**CSA5109 - CRYPTOGRAPHY AND NETWORK SECURITY**

**LAB PRACTICAL**

**1.To implement Caesar Cipher**

**Ex. No**: 01

**Date:** 02-11-2023

**Program:**

#include<stdio.h>

#include<string.h>

#include<ctype.h>

int main () {

int k=3;

char input [1000], cipher [1000];

printf ("Enter input string:");

scanf ("%s”, &input);

printf ("Encryption is: ");

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

if (is lower(input[i])) {

cipher[i]=((input[i]-'a'+k) %26) +'a';

}

else {

cipher[i]=((input[i]-'A'+k) %26) +'A';

}

printf ("%c”, cipher[i]);

}

printf ("\decryption is: ");

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

if (is lower(input[i])) {

input[i]=((cipher[i]-'a'-k) %26) +'a';

}

else {

input[i]=((cipher[i]-'A'-k) %26) +'A';

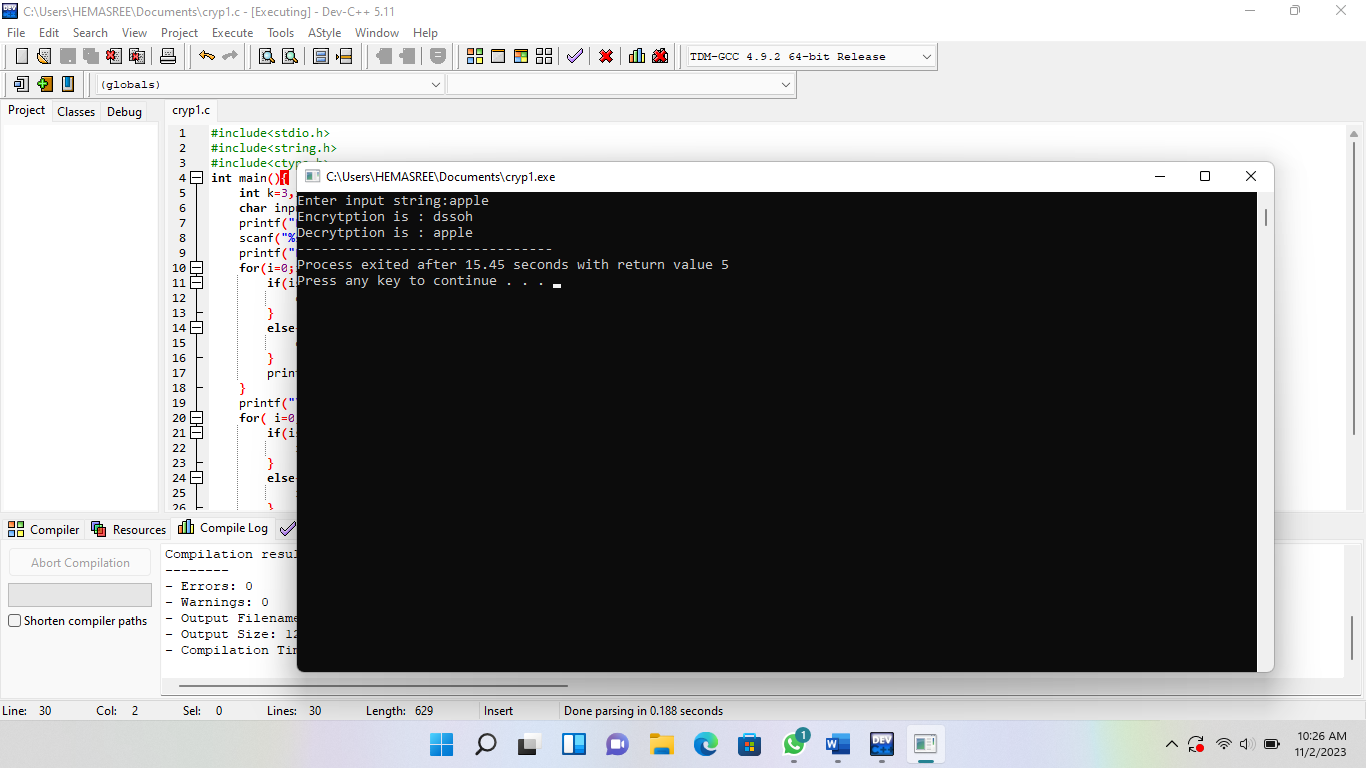
}

printf ("%c”, input[i]);

