**EXP-9:**

**PROGRAM:**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define SIZE 5

#define MAX\_TEXT 1000

char matrix[SIZE][SIZE];

int used[26] = {0};

typedef struct {

int row;

int col;

} Position;

Position find\_position(char ch) {

Position pos;

if (ch == 'J') ch = 'I'; // Treat I and J as same

for (int i = 0; i < SIZE; i++) {

for (int j = 0; j < SIZE; j++) {

if (matrix[i][j] == ch) {

pos.row = i;

pos.col = j;

return pos;

}

}

}

pos.row = pos.col = -1;

return pos;

}

void generate\_matrix(const char \*keyword) {

int k = 0;

memset(used, 0, sizeof(used));

for (int i = 0; keyword[i] != '\0'; i++) {

char ch = toupper(keyword[i]);

if (ch == 'J') ch = 'I';

if (!isalpha(ch) || used[ch - 'A']) continue;

matrix[k / SIZE][k % SIZE] = ch;

used[ch - 'A'] = 1;

k++;

}

for (char ch = 'A'; ch <= 'Z'; ch++) {

if (ch == 'J') continue;

if (!used[ch - 'A']) {

matrix[k / SIZE][k % SIZE] = ch;

used[ch - 'A'] = 1;

k++;

}

}

}

void decrypt\_pair(char a, char b, char \*out) {

Position p1 = find\_position(a);

Position p2 = find\_position(b);

if (p1.row == p2.row) {

out[0] = matrix[p1.row][(p1.col + SIZE - 1) % SIZE];

out[1] = matrix[p2.row][(p2.col + SIZE - 1) % SIZE];

} else if (p1.col == p2.col) {

out[0] = matrix[(p1.row + SIZE - 1) % SIZE][p1.col];

out[1] = matrix[(p2.row + SIZE - 1) % SIZE][p2.col];

} else {

out[0] = matrix[p1.row][p2.col];

out[1] = matrix[p2.row][p1.col];

}

}

void decrypt\_playfair(const char \*ciphertext, const char \*keyword) {

generate\_matrix(keyword);

printf("Decrypted message: ");

int len = strlen(ciphertext);

for (int i = 0; i < len; i += 2) {

char a = toupper(ciphertext[i]);

char b = toupper(ciphertext[i + 1]);

if (!isalpha(a) || !isalpha(b)) continue;

if (a == 'J') a = 'I';

if (b == 'J') b = 'I';

char out[3] = {0};

decrypt\_pair(a, b, out);

printf("%c%c", out[0], out[1]);

}

printf("\n");

}

int main() {

const char \*ciphertext = "KXJEYUREBEZWEHEWRYTUHEYFSKREHEGOYFIWTTTUOLKSYCAJPOBOTEIZONTXBYBNTGONEYCUZWRGDSONSXBOUYWRHEBAAHYUSEDQ";

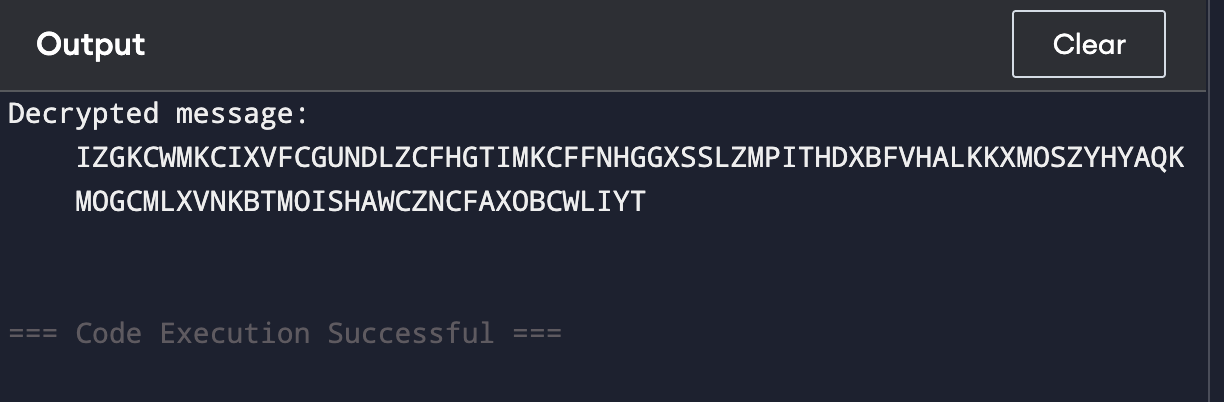
const char \*keyword = "MONARCHY";

decrypt\_playfair(ciphertext, keyword);

return 0;

}

**OUTPUT:**

****