**DAY-01 LAB PROGRAMMS**

1.Write a high level code for Caesar cipher involves replace each letter of the alphabet with the letter standing k places further down the alphabet for k in range ( 1-25)?

**C program:**

**#include<stdio.h>**

**#include<string.h>**

**#include<conio.h>**

**#include<ctype.h>**

**int main (**

**{**

**Char plain [50], cipher [10];**

**int key ,i ,result;**

**int length;**

**clrscr ();**

**printf (“enter the plain text:”);**

**scanf (“%s”, plain);**

**printf (“enter the key value:”);**

**scanf (“%s”, key);**

**printf (“\n \n \n PLAIN TEXT:%s”, plain);**

**printf (“\n \n \n ENCRYPTED TEXT:”);**

**for (i=0; length=strlen(plain);i<length;i++);**

**{**

**Cipher[i]=plain[i]+key;**

**if(i supper(plain[i])&&(cipher[i]>’Z’))**

**cipher[i]=cipher[i]-26;**

**printf(i slower(plain[i])&&(cipher[i]>’z’))**

**printf(“%c”,cipher[i]);**

**}**

**printf (“\n \n \n DECRYPTED TEXT:”);**

**for (i=0;i<length;i++);**

**{**

**plain[i]=cipher[i]-key;**

**if(i supper(cipher[i])&&(plain[i]>’A’))**

**plain[i]=plain[i]+26;**

**printf(i slower(cipher[i])&&(plain[i]>’a’))**

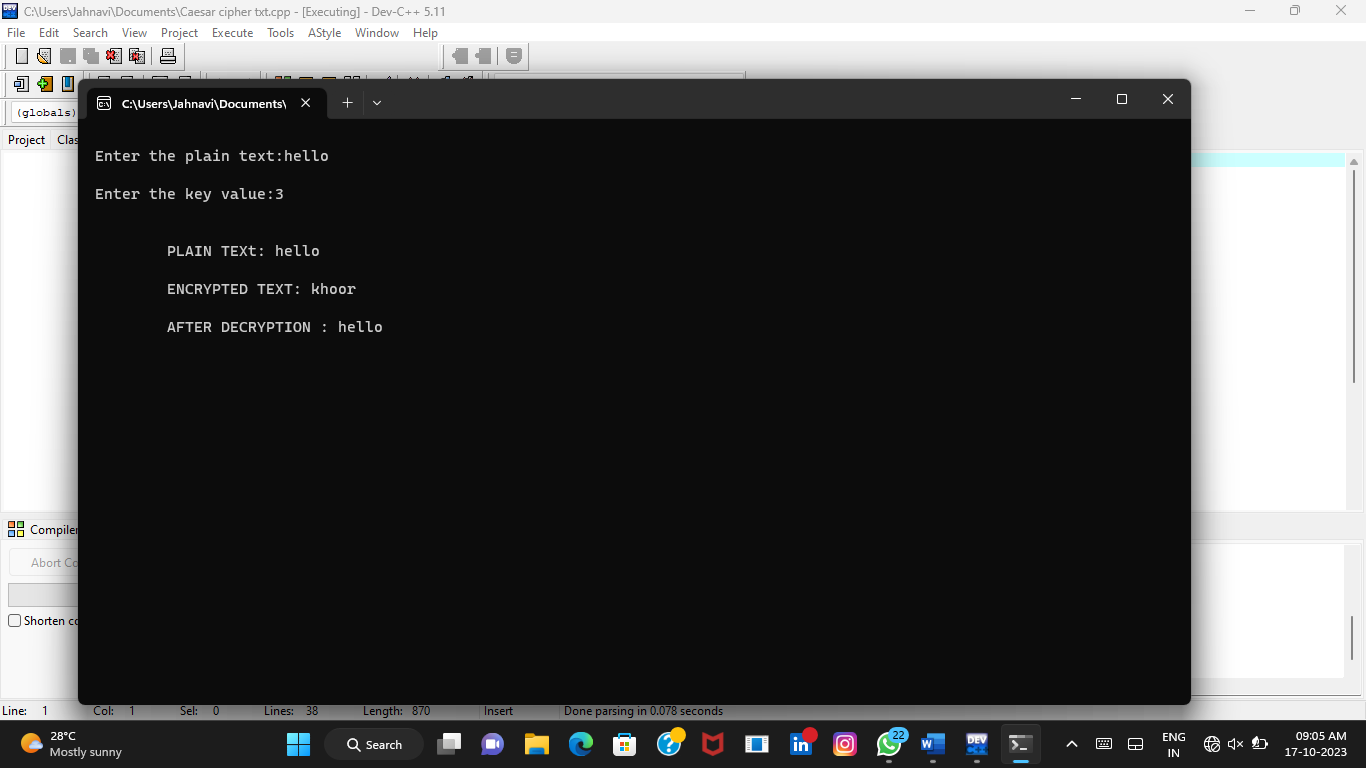
**printf(“%c”,plain[i]);**

**}**

**getch();**

**}**

**OUTPUT:**

****

2.Write a high level code for monoalphabetic substitution cipher maps a a plain text alphabet to a cipher text alphabet , so that each letter of the plain text alphabet maps to a single unique letter of the cipher text alphabet