}

}

**output:**

****

**2.To Implement Mono-alphabetic cipher**

**Ex. No**: 02

**Date**: 02-11-2023

**Program:**

#include<stdio.h>

int main () {

char alpha [100] ="abcdefghijklmnopqrstuvwxyz”, key [100] ="zyxwvutsrqponmlkjihgfedcba”, plain [100], cipher [100];

int m=0, index [100];

printf ("Enter plain text:");

scanf ("%s”, &plain);

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

for (int j=0; j<strlen(alpha); j++) {

if(plain[i]==alpha[j]) {

index[m]=j;

m++;

}

}

}

printf ("Cipher text: ");

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

cipher[i]=key[index[i]];

printf ("%c”, cipher[i]);

}

printf ("\n Plain text: ");

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

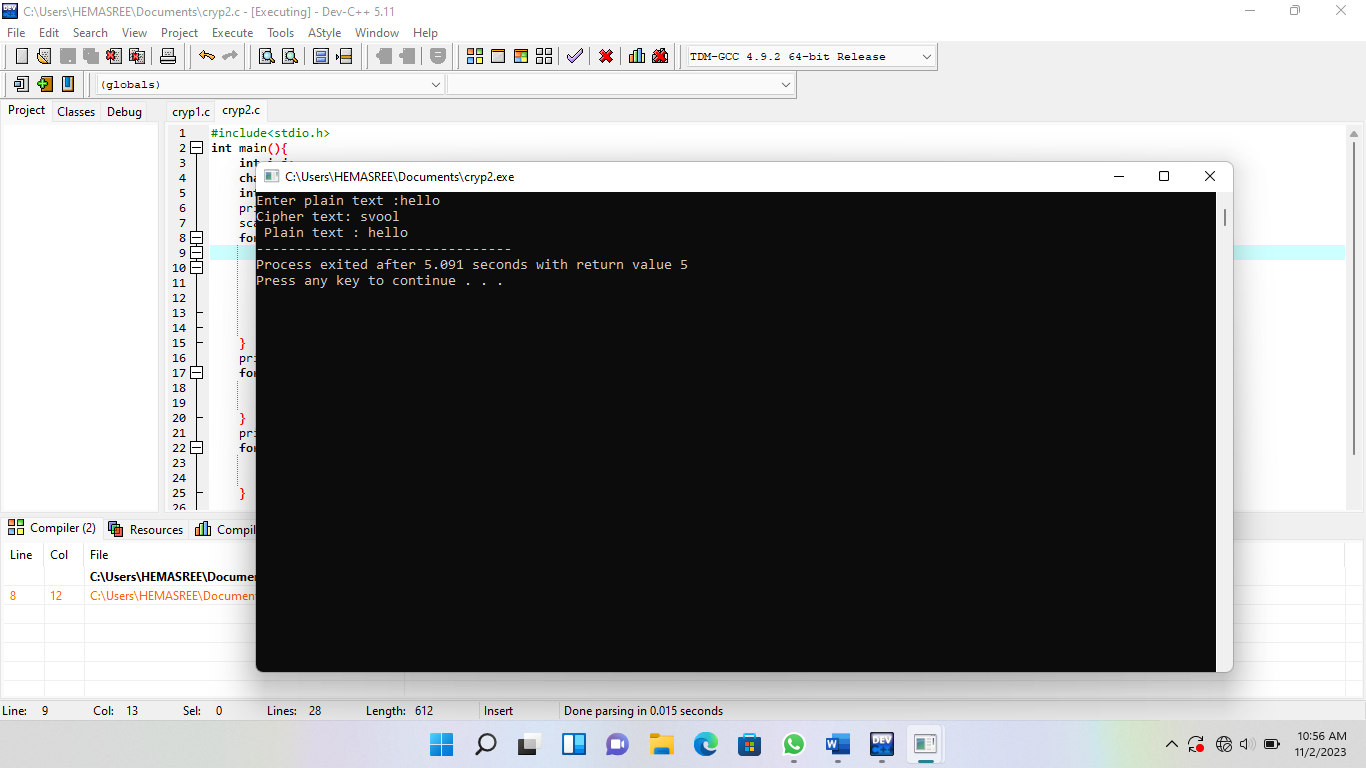
plain[i]=alpha[index[i]];

