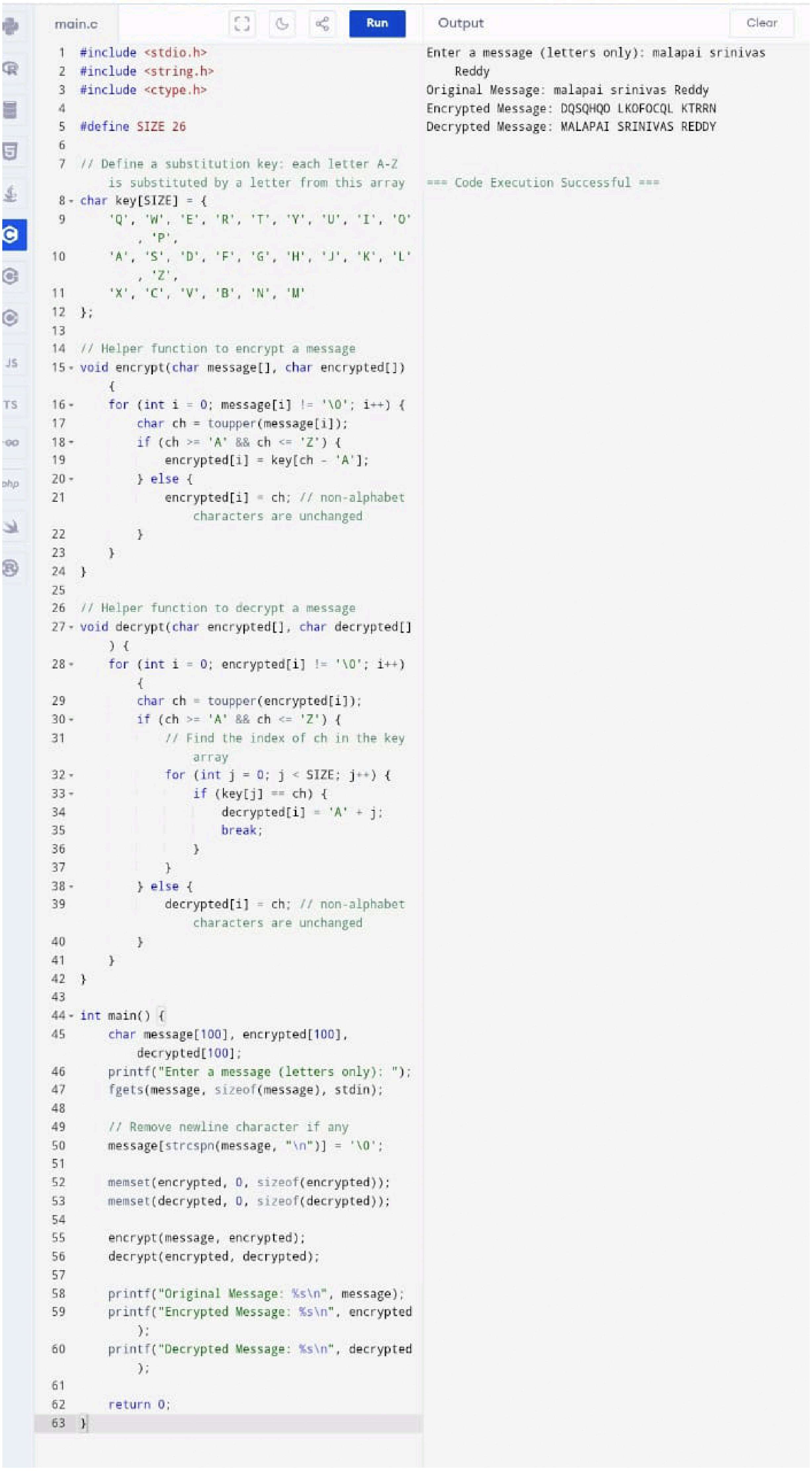
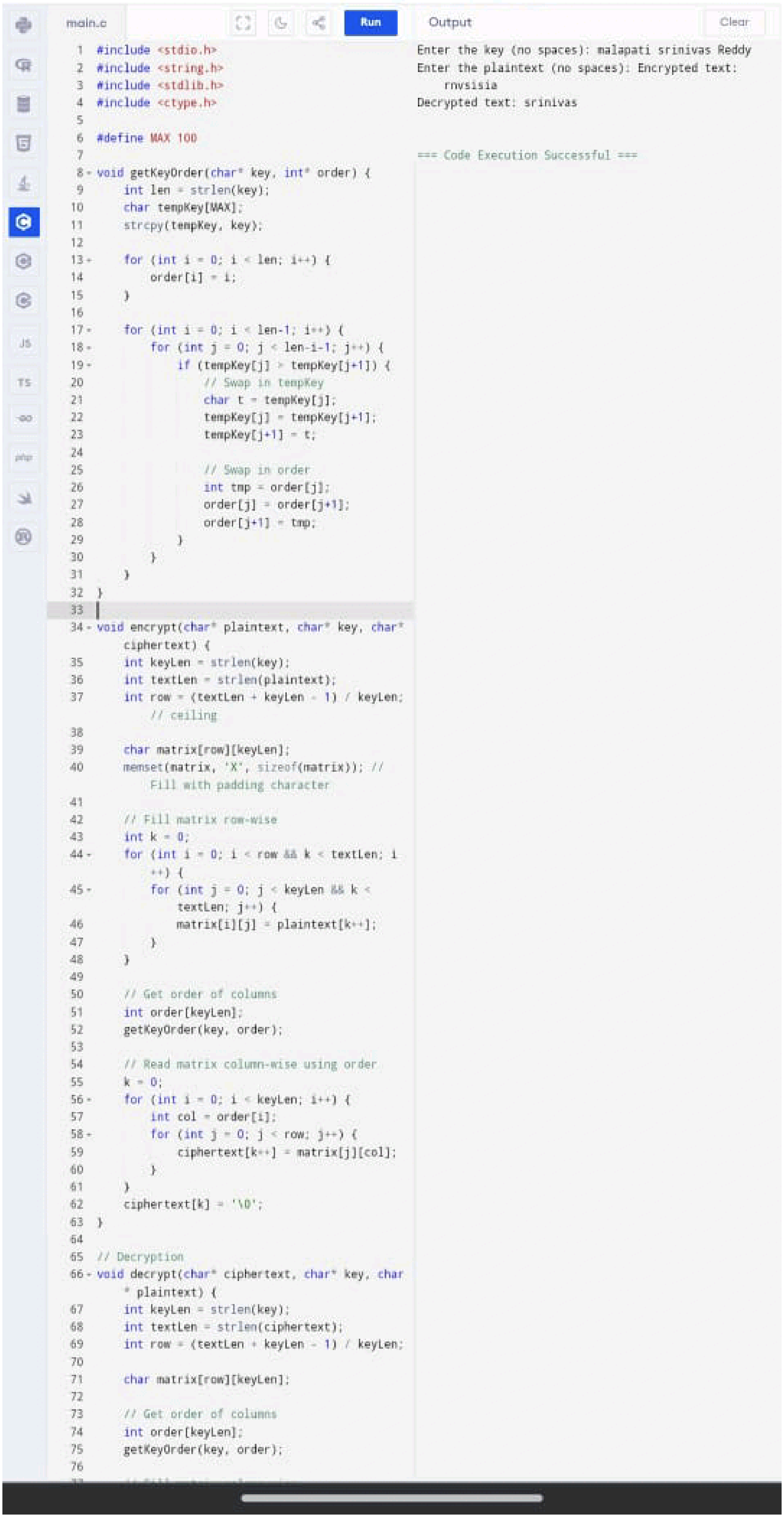


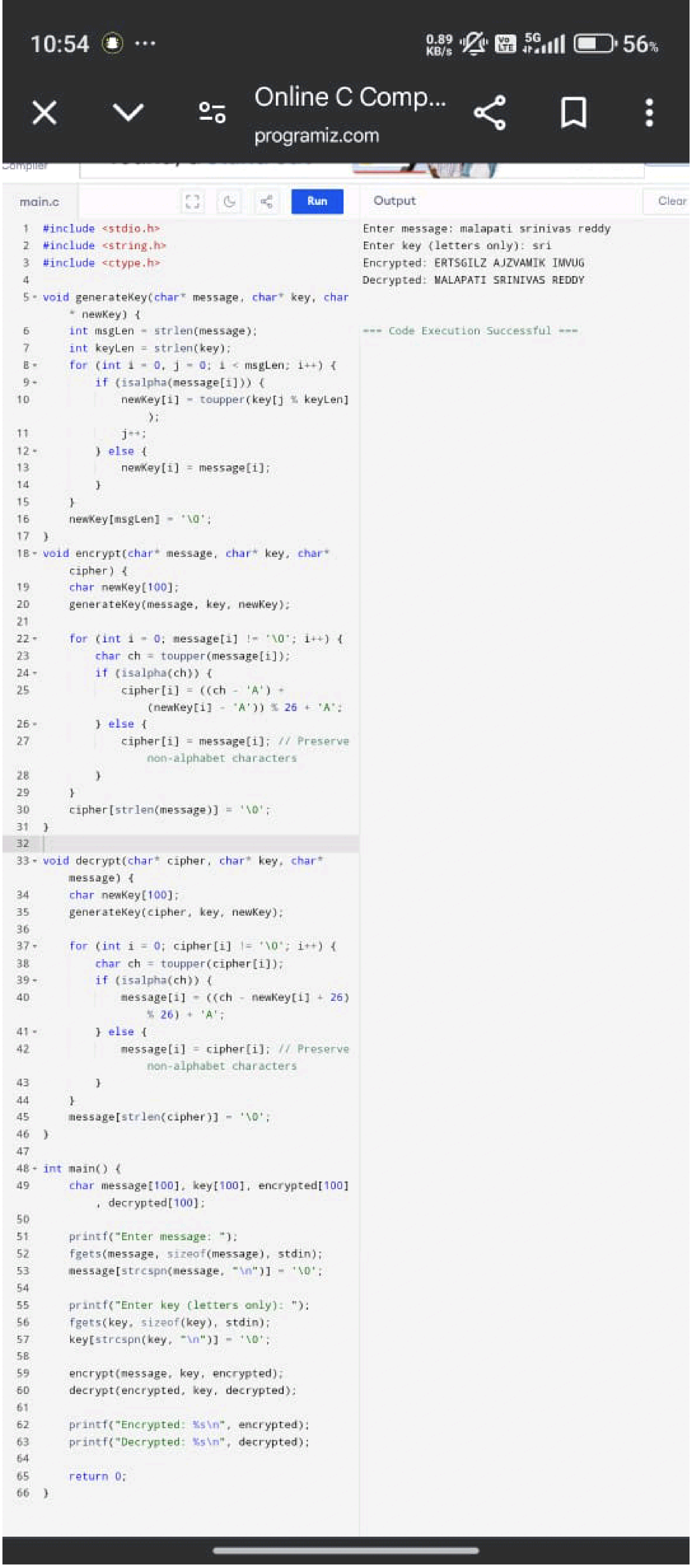
```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define SIZE 26

// Define a substitution key: each letter A-Z is substituted by a letter from this array
char key[SIZE] = {
    'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P',
    'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'Z',
    'X', 'C', 'V', 'B', 'N', 'M'
};

// Helper function to encrypt a message
void encrypt(char message[], char encrypted[]) {
    for (int i = 0; message[i] != '\0'; i++) {
        char ch = toupper(message[i]);
        if (ch >= 'A' && ch <= 'Z') {
            encrypted[i] = key[ch - 'A'];
        } else {
            encrypted[i] = ch; // non-alphabet characters are unchanged
        }
    }
}

// Helper function to decrypt a message
void decrypt(char encrypted[], char decrypted[]) {
    for (int i = 0; encrypted[i] != '\0'; i++) {
        char ch = toupper(encrypted[i]);
        if (ch >= 'A' && ch <= 'Z') {
            // Find the index of ch in the key array
            for (int j = 0; j < SIZE; j++) {
                if (key[j] == ch) {
                    decrypted[i] = 'A' + j;
                    break;
                }
            }
        } else {
            decrypted[i] = ch; // non-alphabet characters are unchanged
        }
    }
}

int main() {
    char message[100], encrypted[100], decrypted[100];
    printf("Enter a message (letters only): ");
    fgets(message, sizeof(message), stdin);

    // Remove newline character if any
    message[strcspn(message, "\n")] = '\0';

    memset(encrypted, 0, sizeof(encrypted));
    memset(decrypted, 0, sizeof(decrypted));

    encrypt(message, encrypted);
    decrypt(encrypted, decrypted);

    printf("Original Message: %s\n", message);
    printf("Encrypted Message: %s\n", encrypted);
    printf("Decrypted Message: %s\n", decrypted);

    return 0;
}
```

```
Enter a message (letters only): malapai srinivas
    Reddy
Original Message: malapai srinivas Reddy
Encrypted Message: DQSQHQO LKOFOCQL KTRRN
Decrypted Message: MALAPAI SRINIVAS REDDY


=== Code Execution Successful ===
```



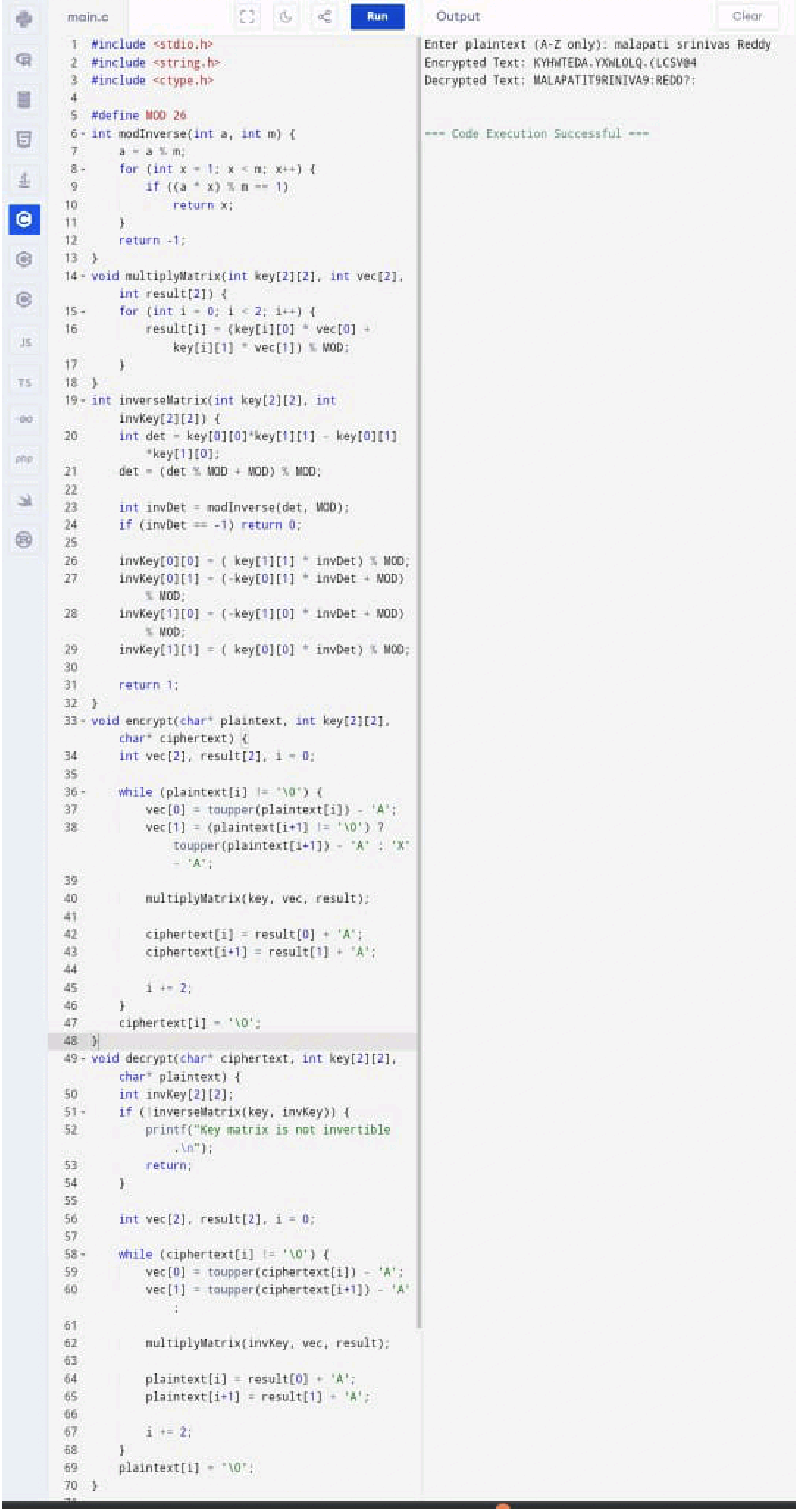
```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#define MAX 100

void getKeyOrder(char* key, int* order) {
    int len = strlen(key);
    char tempKey[MAX];
    strcpy(tempKey, key);

    for (int i = 0; i < len; i++) {
        order[i] = i;
    }

    for (int i = 0; i < len-1; i++) {
        for (int j = 0; j < len-i-1; j++) {
            if (tempKey[j] > tempKey[j+1]) {
                // Swap in tempKey
                char t = tempKey[j];
                tempKey[j] = tempKey[j+1];
                tempKey[j+1] = t;

                // Swap in order
                int tmp = order[j];
                order[j] = order[j+1];
                order[j+1] = tmp;
            }
        }
    }
}

void encrypt(char* plaintext, char* key, char* ciphertext) {
    int keyLen = strlen(key);
    int textLen = strlen(plaintext);
    int row = (textLen + keyLen - 1) / keyLen; // ceiling

    char matrix[row][keyLen];
    memset(matrix, 'X', sizeof(matrix)); // Fill with padding character

    // Fill matrix row-wise
    int k = 0;
    for (int i = 0; i < row && k < textLen; i++) {
        for (int j = 0; j < keyLen && k < textLen; j++) {
            matrix[i][j] = plaintext[k++];
        }
    }

    // Get order of columns
    int order[keyLen];
    getKeyOrder(key, order);

    // Read matrix column-wise using order
    k = 0;
    for (int i = 0; i < keyLen; i++) {
        int col = order[i];
        for (int j = 0; j < row; j++) {
            ciphertext[k++] = matrix[j][col];
        }
    }
    ciphertext[k] = '\0';
}

// Decryption
void decrypt(char* ciphertext, char* key, char* plaintext) {
    int keyLen = strlen(key);
    int textLen = strlen(ciphertext);
    int row = (textLen + keyLen - 1) / keyLen;

    char matrix[row][keyLen];

    // Get order of columns
    int order[keyLen];
    getKeyOrder(key, order);

```

```
Enter the key (no spaces): malapati srinivas Reddy
Enter the plaintext (no spaces): Encrypted text:
    rnvsisia
Decrypted text: srinivas


=== Code Execution Successful ===
```



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

void generateKey(char* message, char* key, char* newKey) {
    int msgLen = strlen(message);
    int keyLen = strlen(key);
    for (int i = 0, j = 0; i < msgLen; i++) {
        if (isalpha(message[i])) {
            newKey[i] = toupper(key[j % keyLen]);
            j++;
        } else {
            newKey[i] = message[i];
        }
    }
    newKey[msgLen] = '\0';
}
void encrypt(char* message, char* key, char* cipher) {
    char newKey[100];
    generateKey(message, key, newKey);

    for (int i = 0; message[i] != '\0'; i++) {
        char ch = toupper(message[i]);
        if (isalpha(ch)) {
            cipher[i] = ((ch - 'A') + (newKey[i] - 'A')) % 26 + 'A';
        } else {
            cipher[i] = message[i]; // Preserve non-alphabet characters
        }
    }
    cipher[strlen(message)] = '\0';
}

void decrypt(char* cipher, char* key, char* message) {
    char newKey[100];
    generateKey(cipher, key, newKey);

    for (int i = 0; cipher[i] != '\0'; i++) {
        char ch = toupper(cipher[i]);
        if (isalpha(ch)) {
            message[i] = ((ch - newKey[i] + 26) % 26) + 'A';
        } else {
            message[i] = cipher[i]; // Preserve non-alphabet characters
        }
    }
    message[strlen(cipher)] = '\0';
}

int main() {
    char message[100], key[100], encrypted[100], decrypted[100];

    printf("Enter message: ");
    fgets(message, sizeof(message), stdin);
    message[strcspn(message, "\n")] = '\0';

    printf("Enter key (letters only): ");
    fgets(key, sizeof(key), stdin);
    key[strcspn(key, "\n")] = '\0';

    encrypt(message, key, encrypted);
    decrypt(encrypted, key, decrypted);

    printf("Encrypted: %s\n", encrypted);
    printf("Decrypted: %s\n", decrypted);

    return 0;
}
```

```
Enter message: malapati srinivas reddy
Enter key (letters only): sri
Encrypted: ERTSGILZ AJZVAMIK IMVUG
Decrypted: MALAPATI SRINIVAS REDDY


=== Code Execution Successful ===
```



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

#define MOD 26
int modInverse(int a, int m) {
    a = a % m;
    for (int x = 1; x < m; x++) {
        if ((a * x) % m == 1)
            return x;
    }
    return -1;
}
void multiplyMatrix(int key[2][2], int vec[2],
    int result[2]) {
    for (int i = 0; i < 2; i++) {
        result[i] = (key[i][0] * vec[0] +
            key[i][1] * vec[1]) % MOD;
    }
}
int inverseMatrix(int key[2][2], int
    invKey[2][2]) {
    int det = key[0][0]*key[1][1] - key[0][1]
        *key[1][0];
    det = (det % MOD + MOD) % MOD;

    int invDet = modInverse(det, MOD);
    if (invDet == -1) return 0;

    invKey[0][0] = ( key[1][1] * invDet) % MOD;
    invKey[0][1] = (-key[0][1] * invDet + MOD)
        % MOD;
    invKey[1][0] = (-key[1][0] * invDet + MOD)
        % MOD;
    invKey[1][1] = ( key[0][0] * invDet) % MOD;

    return 1;
}
void encrypt(char* plaintext, int key[2][2],
    char* ciphertext) {
    int vec[2], result[2], i = 0;

    while (plaintext[i] != '\0') {
        vec[0] = toupper(plaintext[i]) - 'A';
        vec[1] = (plaintext[i+1] != '\0') ?
            toupper(plaintext[i+1]) - 'A' : 'X'
            - 'A';

        multiplyMatrix(key, vec, result);

        ciphertext[i] = result[0] + 'A';
        ciphertext[i+1] = result[1] + 'A';

        i += 2;
    }
    ciphertext[i] = '\0';
}
void decrypt(char* ciphertext, int key[2][2],
    char* plaintext) {
    int invKey[2][2];
    if (!inverseMatrix(key, invKey)) {
        printf("Key matrix is not invertible
            .\n");
        return;
    }

    int vec[2], result[2], i = 0;

    while (ciphertext[i] != '\0') {
        vec[0] = toupper(ciphertext[i]) - 'A';
        vec[1] = toupper(ciphertext[i+1]) - 'A'
            ;

        multiplyMatrix(invKey, vec, result);

        plaintext[i] = result[0] + 'A';
        plaintext[i+1] = result[1] + 'A';

        i += 2;
    }
    plaintext[i] = '\0';
}
```

```
Enter plaintext (A-Z only): malapati srinivas Reddy
Encrypted Text: KYHWTEDA.YXWLOLQ.(LCSV@4
Decrypted Text: MALAPATIT9RINIVA9:REDD?:


=== Code Execution Successful ===
```