DAY 1

1. Write a C program for Caesar

cipher involves replacing each letter of the alphabet with the letter standing

k places further down the alphabet, for k in the range 1 through 25.

PROGRAM

#include <stdio.h>

#include <string.h>

// Function to encrypt a message using Caesar cipher

void encrypt(char message[], int key) {

int i;

char ch;

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

ch = message[i];

if(ch >= 'a' && ch <= 'z') {

ch = ch + key;

if(ch > 'z') {

ch = ch - 'z' + 'a' - 1;

}

message[i] = ch;

}

else if(ch >= 'A' && ch <= 'Z') {

ch = ch + key;

if(ch > 'Z') {

ch = ch - 'Z' + 'A' - 1;

}

message[i] = ch;

}

}

}

// Function to decrypt a message using Caesar cipher

void decrypt(char message[], int key) {

encrypt(message, -key);

}

int main() {

char message[100];

int key;

printf("Enter a message: ");

gets(message);

printf("Enter the key (an integer): ");

scanf("%d", &key);

encrypt(message, key);

printf("Encrypted message: %s\n", message);

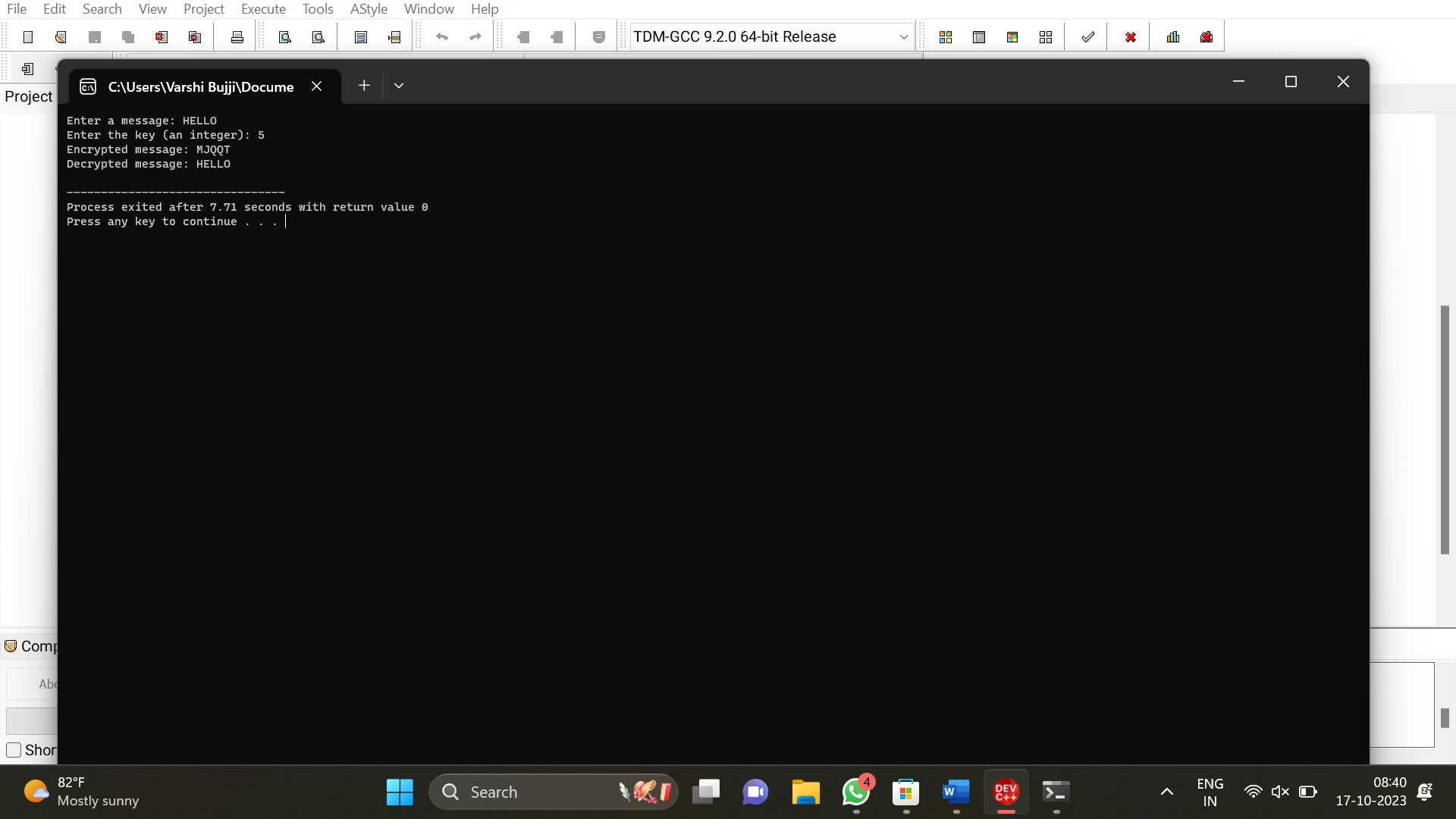
decrypt(message, key);