printf ("%c”, plain[i]);

}

}

**Output:**



**3.To implement Playfair cipher**

**Ex. No:**03

**Date:** 02-11-2023

**Program:**

#include<stdio.h>

#include<string.h>

#include<ctype.h>

int remove repeated (int size, int a []);

int interelement (int position, int a [], int size);

main ()

{

int i, j, k, numstr [100], numcipher [100], numkey [100], lenkey, templen, tempkey [100], flag=-1, size, cipherkey [5][5], lennumstr, row1, row2, col1, col2;

char str [100], key[100];

printf ("Enter a string\n");

gets(str);

//converting entered string to Capital letters

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

{

if(str[i]! =' ')

{

str[j]=toupper(str[i]);

j++;

}

}

str[j]='\0';

printf ("Entered String is %s\n”, str);

//Storing string in terms of ascii and to restore spaces I used -20

size=strlen(str);

for (i=0; i<size; i++)

{

if(str[i]! =' ')

numstr[i]=str[i]-'A';

}

lennumstr=i;

//Key processing

printf ("Enter the key (Non repeated elements if possible) \n");

gets(key);

//converting entered key to Capital letters

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

{

if(key[i]! =' ')

{

key[j]=toupper(key[i]);

j++;

}

}

key[j]='\0';

printf ("%s\n”, key);

//Storing key in terms of ascii

k=0;

for (i=0; i<strlen(key)+26; i++)

{

if(i<strlen(key))

{

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

{

flag=8;

printf ("%d”, flag);

}

numkey[i]=key[i]-'A';

}

else

{

if (k! =9 && k! =flag)//Considering I=J and taking I in place of J except when J is there in key ignoring I

{

numkey[i]=k;

}

k++;

}

}

templen=i;

lenkey=remove repeated (templen, numkey);

printf ("Entered key converted according to Play Fair Cipher rule\n");

for (i=0; i<lenkey; i++)

{

printf ("%c”, numkey[i]+'A');

}

printf("\n");

//Arranging the key in 5x5 grid

k=0;

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

{

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

{

cipherkey[i][j] =numkey[k];

k++;

}

}

printf ("Arranged key\n");

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

{

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

{

printf ("%c “, cipherkey[i][j] +'A');

}

printf("\n");

}

//Message Processing

for (i=0; i<lennumstr; i+=2)

{

if(numstr[i]==numstr[i+1])

{

insertelementat(i+1,numstr,lennumstr);

lennumstr++;

}

}

if(lennumstr%2!=0)

{

insertelementat(lennumstr,numstr,lennumstr);

lennumstr++;

}

printf("Entered String/Message After Processing according to Play fair cipher rule\n");

for(i=0;i<lennumstr;i++)

{

printf("%c",numstr[i]+'A');

}

for(k=0;k<lennumstr;k+=2)

{

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

{

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

{

if(numstr[k]==cipherkey[i][j])

{

row1=i;

col1=j;

}

if(numstr[k+1]==cipherkey[i][j])

{

row2=i;

col2=j;

}

}

}

//Only change between Ecryption to decryption is changing + to -

//If negative add 5 to that row or column

if(row1==row2)

{

col1=(col1-1)%5;

col2=(col2-1)%5;

if(col1<0)

{

col1=5+col1;

