```
main.c
                                                      Share
                                                                    Run
                                                                              Output
13 -
            } else if (islower(ch)) {
                                                                            Enter a message to encrypt: hello world
14
                ciphertext[i] = ((ch - 'a' + key) % 26) + 'a';
                                                                            Enter key (1-25): 5
15 -
            } else {
                                                                            Encrypted message: mjqqt btwqi
16
                ciphertext[i] = ch;
17
18
       }
                                                                             === Code Execution Successful ===
19
       ciphertext[i] = '\0';
20 }
21 - int main () {
22
        char plaintext [MAX_LEN];
23
        char ciphertext [MAX_LEN];
24
        int key;
25
        printf("Enter a message to encrypt: ");
26
        fgets(plaintext, MAX_LEN, stdin);
27
        size_t len = strlen(plaintext);
28 -
        if (len > 0 && plaintext [len - 1] == '\n') {
29
            plaintext [len - 1] = '\0';
30
31
        printf("Enter key (1-25): ");
32
        scanf("%d", &key);
33 -
        if (key < 1 || key > 25) {
34
            printf("Invalid key. Must be between 1 and 25.\n");
35
            return 1;
36
        }
37
        encrypt (plaintext, ciphertext, key);
38
        printf("Encrypted message: %s\n", ciphertext);
```

JS

```
Run
                                                                                    Output
                                                                                                                                                      Clear
                                                 [] 6
       main.c
       23 -
               for (int i = 0; i < 26; i++) {
                                                                                  Enter the plaintext: hello world
                                                                                   Enter 26-letter substitution key (A-Z): efghijklmnopqrstuvwxyzabcd
A
                   if (!isalpha(key[i]))
       24
                       return 0;
       25
                                                                                   Encrypted message: lipps asvph
                   int index = toupper(key[i]) - 'A';
if (freq[index]++)
       26
       27
       28
                       return 0;
                                                                                   === Code Execution Successful ===
ⅎ
       29
       30
               return 1;
$
       31 }
       32 - int main () {
               char plaintext [MAX_LEN], ciphertext [MAX_LEN];
0
       33
               char key [27];
       34
               printf("Enter the plaintext: ");
       35
(
       36
               fgets(plaintext, MAX_LEN, stdin);
       37
               size_t len = strlen(plaintext);
(
               if (len > 0 && plaintext [len - 1] == '\n')
       38
       39
                   plaintext [len - 1] = '\0';
JS
               printf("Enter 26-letter substitution key (A-Z): ");
       40
       41
               fgets(key, 27, stdin);
       42 -
               if (!isValidKey(key)) {
TS
                   printf("Invalid key! Key must be 26 unique alphabetic
       43
                       letters.\n");
E
                   return 1;
       44
                                                                                                                                                   Activate W
       45
1
                                                                                                                                                   Go to Settings
               encrypt (plaintext, ciphertext, key);
       46
       47
               printf("Encrypted message: %s\n", ciphertext);
```

```
∝ Share
                                                                          Run
                                                                                     Output
       main.c
      16
                                                                                   Key: MONARCHY
       17
                                                                                   Plaintext: BOMMU
               for (i = 0; i < 26; i++) {
       18 -
                                                                                   Encrypted: HRHRHR
                   if (i + 'a' == 'j' || used[i]) continue;
       19
                   m[k / 5][k \% 5] = i + 'a';
       20
                                                                                   === Code Execution Successful ===
       21
回
       22
              }
      23
              printf("Encrypted: ");
      24 -
               for (i = 0; text[i]; i += 2) {
                   char a = text[i], b = text[i + 1] ? text[i + 1] : 'x';
      25
                   if (a == 'j') a = 'i'; if (b == 'j') b = 'i';
      26
       27
                   if (a == b) b = 'x';
      28
(
      29
                   for (j = 0; j < 5; j++)
      30 -
                       for (k = 0; k < 5; k++) {
0
      31
                           if (m[j][k] == a) \{ r1 = j; c1 = k; \}
      32
                           if (m[j][k] == b) \{ r2 = j; c2 = k; \}
      33
                      }
      34
                   if (r1 == r2)
      35
TS
                       printf("%c%c", m[r1][(c1 + 1) % 5], m[r2][(c2 + 1) % 5]
      36
                           );
      37
                   else if (c1 == c2)
                       printf("%c%c", m[(r1 + 1) % 5][c1], m[(r2 + 1) % 5][c2]
      38
                           );
      39
                   else
```

```
∝ Share
                                                                  Run
                                                                            Output
main.c
                                                                         * Enter the plaintext: sudharshan
      for (int i = 0; i < len; i++) {
                                                                           Enter the keyword: bommu
          char ch = plaintext[i];
                                                                           Encrypted text: tipturshan
          if (isalpha(ch)) {
               int shift = fullKey[i] - 'A';
               if (isupper(ch)) {
                                                                           === Code Execution Successful ===
                  ciphertext[i] = ((ch - 'A' + shift) % 26) + 'A';
               } else {
                   ciphertext[i] = ((ch - 'a' + shift) % 26) + 'a';
               }
          } else {
               ciphertext[i] = ch;
       }
       ciphertext[len] = '\0';
38 }
39 - int main() {
       char plaintext[MAX_LEN], key[MAX_LEN], ciphertext[MAX_LEN];
       printf("Enter the plaintext: ");
       fgets(plaintext, MAX_LEN, stdin);
       plaintext[strcspn(plaintext, "\n")] = '\0';
       printf("Enter the keyword: ");
       fgets(key, MAX_LEN, stdin);
       key[strcspn(key, "\n")] = '\0';
       encrypt(plaintext, key, ciphertext);
       printf("Encrypted text: %s\n", ciphertext);
```