printf("Decrypted message: %s\n", message);

    return 0;

}

OUTPUT



2. Write a C program for monoalphabetic substitution cipher maps a

plaintext alphabet to a ciphertext alphabet, so that each letter of the

plaintext alphabet maps to a single unique letter of the ciphertext alphabet.

PROGRAM

#include <stdio.h>

#include <string.h>

void caesarCipher(char \*text, int shift) {

for (int i = 0; i < strlen(text); i++) {

char c = text[i];

if (c >= 'a' && c <= 'z') {

text[i] = 'a' + (c - 'a' + shift) % 26;

} else if (c >= 'A' && c <= 'Z') {

text[i] = 'A' + (c - 'A' + shift) % 26;

}

}

}

int main() {

char text[100];

int shift;

printf("Enter text: ");

gets(text);

printf("Enter shift value (1-25): ");

scanf("%d", &shift);

if (shift >= 1 && shift <= 25) {

caesarCipher(text, shift);

printf("Caesar Cipher: %s\n", text);

} else {

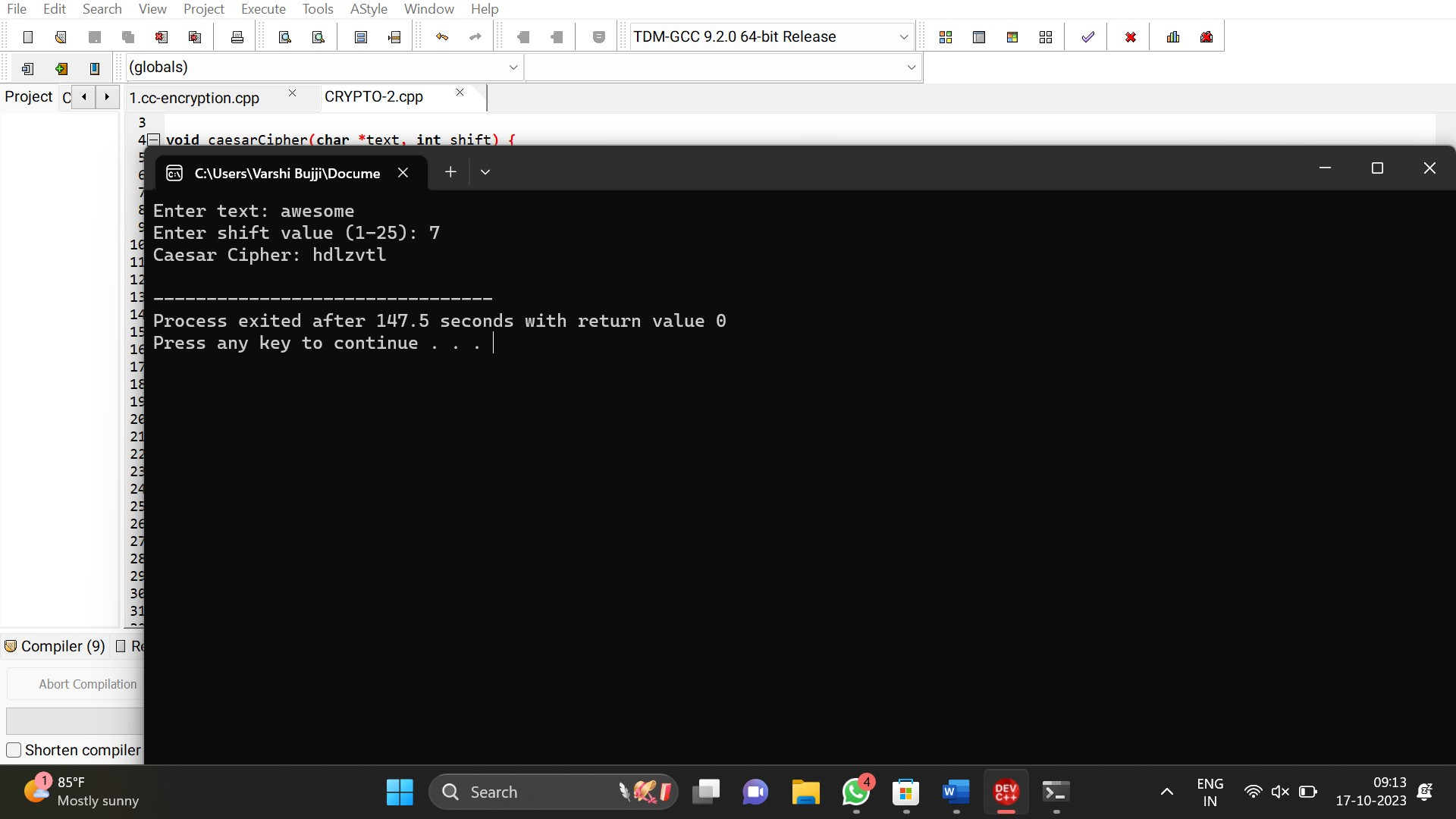
printf("Shift value must be between 1 and 25.\n");