}

if(col2<0)

{

col2=5+col2;

}

numcipher[k]=cipherkey[row1][col1];

numcipher[k+1]=cipherkey[row2][col2];

}

if(col1==col2)

{

row1=(row1-1)%5;

row2=(row2-1)%5;

if(row1<0)

{

row1=5+row1;

}

if(row2<0)

{

row2=5+row2;

}

numcipher[k]=cipherkey[row1][col1];

numcipher[k+1]=cipherkey[row2][col2];

}

if(row1!=row2&&col1!=col2)

{

numcipher[k]=cipherkey[row1][col2];

numcipher[k+1]=cipherkey[row2][col1];

}

}

printf("\nCipher Text is\n");

for(i=0;i<lennumstr;i++)

{

if((numcipher[i]+'A')!='X')//Should remove extra 'X' which were created during Encryption

printf("%c",numcipher[i]+'A');

}

printf("\n");

}

int removerepeated(int size,int a[])

{

int i,j,k;

for(i=0;i<size;i++)

{

for(j=i+1;j<size;)

{

if(a[i]==a[j])

{

for(k=j;k<size;k++)

{

a[k]=a[k+1];

}

size--;

}

else

{

j++;

}

}

}

return(size);

}

int insertelementat(int position,int a[],int size)

{

int i,insitem=23,temp[size+1];

for(i=0;i<=size;i++)

{

if(i<position)

{

temp[i]=a[i];

}

if(i>position)

{

temp[i]=a[i-1];

}

if(i==position)

{

temp[i]=insitem;

}

}

for(i=0;i<=size;i++)