**C program:**

**#include <stdio.h>**

**#include <ctype.h>**

**char monoalphabetic Encrypt(char ch) {**

**if (i salpha(ch)) {**

**if (i slower(ch)) {**

**return 'a' + (ch - 'a' + 3) % 26;**

**} else if (i supper(ch)) {**

**return 'A' + (ch - 'A' + 3) % 26;**

**}**

**return ch;**

**}**

**int main() {**

**char plaintext[100];**

**printf("Enter the plaintext: ");**

**fgets(plaintext, sizeof(plaintext), stdin);**

**for (int i = 0; plaintext[i] != '\0'; i++) {**

**plaintext[i] = monoalphabetic Encrypt(plaintext[i]);**

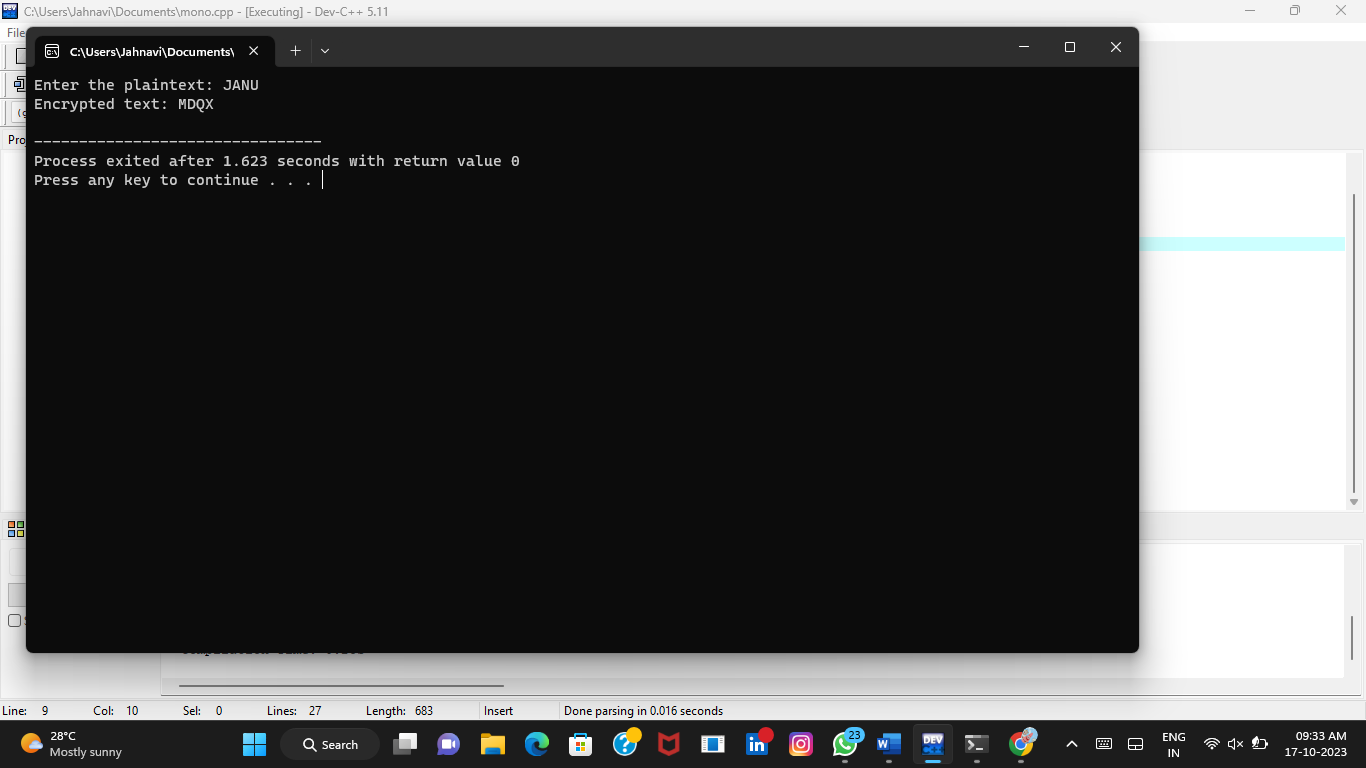
**}**

**printf("Encrypted text: %s", plaintext);**

**return 0;**

**}**

**OUTPUT:**



3.Write a high level program for play fair matrix is on the basis of 5\*5 matrix of letters constructed using a keyword plaintext Is encrypted at a time using this matrix?