3

4 -

5

6 -7

8 -

9 0 -

1 32

33 -

34

35 36

37

10 11

12

43

14 45

46 47

48

```
♦ Share
                                                                              Output
                                                                   Run
main.c
            printf("Decryption not possible. No modular inverse for a =
38
                                                                            Enter plaintext: hello
                                                                            Enter values of a and b (a must be coprime with 26): 5 8
                %d.\n", a);
                                                                            Encrypted text: rclla
39
            return;
                                                                            Decrypted text: hello
40
        }
41 -
        for (int i = 0; ciphertext[i] != '\0'; i++) {
            decrypted[i] = decryptChar(ciphertext[i], aInv, b);
42
                                                                            === Code Execution Successful ===
43
        decrypted[strlen(ciphertext)] = '\0';
44
45 }
46 - int main() {
47
        char plaintext[1000], ciphertext[1000], decrypted[1000];
48
        int a, b;
        printf("Enter plaintext: ");
49
        fgets(plaintext, sizeof(plaintext), stdin);
50
        plaintext[strcspn(plaintext, "\n")] = '\0';
51
        printf("Enter values of a and b (a must be coprime with 26): ");
53
        scanf("%d %d", &a, &b);
        if (gcd(a, 26) != 1) {
54 .
            printf("Invalid value of a. It must be coprime with 26.\n");
55
56
            return 1;
57
        3
58
        encryptText(plaintext, ciphertext, a, b);
59
        printf("Encrypted text: %s\n", ciphertext);
60
        decryptText(ciphertext, decrypted, a, b);
61
        printf("Decrypted text: %s\n", decrypted);
62
        return 0;
63 }
```

(P)

E

5

E,

0

0

(

JS

```
Run
                                          [] & & Share
main.c
                                                                             Output
        int dp = (p1 - p2 + 26) % 26;
31
                                                                            Enter ciphertext (uppercase letters only):
32
        int dc = (c1 - c2 + 26) % 26;
        int inv = modInverse(dp, 26);
                                                                            Guessed keys: a = 3, b = 15
33
        if (inv == -1)
34
                                                                            Decrypted text:
35
           return 0;
                                                                            ERZZT
36
        *a = (dc * inv) % 26;
37
        *b = (c1 - (*a * p1) + 26 * 26) % 26;
38
        return 1;
                                                                            === Code Execution Successful ===
39 }
40 - int main() {
41
        char ciphertext[1000], plaintext[1000];
42
        printf("Enter ciphertext (uppercase letters only):\n");
43
44
        fgets(ciphertext, sizeof(ciphertext), stdin);
45
        ciphertext[strcspn(ciphertext, "\n")] = '\0';
        int c1 = 'B' - 'A'; // 1
46
        int c2 = 'U' - 'A'; // 20
47
48
        int p1 = 'E' - 'A'; // 4
49
        int p2 = 'T' - 'A'; // 19
50
        if (!solveAffineKeys(c1, p1, c2, p2, &a, &b)) {
51 -
52
            printf("Failed to solve affine key equations.\n");
53
            return 1;
54
55
        printf("Guessed keys: a = %d, b = %d\n", a, b);
        decryptText(ciphertext, plaintext, a, b);
56
57
        printf("Decrypted text:\n%s\n", plaintext);
58
        return 0.
```

Q

5

0

(6)

0

JS

TS

E

1

B