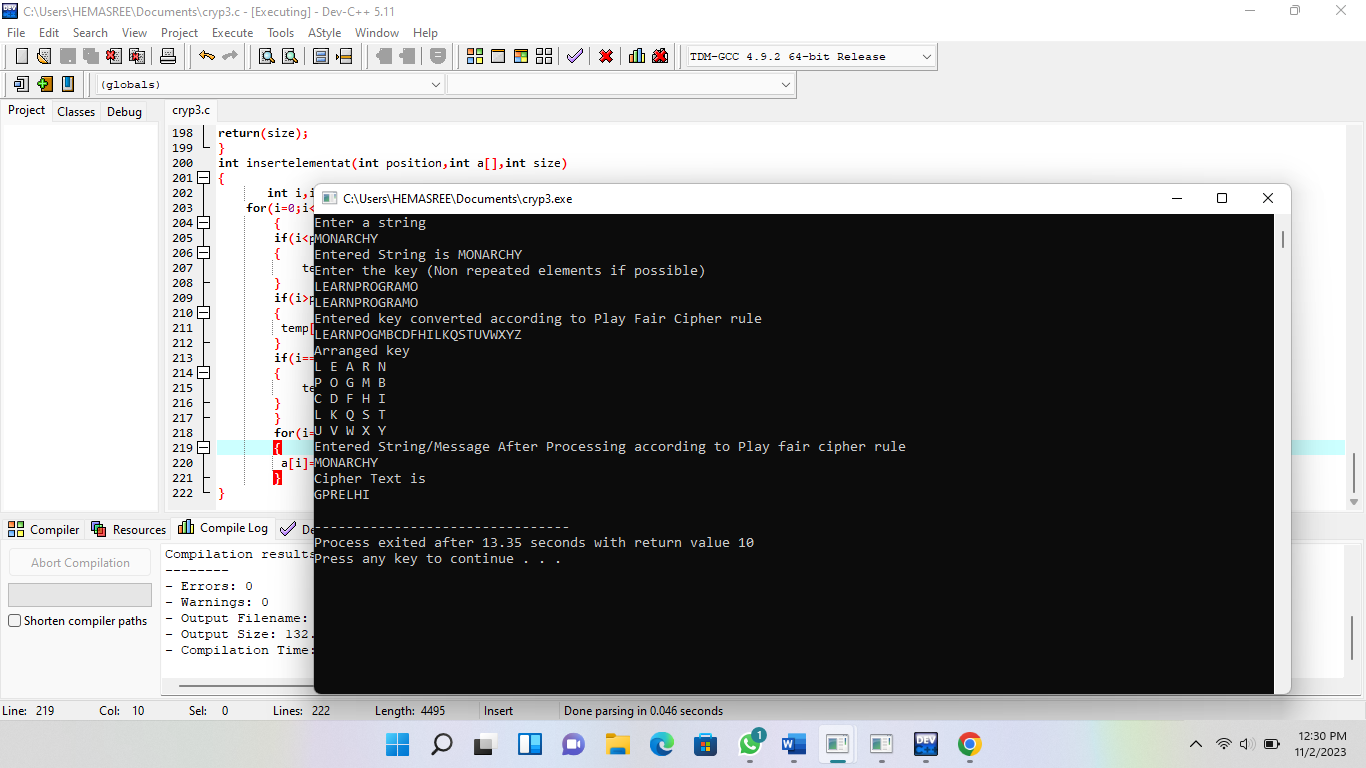
{

a[i]=temp[i];

}

}

**Output:**



**4.To implement poly-alphabetic cipher**

**Ex. No:** 04

**Date:** 02-08-2023

**Program:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

char pt[40]={'\0'},key[40]={'\0'},ct[40]={'\0'},pta[40]={'\0'},k[40]={'\0'};

int i,j; // global values

int main()

{

printf("\nEnter the keyword:\n");

gets(k);//read the key

printf("\nEnter the Plain text:\n");

gets(pt); //read the plain text

// print the table

printf("The convergen matrix\n");

printf("\n ");

for(j=97;j<=122;j++)

{

printf(" %c",j);

}

printf("\n--------------------------------------------------------\n");

for(i=97;i<=122;i++)

{

printf("%c |",i);

for(j=97;j<=122;j++)

{

if(((i+j))>219)

{

printf(" %c",toupper((i+j)-123));

}

else

{

printf(" %c",toupper((i+j)-97));

}

}

printf("\n");

}

// for keyword

j=0;

for(i=0;i<strlen(pt);i++)

{

key[i]=k[j];

if(j==(strlen(k)-1))

{

j=0;

}

else

{

j++;

}

}

for(i=0;i<(strlen(pt)-1);i++);

k[i]='\0';

printf("\nThe encrypted text is:\n");

// encryption

for(i=0;i<strlen(pt);i++)

{

if(97<=(int)pt[i] && (int)pt[i]<=122)

{

if(((int)pt[i]+(int)key[i])>219)

{

ct[i]=(int)pt[i]+(int)key[i]-123;

}

else

{

ct[i]=(int)pt[i]+(int)key[i]-97;

}

printf("%c",toupper(ct[i]));

}

else

{

ct[i]=pt[i];

printf("%c",pt[i]);

}

}

// decryption

printf("\n\nDecrypted text is:\n");

for(i=0;i<strlen(ct);i++)

{

if(97<=(int)ct[i] && (int)ct[i]<=122)

{

if(((int)ct[i]-(int)key[i])<0)

{

pta[i]=((int)ct[i]-(int)key[i])+123;

}

else

{

pta[i]=(int)ct[i]-(int)key[i]+97;

}

printf("%c",pta[i]);

}

else

{

pta[i]=ct[i];

printf("%c",pta[i]);