}

    return 0;

}

OUTPUT



3. Write a C program for

Playfair algorithm is based on the use of a 5 X 5 matrix of letters constructed

using a keyword. Plaintext is encrypted two letters at a time using this

matrix.

PROGRAM

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void constructMatrix(char key[], char matrix[5][5]) {

int k, flag = 0;

char table[26] = {0};

int keylen = strlen(key);

for (k = 0; k < keylen; k++) {

if (key[k] == 'J')

key[k] = 'I';

key[k] = toupper(key[k]);

if (table[key[k] - 'A'] == 0) {

table[key[k] - 'A'] = 1;

matrix[flag / 5][flag % 5] = key[k];

flag++;

}

}

for (k = 0; k < 26; k++) {

if (table[k] == 0) {

matrix[flag / 5][flag % 5] = (char)('A' + k);

flag++;

}

}

}

void findPosition(char matrix[5][5], char ch, int \*row, int \*col) {

if (ch == 'J')

ch = 'I';

int i, j;

for (i = 0; i < 5; i++) {

for (j = 0; j < 5; j++) {

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

\*row = i;

\*col = j;

return;

}

}

}

}

void encrypt(char matrix[5][5], char plaintext[], char ciphertext[]) {

int len = strlen(plaintext);

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

char ch1 = plaintext[i];

char ch2 = plaintext[i + 1];

int row1, col1, row2, col2;

findPosition(matrix, ch1, &row1, &col1);

findPosition(matrix, ch2, &row2, &col2);

if (row1 == row2) {

ciphertext[i] = matrix[row1][(col1 + 1) % 5];

ciphertext[i + 1] = matrix[row2][(col2 + 1) % 5];

} else if (col1 == col2) {

ciphertext[i] = matrix[(row1 + 1) % 5][col1];

ciphertext[i + 1] = matrix[(row2 + 1) % 5][col2];

} else {

ciphertext[i] = matrix[row1][col2];

ciphertext[i + 1] = matrix[row2][col1];

}

}

}

int main() {

char key[] = "KEYWORD";

char matrix[5][5];

char plaintext[] = "HELLO";

char ciphertext[100];

constructMatrix(key, matrix);

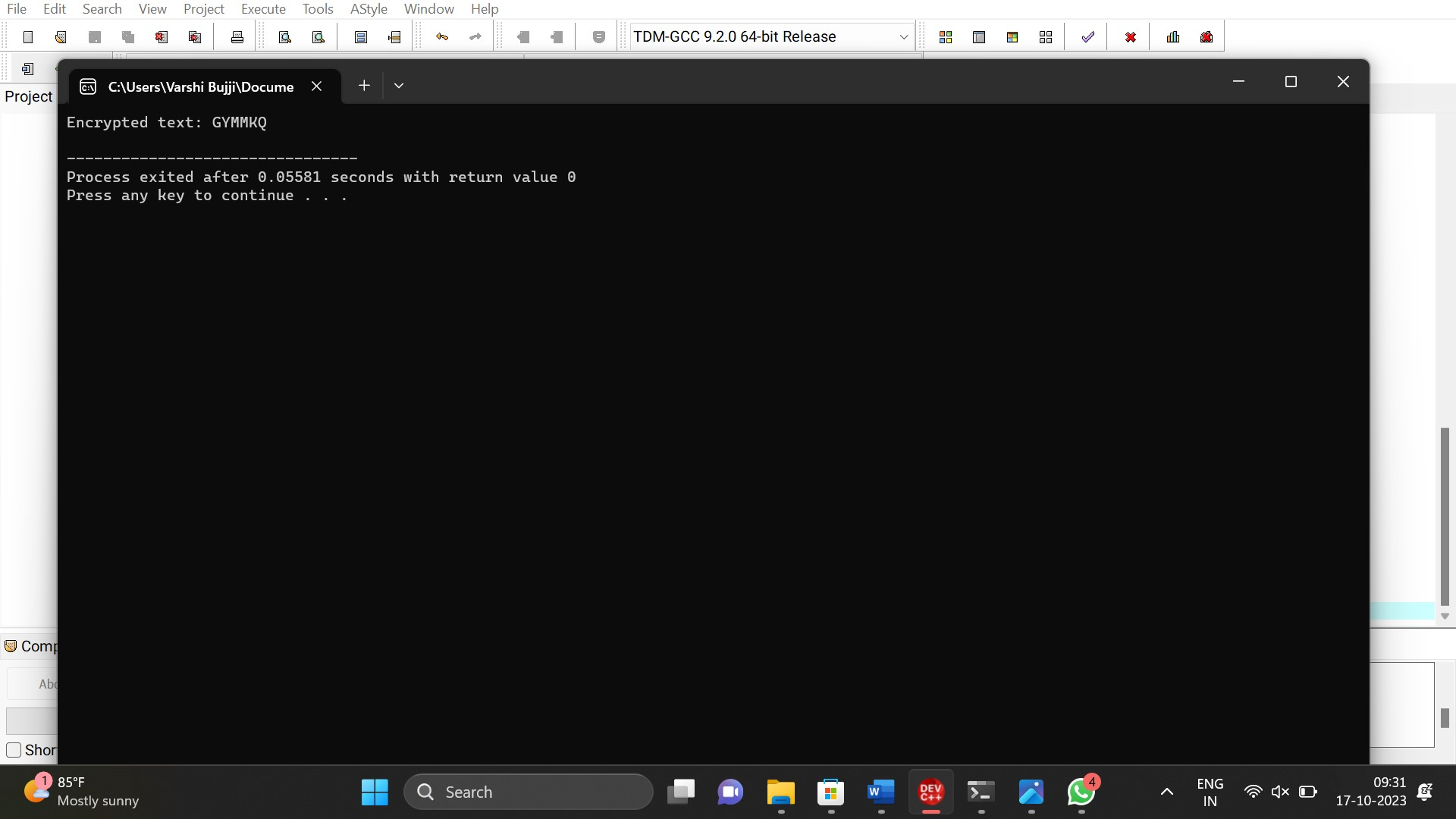
encrypt(matrix, plaintext, ciphertext);

printf("Encrypted text: %s\n", ciphertext);

return 0;

}

OUTPUT



4. Write a C program for

polyalphabetic substitution cipher uses a separate monoalphabetic substitution cipher for each

successive letter of plaintext, depending on

a key.

PROGRAM

#include <stdio.h>

#include <string.h>

// Function to perform polyalphabetic substitution

void polyalphabeticCipher(char plaintext[], char key[]) {

int i, j;

int plaintextLength = strlen(plaintext);

int keyLength = strlen(key);

for (i = 0; i < plaintextLength; i++) {

// Shift the character in the plaintext using the corresponding character from the key

char currentChar = plaintext[i];

char keyChar = key[i % keyLength];

char encryptedChar = 'A' + (currentChar - 'A' + keyChar - 'A') % 26;

// Print the encrypted character

printf("%c", encryptedChar);

}

printf("\n");

}

int main() {

char plaintext[100];

char key[100];

printf("Enter the plaintext: ");

scanf("%s", plaintext);

printf("Enter the key: ");

scanf("%s", key);

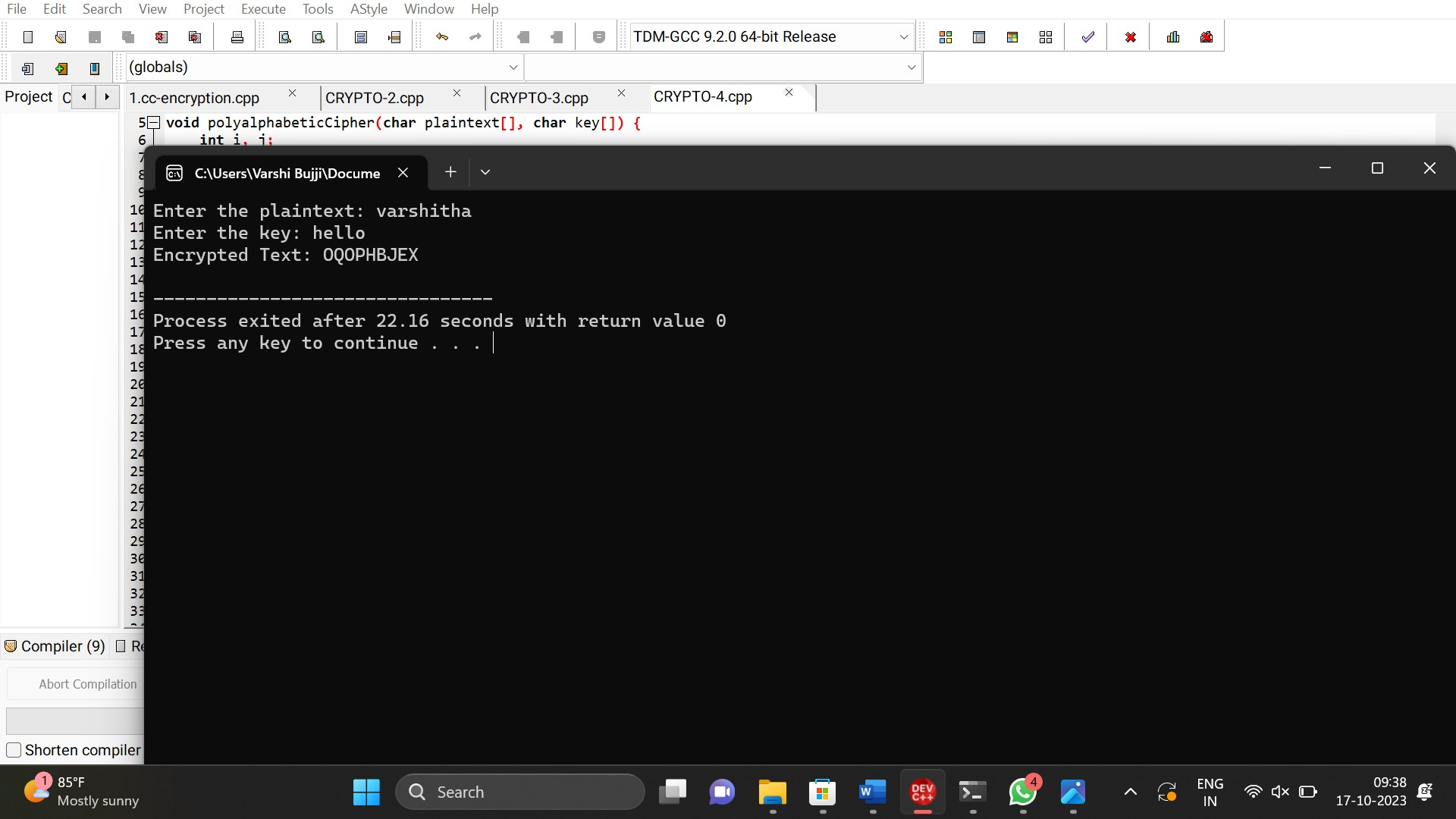
printf("Encrypted Text: ");