**C program:**

**#include <stdio.h>**

**#include <string.h>**

**#include <ctype.h>**

**void createPlayfairMatrix(char matrix[5][5], const char \*key) {**

**int used[26] = {0};**

**char sanitizedKey[100];**

**int keyLength = 0;**

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

**if (isalpha(key[i])) {**

**sanitizedKey[keyLength] = toupper(key[i] == 'J' ? 'I' : key[i]);**

**keyLength++;**

**}**

**}**

**int row = 0, col = 0;**

**for (int i = 0; i < keyLength; i++) {**

**if (!used[sanitizedKey[i] - 'A']) {**

**matrix[row][col] = sanitizedKey[i];**

**used[sanitizedKey[i] - 'A'] = 1;**

**col++;**

**if (col == 5) {**

**col = 0;**

**row++;**

**}**

**}**

**}**

**for (char ch = 'A'; ch <= 'Z'; ch++) {**

**if (ch != 'J' && !used[ch - 'A']) {**

**matrix[row][col] = ch;**

**col++;**

**if (col == 5) {**

**col = 0;**

**row++;**

**}**

**}**

**}**

**}**

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

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

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

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

**\*row = i;**

**\*col = j;**

**return;**

**}**

**}**

**}**

**}**

**void playfairEncrypt(char matrix[5][5], const char \*plaintext, char \*ciphertext) {**

**int len = strlen(plaintext);**

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

**char a = plaintext[i] == 'J' ? 'I' : toupper(plaintext[i]);**

**char b = plaintext[i + 1] == 'J' ? 'I' : toupper(plaintext[i + 1]);**

**int row\_a, col\_a, row\_b, col\_b;**

**findCharLocation(matrix, a, &row\_a, &col\_a);**

**findCharLocation(matrix, b, &row\_b, &col\_b);**

**if (row\_a == row\_b) {**

**ciphertext[i] = matrix[row\_a][(col\_a + 1) % 5];**

**ciphertext[i + 1] = matrix[row\_b][(col\_b + 1) % 5];**

**} else if (col\_a == col\_b) {**

**ciphertext[i] = matrix[(row\_a + 1) % 5][col\_a];**

**ciphertext[i + 1] = matrix[(row\_b + 1) % 5][col\_b];**

**} else {**

**ciphertext[i] = matrix[row\_a][col\_b];**

**ciphertext[i + 1] = matrix[row\_b][col\_a];**

**}**

**}**

**}**

**int main() {**

**char key[100];**

**char plaintext[100];**

**char ciphertext[100];**

**char matrix[5][5];**

**printf("Enter the keyword (no spaces, no 'J's): ");**

**scanf("%s", key);**

**getchar();**

**createPlayfairMatrix(matrix, key);**

**printf("Enter the plaintext: ");**

**fgets(plaintext, sizeof(plaintext), stdin);**

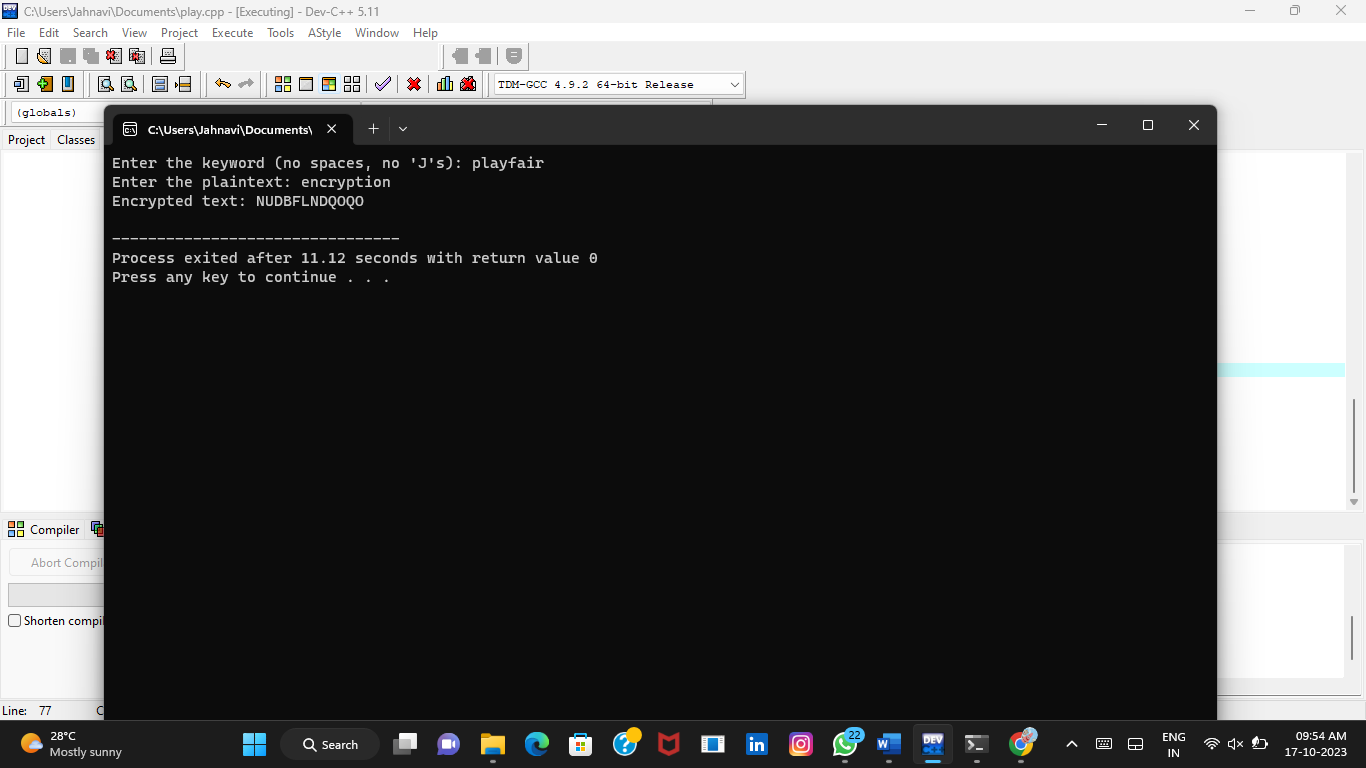
**playfairEncrypt(matrix, plaintext, ciphertext);**

**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?

**C programm:**

#include <stdio.h>

#include<conio.h>

#include <ctype.h>

#include <string.h>

void encipher();

void decipher();

int main()

{

int choice;

while(1)

{

printf("\n1. Encrypt Text");

printf("\t2. Decrypt Text");

printf("\t3. Return -1");

printf("\n\nEnter Your Choice : ");

scanf("%d",&choice);

if(choice == 1)

encipher();

else if(choice == 2)

decipher();

else

return -1;

printf("Please Enter Valid Option.");

}

}

void encipher()

{

unsigned int i,j;

char input[50],key[10];

printf("\n\nEnter Plain Text: ");

scanf("%s",input);

printf("\nEnter Key Value: ");

scanf("%s",key);

printf("\nResultant Cipher Text: ");

for(i=0,j=0;i<strlen(input);i++,j++)

{

if(j>=strlen(key))

{ j=0;

}

printf("%c",65+(((toupper(input[i])-65)+(toupper(key[j])-

65))%26));

}}

void decipher()

{

unsigned int i,j;

char input[50],key[10];

int value;

printf("\n\nEnter Cipher Text: ");

scanf("%s",input);

printf("\n\nEnter the key value: ");

scanf("%s",key);

for(i=0,j=0;i<strlen(input);i++,j++)

{

if(j>=strlen(key))

{ j=0; }

value = (toupper(input[i])-64)-(toupper(key[j])-64);

if( value < 0)

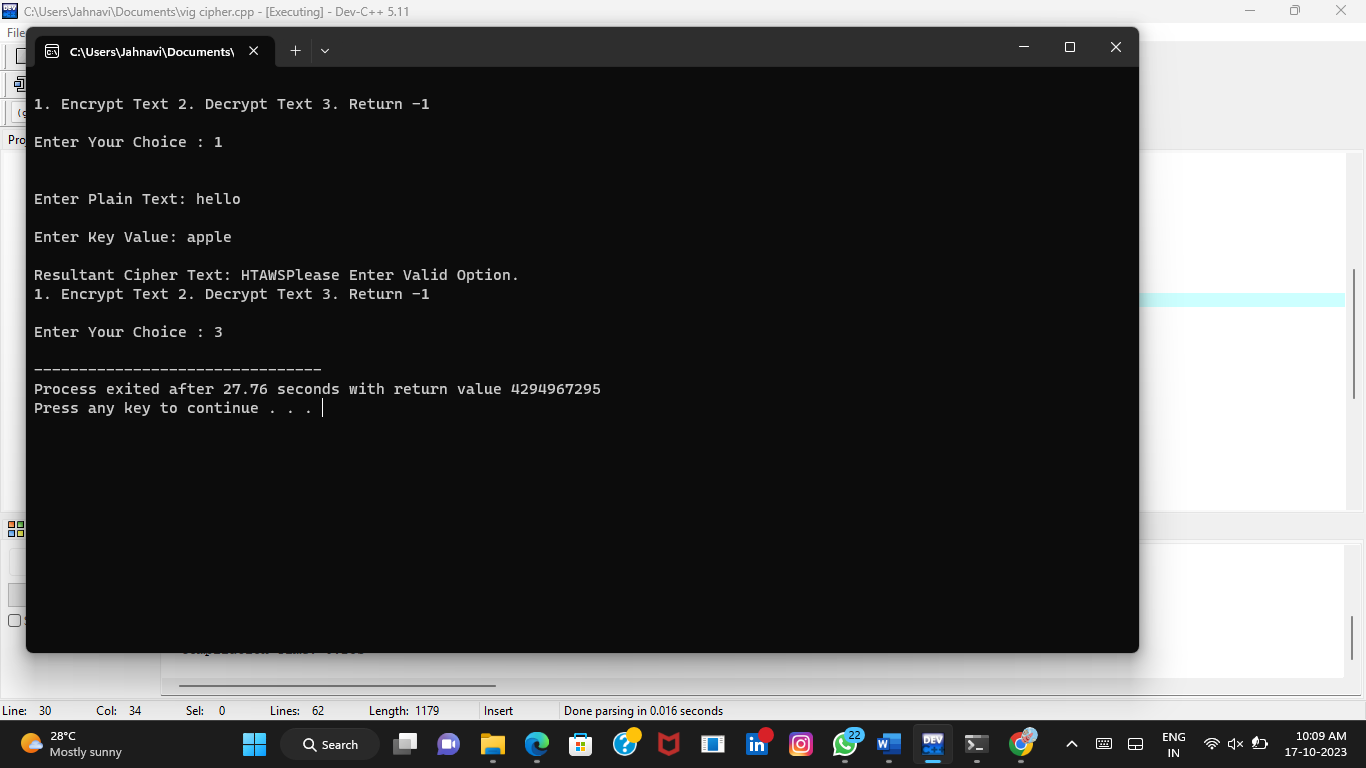
{ value = value \* -1;

}

printf("%c",65 + (value % 26));

}}

**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 Elk, p) Elk, q). Otherwise, decryption is impossible, because more than one plaintext character maps intothe 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 Ella, 6), 0) = Ella, b), 13) = 3.

a. Are there any limitations on the value of b?

b. Determine which values of a are not allowed.

