***SAVEETHA SCHOOL OF ENGINEERING***

***SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCE***

**EXP NO 2: A program for demonstration of encrypting and decrypting the messages by Playfair Substitution technique.**

**AIM**

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

**PROCEDURE**

* Traverse the given text one character at a time .
* For each character, transform the given character as per the rule,depending on whether we’re encrypting or decrypting the text.
* Return the new string generated.

**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;

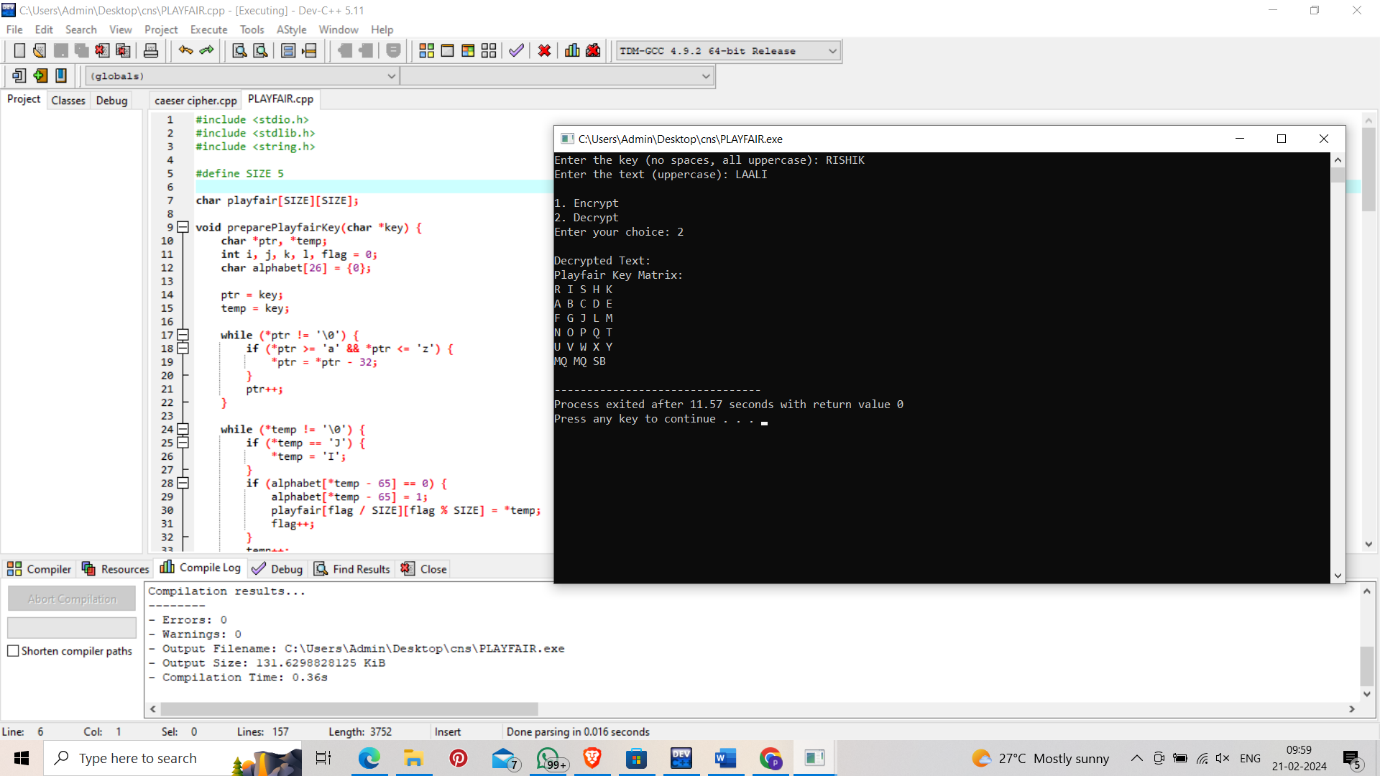
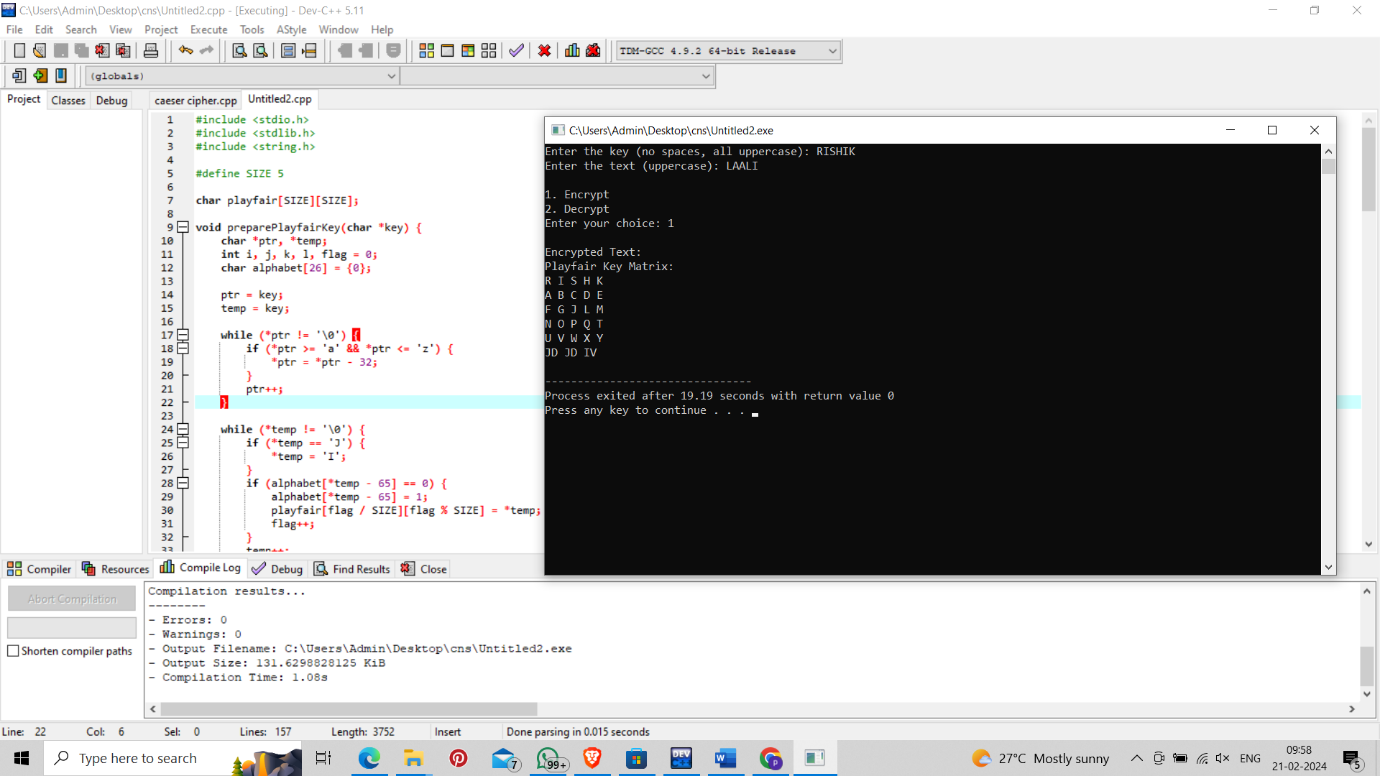
default:

printf("\nInvalid choice!\n");

}

return 0;

}

**OUTPUT**