polyalphabeticCipher(plaintext, key);

    return 0;

}

OUTPUT



5. Write a C program for

generalization of the Caesar cipher, known as the affine Caesar cipher, has the

following form: For each plaintext letter p, substitute the ciphertext letter

C: C = E([a, b], p) = (ap + b) mod 26 A basic requirement of any encryption algorithm

is that it be one-to-one. That is, if p q,

then E(k, p) E(k, q). Otherwise,

decryption is impossible, because more than

one plaintext character maps into the same ciphertext

character. The affine Caesar cipher is

not one-to-one for all values of a. For example, for a = 2 and b = 3, then E([a,

b], 0) = E([a, b], 13) = 3.

a. Are there any limitations

on the value of b?

b. Determine which values

of a are not allowed.

PROGRAM

#include <stdio.h>

#include <string.h>

// Function to decrypt the affine cipher

char decrypt(char ch, int a, int b) {

if (ch >= 'A' && ch <= 'Z') {

return ((ch - 'A' - b + 26) \* 15) % 26 + 'A';

} else if (ch >= 'a' && ch <= 'z') {

return ((ch - 'a' - b + 26) \* 15) % 26 + 'a';

} else {

return ch;

}

}

int main() {

char ciphertext[] = "your\_cipher\_text\_here"; // Replace with your ciphertext

char mostFrequentLetter = 'b'; // Replace with the most frequent letter

char secondMostFrequentLetter = 'U'; // Replace with the second most frequent letter

int a, b;

// Find the value of 'b' in the affine cipher

b = (mostFrequentLetter - secondMostFrequentLetter + 26) % 26;

// Iterate over possible values of 'a' and decrypt the text

for (a = 1; a < 26; a++) {

printf("a = %d, b = %d: ", a, b);

for (int i = 0; i < strlen(ciphertext); i++) {

char decrypted = decrypt(ciphertext[i], a, b);

printf("%c", decrypted);