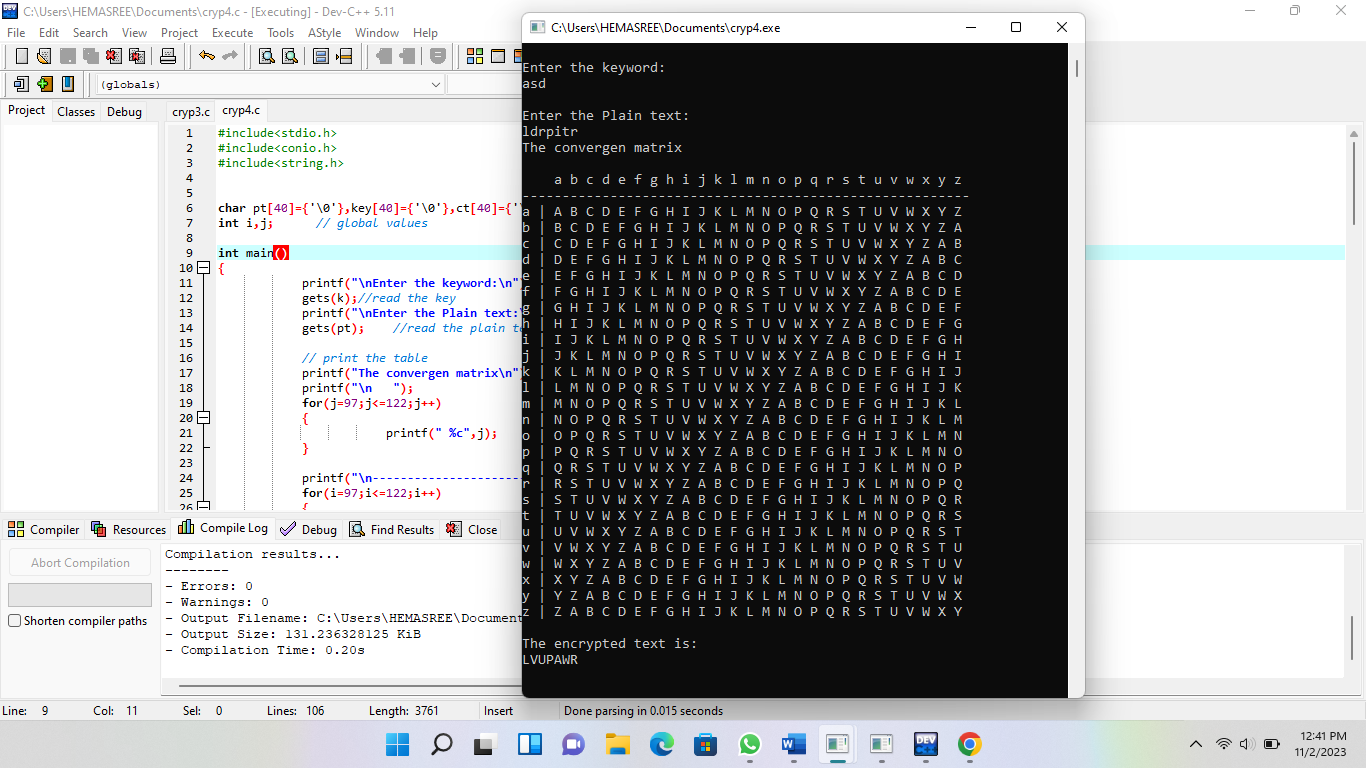
}

}

return 0;

}

**Output:**



**5. To implement for generalization of the affine Caesar**

**Ex. No:**5

**Date:** 02-11-2023

**Program:**

#include <stdio.h>

int gcd(int a, int b) {

if (b == 0) {

return a;

}

return gcd(b, a % b);

}

int main() {

int a;

printf("Allowed values of 'a' for the affine Caesar cipher: ");

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

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

printf("%d ", a);

