - 1][b];
}
printf("%c%c ", p1, p2);
}
printf("\n");
}

void decryptPlayfair(char *text, char *key) {
    constructPlayfairTable(key);

    int i, j, a, b, m, n;
    char p1, p2;

    for (i = 0; i < strlen(text); i += 2) {
        p1 = text[i];
        p2 = text[i + 1];
        for (j = 0; j < SIZE; j++) {
            for (m = 0; m < SIZE; m++) {
                if (playfair[j][m] == p1) {
                    a = j;
                    b = m;
                } else if (playfair[j][m] == p2) {
                    a = j;
                    b = m;
                }
            }
        }
        if (a == SIZE - 1) {
            p1 = playfair[SIZE - 1][b];
            p2 = playfair[0][b];
        } else if (b == SIZE - 1) {
            p1 = playfair[a][SIZE - 1];

```

```

        p2 = playfair[a][0];
    } else {
        p1 = playfair[a][b + 1];
        p2 = playfair[a + 1][b];
    }
    printf("%c%c ", p1, p2);
}
printf("\n");
}

int main() {
    char text[100], key[25];
    int choice;

    printf("Enter the key (no spaces, all uppercase): ");
    scanf("%s", key);

    printf("Enter the text (uppercase): ");
    scanf("%s", text);

    printf("\n1. Encrypt\n2. Decrypt\nEnter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("\nEncrypted Text: ");
            encryptPlayfair(text, key);
            break;
        case 2:
            printf("\nDecrypted Text: ");
            decryptPlayfair(text, key);
            break;
        default:
            printf("\nInvalid choice!\n");
    }

    return 0;
}

```

Input and Output:

```

101 cipher.cpp playfair.cpp pf2.cpp hill cipher 2.cpp 4th ex (vigenere cipher).cpp
102
103 p1 = text[i];
104 p2 = text[i + 1];
105 for (j = 0; j < SIZE; j++) {
106     for (m = 0; m < SIZE; m++) {
107         if (playfair[j][m] == p1) {
108             a = j;
109             b = m;
110         } else if (playfair[j][m] == p2) {
111             a = j;
112             b = m;
113         }
114     }
115     if (a == SIZE - 1) {
116         p1 = playfair[SIZE - 1][b];
117         p2 = playfair[0][b];
118     } else if (b == SIZE - 1) {
119         p1 = playfair[a][SIZE - 1];
120         p2 = playfair[a][0];
121     } else {
122         p1 = playfair[a][b + 1];
123         p2 = playfair[a + 1][b];
124     }
125     printf("%c%c ", p1, p2);
126 }
127 printf("\n");
128
129 int main() {
130     char text[100], key[25];
131     int choice;
132 }

```

```

C:\Users\aiswarya\Downloads\4th ex (vigenere cipher).exe
Enter the key (no spaces, all uppercase): NETWORK
Enter the text (uppercase): WORLD

1. Encrypt
2. Decrypt
Enter your choice: 1

Encrypted Text:
Playfair Key Matrix:
W E T H O
R K A B C
D F G H I
J L M P Q
S U V X Y
O Y J P D I

-----
Process exited after 8.192 seconds with return value 0
Press any key to continue . . .

```

Compilation results...

```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\aiswarya\Downloads\4th ex (vigenere cipher).exe
- Output Size: 131.658203125 KiB
- Compilation Time: 5.56s

```

Result: A c program to perform encryption and decryption of a message using Play fair substitution technique is successfully executed.

SAVEETHA SCHOOL OF ENGINEERING

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES