```
8 6
                                                             Output
       main.c
       1 #include <stdio.h>
                                                            Enter a message (letters only): malapai srinivas
@
       2 #include <string.h>
       3 #include <ctype.h>
                                                            Original Message: malapai srinivas Reddy
                                                            Encrypted Message: DQSQHQO LKOFOCQL KTRRN
Ħ
       4
       5 #define SIZE 26
                                                            Decrypted Message: MALAPAI SRINIVAS REDDY
5
       7 // Define a substitution key: each letter A-Z
              is substituted by a letter from this array
                                                           === Code Execution Successful ===
$
       8 - char key[SIZE] = {
       9
              'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O'
0
                  , 'P',
      10
              'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L'
0
                  , 'Z',
              'X', 'C', 'V', 'B', 'N', 'M'
      11
      12 };
0
      13
      14 // Helper function to encrypt a message
JS
      15 - void encrypt(char message[], char encrypted[])
TS
      16 -
              for (int i = 0; message[i] != '\0'; i++) {
      17
                  char ch = toupper(message[i]);
                  if (ch >= 'A' && ch <= 'Z') {
      18 -
                      encrypted[i] = key[ch - 'A'];
      19
      20 -
                  } else {
php
                      encrypted[i] = ch; // non-alphabet
      21
                          characters are unchanged
3
      22
                  3
      23
              }
3
      24 }
      25
      26 // Helper function to decrypt a message
      27 - void decrypt(char encrypted[], char decrypted[]
              ) {
      28 -
              for (int i = 0; encrypted[i] != '\0'; i++)
      29
                  char ch = toupper(encrypted[i]);
      30 -
                  if (ch >= 'A' && ch <= 'Z') {
      31
                      // Find the index of ch in the key
                          array
                      for (int j = 0; j < SIZE; j++) {
      33 -
                          if (key[j] == ch) {
                              decrypted[i] = 'A' + j;
      34
      35
                              break:
      36
      37
                      1
      38 -
                  } else {
      39
                      decrypted[i] = ch; // non-alphabet
                          characters are unchanged
      40
                  }
      41
              }
      42 }
      43
      44 - int main() {
      45
              char message[100], encrypted[100],
                  decrypted[100];
      46
              printf("Enter a message (letters only): ");
      47
              fgets(message, sizeof(message), stdin);
      48
      49
              // Remove newline character if any
              message[strcspn(message, "\n")] = '\0';
      50
      51
              memset(encrypted, 0, sizeof(encrypted));
      52
      53
              memset(decrypted, 0, sizeof(decrypted));
      54
      55
              encrypt(message, encrypted);
      56
              decrypt(encrypted, decrypted);
      57
      58
              printf("Original Message: %s\n", message);
      59
              printf("Encrypted Message: %s\n", encrypted
      60
               printf("Decrypted Message: %s\n", decrypted
                  );
      61
      62
              return 0;
      63 }
```

```
main.c
                                                    Run
         1 #include <stdio.h>
                                                             Enter the key (no spaces): malapati srinivas Reddy
Q
         2 #include <string.h>
                                                             Enter the plaintext (no spaces): Encrypted text:
         3 Winclude <stdlib.h>
                                                                 rnysisia
         4 #include <ctype.h>
                                                             Decrypted text: srinivas
6 #define MAX 100
日
                                                             === Code Execution Successful ===
         8- void getKeyOrder(char* key, int* order) {
               int len = strlen(key);
        10
                char tempKey[MAX];
        11
                strcpy(tempKey, key);
        12
        13+
                for (int i = 0; i < len; i++) {
        14
                    order[i] = i;
        15
        16
        17 -
               for (int i = 0; i < len-1; i++) (
        18 -
                    for (int j = 0; j < len-i-1; j++) {
                        if (tempKey[j] > tempKey[j+1]) {
        19 -
TS
        20
                            // Swap in tempKey
        21
                            char t = tenpKey[j];
                            tempKey[j] = tempKey[j+1];
        22
        23
                            tempKey[j+1] = t;
        24
        25
                            // Swap in order
        26
                            int tmp = order[j];
        27
                            order[j] = order[j+1];
        28
                            order[j+1] = tmp;
00
        29
        30
        31
        32
        33
        34 - void encrypt(char* plaintext, char* key, char*
                ciphertext) {
                int keyLen = strlen(key);
        35
        36
                int textLen = strlen(plaintext);
        37
                int row = (textLen + keyLen - 1) / keyLen;
                    // ceiling
        38
        39
                char matrix[row][keyLen];
        40
                nemset(matrix, 'X', sizeof(matrix)); //
                    Fill with padding character
       41
        42
                // Fill matrix row-wise
        43
                int k = 0;
        44 -
                for (int i = 0; i < row & k < textLen; i
                    ++) {
                    for (int j = 0; j < keyLen && k <
        45 -
                        textLen; j++) {
                        matrix[i][j] = plaintext[k++];
        46
        47
        48
        49
        50
                // Get order of columns
        51
                int order[keyLen];
        52
                getKeyOrder(key, order);
        53
        54
                // Read matrix column-wise using order
        55
        56 -
                for (int i = 0; i < keyLen; i++) {
        57
                    int col = order[i];
        58 -
                    for (int j = 0; j < row; <math>j \leftrightarrow) {
        59
                        ciphertext[k++] = matrix[j][col];
        60
        61
        62
                ciphertext[k] = '\0';
        63
           3
        54
        65 // Decryption
        66- void decrypt(char* ciphertext, char* key, char
                * plaintext) {
        67
                int keyLen = strlen(key);
                int textLen = strlen(ciphertext);
        68
        69
                int row = (textLen + keyLen - 1) / keyLen;
        70
        71
                char matrix[row][keyLen];
        72
        73
                // Get order of columns
        74
                int order[keyLen];
        75
                getKeyOrder(key, order);
        76
```



programiz.com

3 6 Run main.c 1 #include <stdio.h> Enter message: malapati srinivas reddy 2 #include <string.h> Enter key (letters only): sri #include <ctype.h> Encrypted: ERTSGILZ AJZVAMIK IMVUG Decrypted: MALAPATI SRINIVAS REDDY 5 - void generateKey(char* message, char* key, char * newKey) { int msgLen = strlen(message); --- Code Execution Successful --int keyLen = strlen(key); for (int i = 0, j = 0; i < msgLen; i++) { B = if (isalpha(message[i])) { 0.-10 newKey[i] = toupper(key[j % keyLen] 11 j++; 12 +) else (13 newKey[i] = message[i]; 14 15 16 newKey[msgLen] = '\0'; 17) 18 - void encrypt(char* message, char* key, char* cipher) { 19 char newKey[100]; 20 generateKey(message, key, newKey); 21 22 for (int i = 0; message[i] != '\0'; i++) { 23 char ch = toupper(message[i]); 24 if (isalpha(ch)) { cipher[i] = ((ch - 'A') + (newKey[i] - 'A')) % 26 + 'A'; 26 -) else (27 cipher[i] = message[i]; // Preserve non-alphabet characters 28 20 30 cipher[strlen(message)] = '\0'; 31) 32 33 - void decrypt(char* cipher, char* key, char* message) { 34 char newKey[100]; 35 generateKey(cipher, key, newKey); 36 for (int i = 0; cipher[i] != '\0'; i++) { 37 char ch = toupper(cipher[i]); 39 if (isalpha(ch)) (40 message[i] = ((ch - newKey[i] + 26) % 26) + 'A'; 41 -} else { 42 message[i] = cipher[i]; // Preserve non-alphabet characters 43 44 45 message[strlen(cipher)] = '\0'; 46 } 47 48 + int main() { 49 char message[100], key[100], encrypted[100] , decrypted[100]; 50 51 printf("Enter message: "); 52 fgets(message, sizeof(message), stdin); message[strcspn(message, "\n")] = '\0'; 53 54 55 printf("Enter key (letters only): "); 56 fgets(key, sizeof(key), stdin); 57 key[strcspn(key, "\n")] = '\0'; 5.R 59 encrypt(message, key, encrypted); 60 decrypt(encrypted, key, decrypted); 61 62 printf("Encrypted: %s\n", encrypted); 63 printf("Decrypted: %s\n", decrypted); 64 65 return 0; 66 }

```
[] (5 ≪ Run
                                                            Output
       main.c
       1 #include <stdio.h>
                                                           Enter plaintext (A-Z only): malapati srinivas Reddy
       2 #include <string.h>
                                                           Encrypted Text: KYHNTEDA.YXWLOLQ.(LCSV84
       3 #include <ctype.h>
                                                           Decrypted Text: MALAPATIT9RINIVA9:REDD7:
Ħ
       5 #define MOD 26
       6 - int modInverse(int a, int m) {
                                                           --- Code Execution Successful ---
O
              a = a % m;
               for (int x = 1; x < m; x++) {
       8 -
4
                 if ((a * x) % n == 1)
       9
       10
                     return x;
0
       11
       12
              return -1:
0
       13 }
       14 - void multiplyWatrix(int key[2][2], int vec[2],
               int result[2]) {
0
       15 -
               for (int i = 0; i < 2; i ++) {
                  result[i] = (key[i][0] * vec[0] +
js.
                     key[i][1] * vec[1]) % MOD;
       17
       18 }
75
       19 - int inverseMatrix(int key[2][2], int
               invKey[2][2]) {
               int det = key[0][0]*key[1][1] - key[0][1]
                  *key[1][0];
php
       21
               det = (det % MOD + MOD) % MOD;
       22
3)
       23
               int invDet = modInverse(det, MOD);
               if (invDet == -1) return 0;
       24
60
       25
               invKey[0][0] = ( key[1][1] * invDet) % MOD;
       26
       27
               invKey[0][1] - (-key[0][1] * invDet + NOD)
                  W MOD:
       28
               invKey[1][0] = (-key[1][0] * invDet + NOD)
                  * MOD:
       29
               invKey[1][1] = ( key[0][0] * invDet) % MOO;
       30
       31
      32 }
       33 - void encrypt(char* plaintext, int key[2][2],
               char* ciphertext) {
       34
               int vec[2], result[2], i = 0;
       35
       36 -
              while (plaintext[i] != "\0") (
                  vec[0] = toupper(plaintext[i]) - 'A';
       37
       38
                   vec[1] = (plaintext[i+1] != '\0') ?
                       toupper(plaintext[i+1]) - "A" : 'X"
                       - "A";
       30
       40
                  multiplyMatrix(key, vec, result);
       41
       42
                   ciphertext[i] = result[0] + 'A';
                   ciphertext[i+1] = result[1] + "A";
       43
       44
       45
                   1 += 2;
       46
               ciphertext[i] = '\0';
       47
     48 )
       49 - void decrypt(char* ciphertext, int key[2][2],
               char* plaintext) {
       50
               int invKey[2][2];
       51 -
               if (|inverseMatrix(key, invKey)) {
       52
                  printf("Key matrix is not invertible
                       .\n");
       53
                   return;
       54
       55
       56
               int vec[2], result[2], i = 0;
       57
       58 -
               while (ciphertext[i] != '\0') {
                   vec[0] = toupper(ciphertext[i]) - 'A';
       59
       60
                   vec[1] = toupper(ciphertext[i+1]) - 'A'
                  multiplyMatrix(invKey, vec, result);
       62
       63
       64
                   plaintext[i] = result[0] + 'A';
                   plaintext[i+1] = result[1] + 'A';
       65
       66
       67
                   i += 2;
       58
       69
               plaintext[i] = "\0";
       70 1
```