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

void create_playfair_matrix(char *key, char matrix[5][5]) {
    int i, k;
    char temp[26] = {0};

    for (i = 0, k = 0; i < strlen(key); i++) {
        if (key[i] != 'J') {
            if (temp[toupper(key[i]) - 'A'] == 0) {
                temp[toupper(key[i]) - 'A'] = 1;
                matrix[k / 5][k % 5] = toupper(key[i]);
                k++;
            }
        }
    }

    for (i = 0; i < 26; i++) {
        if (temp[i] == 0) {
            if (i == 'J' - 'A') {
                continue;
            }
            matrix[k / 5][k % 5] = 'A' + i;
            k++;
        }
    }
}

void playfair_cipher(char *plaintext, char *key, char *ciphertext) {
    char matrix[5][5];
    int i, k, row1, col1, row2, col2;
    int plaintext_len = strlen(plaintext);

    create_playfair_matrix(key, matrix);

    for (i = 0, k = 0; i < plaintext_len; i += 2) {
        for (row1 = 0; row1 < 5; row1++) {
            for (col1 = 0; col1 < 5; col1++) {
                if (matrix[row1][col1] == toupper(plaintext[i])) {
                    break;
                }
            }
            if (col1 < 5) {
                break;
            }
        }
        for (row2 = 0; row2 < 5; row2++) {
            for (col2 = 0; col2 < 5; col2++) {
                if (matrix[row2][col2] == toupper(plaintext[i + 1])) {
                    break;
                }
            }
            if (col2 < 5) {
                break;
```

```
            }
        }

        if (row1 == row2) {
            ciphertext[k++] = matrix[row1][(col1 + 1) % 5];
            ciphertext[k++] = matrix[row2][(col2 + 1) % 5];
        } else if (col1 == col2) {
            ciphertext[k++] = matrix[(row1 + 1) % 5][col1];
            ciphertext[k++] = matrix[(row2 + 1) % 5][col2];
        } else {
            ciphertext[k++] = matrix[row1][col2];
            ciphertext[k++] = matrix[row2][col1];
        }
    }
    ciphertext[k] = '\0';
}

void playfair_decipher(char *ciphertext, char *key, char *plaintext) {
    char matrix[5][5];
    int i, k, row1, col1, row2, col2;
    int ciphertext_len = strlen(ciphertext);

    create_playfair_matrix(key, matrix);

    for (i = 0, k = 0; i < ciphertext_len; i += 2) {
        for (row1 = 0; row1 < 5; row1++) {
            for (col1 = 0; col1 < 5; col1++) {
                if (matrix[row1][col1] == toupper(ciphertext[i])) {
                    break;
                }
            }
            if (col1 < 5) {
                break;
            }
        }
        for (row2 = 0; row2 < 5; row2++) {
            for (col2 = 0; col2 < 5; col2++) {
                if (matrix[row2][col2] == toupper(ciphertext[i + 1])) {
                    break;
                }
            }
            if (col2 < 5) {
                break;
            }
        }

        if (row1 == row2) {
            plaintext[k++] = matrix[row1][(col1 + 4) % 5];
            plaintext[k++] = matrix[row2][(col2 + 4) % 5];
        } else if (col1 == col2) {
            plaintext[k++] = matrix[(row1 + 4) % 5][col1];
            plaintext[k++] = matrix[(row2 + 4) % 5][col2];
        } else {
            plaintext[k++] = matrix[row1][col2];
            plaintext[k++] = matrix[row2][col1];
```

```
        }
    }
    plaintext[k] = '\0';
}

int main() {
    char plaintext[100];
    char key[100];
    char ciphertext[sizeof(plaintext) * 2];
    char decryptedtext[sizeof(plaintext) * 2];

    printf("Enter the plaintext: ");
    scanf("%99s", plaintext);

    printf("Enter the key: ");
    scanf("%99s", key);

    playfair_cipher(plaintext, key, ciphertext);
    printf("Ciphertext: %s\n", ciphertext);

    playfair_decipher(ciphertext, key, decryptedtext);
    printf("Decrypted text: %s\n", decryptedtext);

    return 0;
}
```