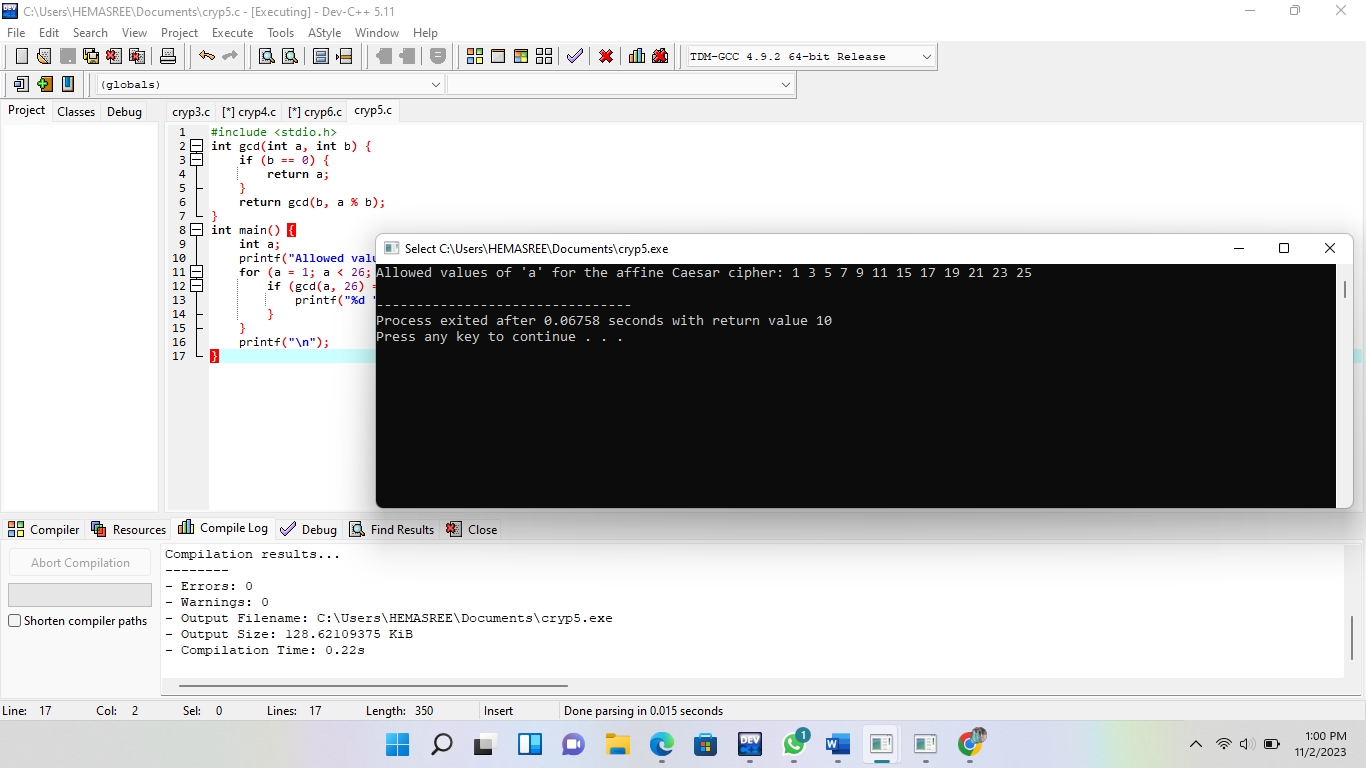
}

}

printf("\n");

}

**Output:**



**6.To implement poly-alphabetic cipher**

**Ex. No:** 06

**Date:** 02-08-3-2023

**Program:**

#include <stdio.h>

int main() {

char ciphertext[] = "BUWXYBZVBQBUCB";

char mostFrequent = 'B';

char secondMostFrequent = 'U';

int mostFrequentNum = mostFrequent - 'A';

int secondMostFrequentNum = secondMostFrequent - 'A';

int a, b, i;

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

int candidateB = (secondMostFrequentNum - (a \* mostFrequentNum) + 26) % 26;

int valid = 1,i;

for (i = 0; i < sizeof(ciphertext) - 1; i++) {

int ciphertextNum = ciphertext[i] - 'A';

int decryptedNum = (a \* ciphertextNum + candidateB) % 26;

if (decryptedNum < 0) {

valid = 0;

break;

}

}

if (valid) {

b = candidateB;

break;

}

}

printf("Decrypted message: ");

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

int ciphertextNum = ciphertext[i] - 'A';

int decryptedNum = (a \* ciphertextNum + b) % 26;

char decryptedChar = decryptedNum + 'A';

printf("%c", decryptedChar);

}

printf("\n");

return 0;

}

**Output:**