}

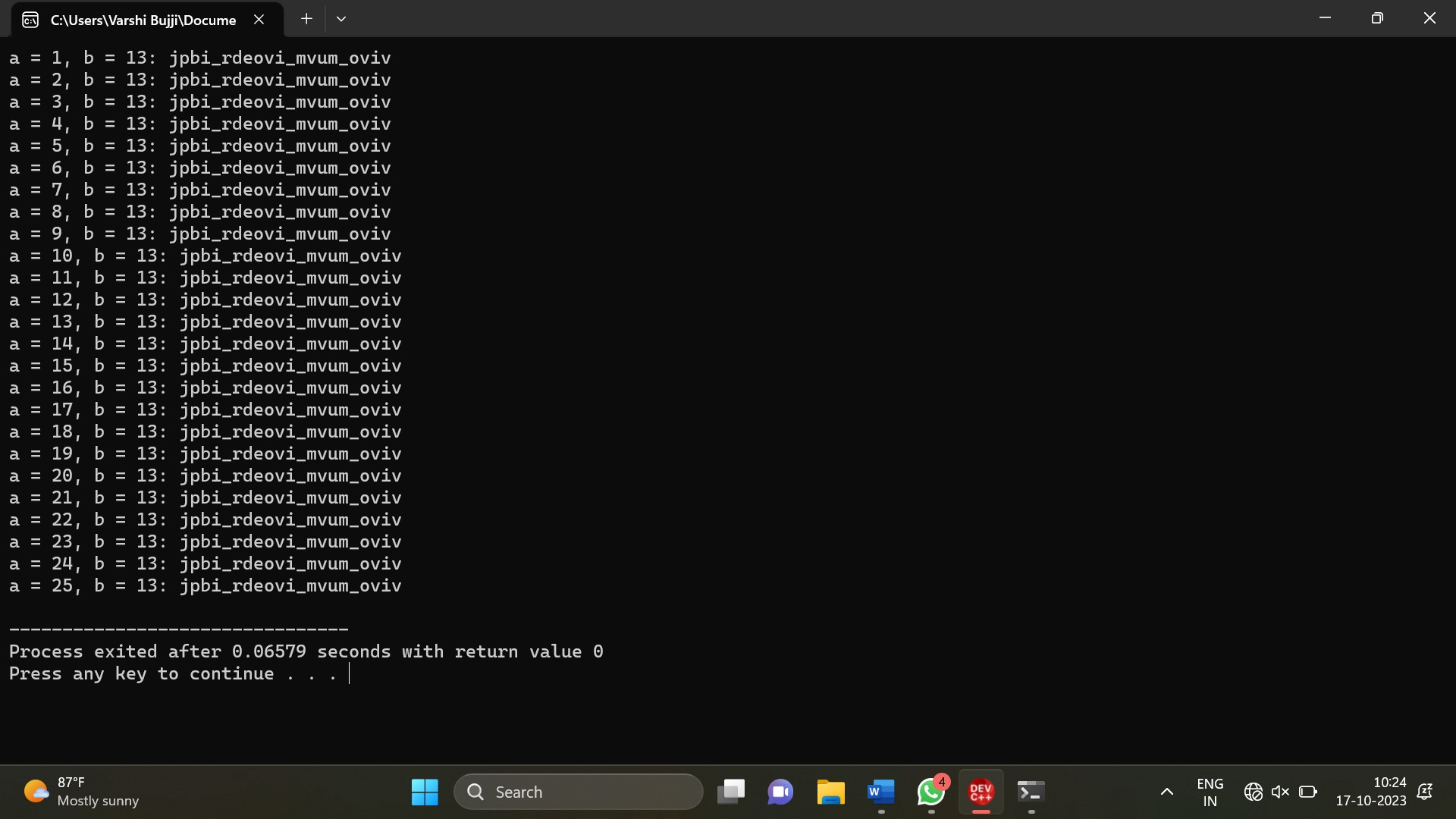
printf("\n");

}

    return 0;

}

OUTPUT



6. Write a C program for

ciphertext has been generated with an affine cipher. The most frequent

letter of the ciphertext is “B,” and the second most frequent letter of the

ciphertext is “U.”Break this code.

PROGRAM

#include <stdio.h>

int gcd(int a, int b) {

if (b == 0)

return a;

return gcd(b, a % b);

}

int is\_allowed\_a(int a) {

return gcd(a, 26) == 1; // 'a' is allowed if it's relatively prime to 26

}

int main() {

int a, b;

printf("Enter the value of 'a': ");

scanf("%d", &a);

if (!is\_allowed\_a(a)) {

printf("Value of 'a' is not allowed. It must be relatively prime to 26.\n");

return 1;

}

printf("Enter the value of 'b': ");

scanf("%d", &b);

if (b < 0 || b >= 26) {

printf("Value of 'b' is not allowed. It must be in the range [0, 25].\n");

return 1;

}

char plaintext;

printf("Enter the plaintext character: ");

scanf(" %c", &plaintext);

if (plaintext >= 'A' && plaintext <= 'Z') {

// Uppercase letter

char ciphertext = 'A' + ((a \* (plaintext - 'A') + b) % 26);

printf("Ciphertext: %c\n", ciphertext);

} else if (plaintext >= 'a' && plaintext <= 'z') {

// Lowercase letter

char ciphertext = 'a' + ((a \* (plaintext - 'a') + b) % 26);

printf("Ciphertext: %c\n", ciphertext);

} else {

printf("Invalid input. Please enter an uppercase or lowercase letter.\n");

return 1;

}

    return 0;

}

OUTPUT