**C program:**

#include <stdio.h>

int gcd(int a, int b) {

if (b == 0)

return a;

return gcd(b, a % b);

}

int main() {

int allowed\_values[26];

int count = 0;

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

if (gcd(a, 26) == 1) {

allowed\_values[count] = a;

count++;

}

}

printf("Allowed values of 'a' are: ");

for (int i = 0; i < count; i++) {

printf("%d ", allowed\_values[i]);

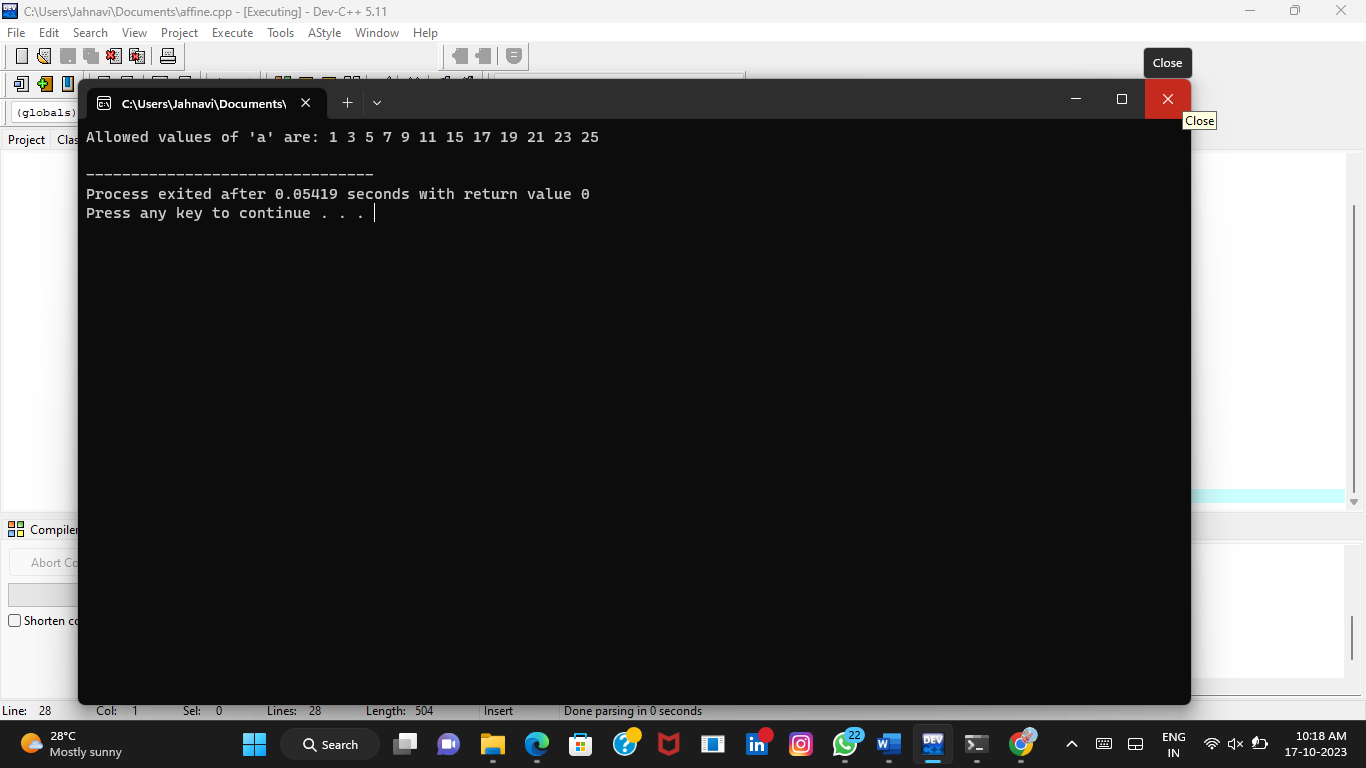
}

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

**C 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:**

