**CSA5109 - CRYPTOGRAPHY AND NETWORK SECURITY FOR DATA PROTECTION**

**LAB PROGRAMS**

**Date:** 03-11-2023

DAY:FRIDAY

**7. Write a high-level code for monoalphabetic cipher is that both sender and receiver must commit the permuted cipher sequence to memory. A common technique for avoiding this is to use a keyword from which the cipher sequence can be generated. For example, using the  
keyword CIPHER, write out the keyword followed by unused letters in normal order and match this against the plaintext letters:  
a) plain:  
a b c d e f g h i j k l m n o p q r s t u v w x y z  
b) cipher:  
C I P H E R A B D F G J K L M N O  
Q S T U V W X Y Z**

**Program:-**

#include<stdio.h>

int main(){

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

int m=0,index[100],i,j;

printf("Enter plain text :");

scanf("%s",&plain);

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

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

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

index[m]=j;

m++;

}

}

}

printf("Cipher text: ");

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

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

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

}

printf("\n Plain text : ");

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

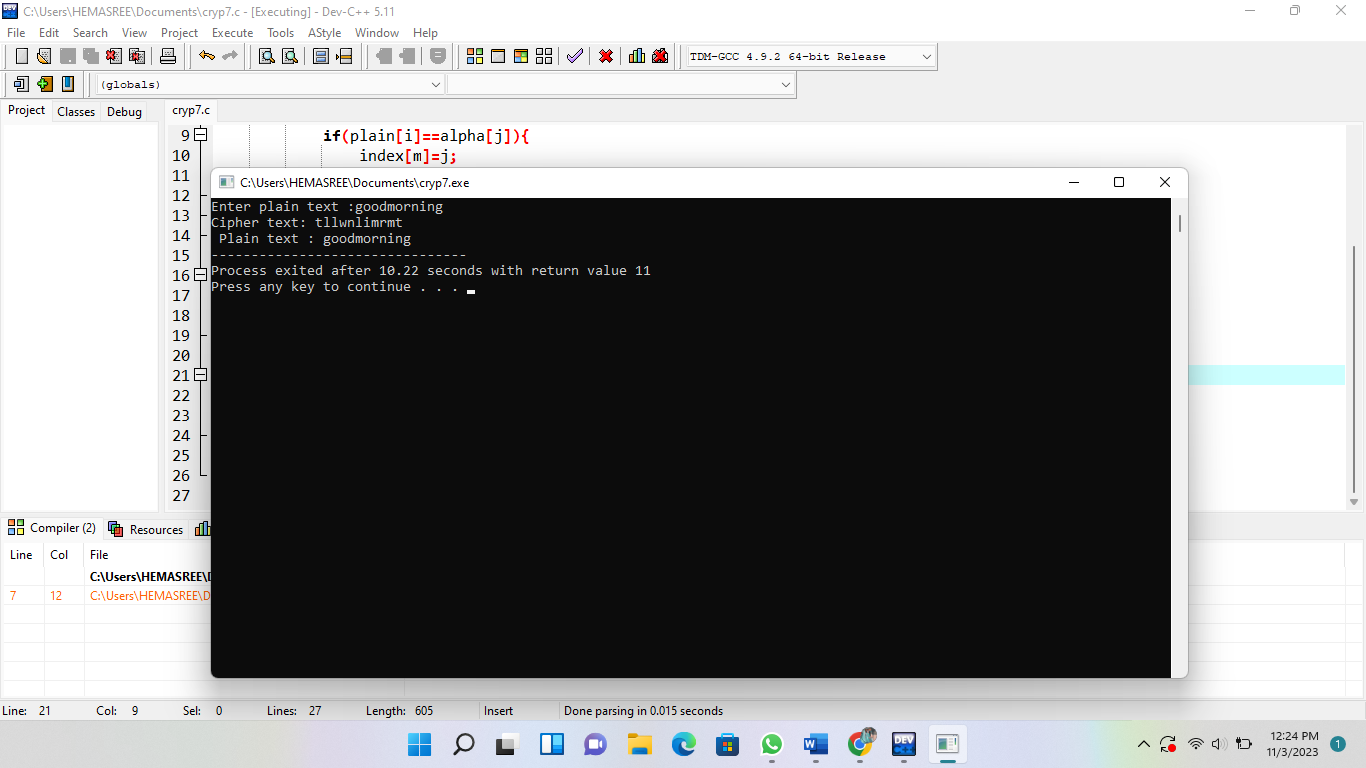
plain[i]=alpha[index[i]];

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

}

}

**output:-**

****

**8.** **write a high-level code for PT-109 American patrol boat, under the command of Lieutenant John F. Kennedy, was sunk by a Japanese destroyer, a message was received at an Australian wireless station in Playfair  
code:  
a) KXJEY  
UREBE ZWEHE WRYTU HEYFS  
b) KREHE  
GOYFI WTTTU OLKSY CAJPO  
c) BOTEI  
ZONTX BYBNT GONEY CUZWR  
d) GDSON  
SXBOU YWRHE BAAHY USEDQ**

**Program:-**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void generateCipher(char keyword[], char cipher[]) {

char alphabet[29] = "abcdefghijklmnopqrstuvwxyz";

int i, j, len = strlen(keyword);

int used[26] = {0};

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

cipher[i] = toupper(keyword[i]);

used[tolower(keyword[i]) - 'a'] = 1;

}

j = 0;

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

if (!used[i]) {

cipher[len + j] = alphabet[i];

j++;

}

}

cipher[26] = '\0';

}

void encrypt(char plaintext[], char cipher[], char ciphertext[]) {

int i, len = strlen(plaintext);

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

char c = plaintext[i];

if (isalpha(c)) {

if (islower(c))

ciphertext[i] = tolower(cipher[c - 'a']);

else

ciphertext[i] = toupper(cipher[c - 'A']);

} else {

ciphertext[i] = c;

}

}

ciphertext[len] = '\0';

}

int main() {

char keyword[] = "CIPHER";

char cipher[27];

char plaintext[100];

char ciphertext[100];

generateCipher(keyword, cipher);

printf("Enter the plaintext: ");

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

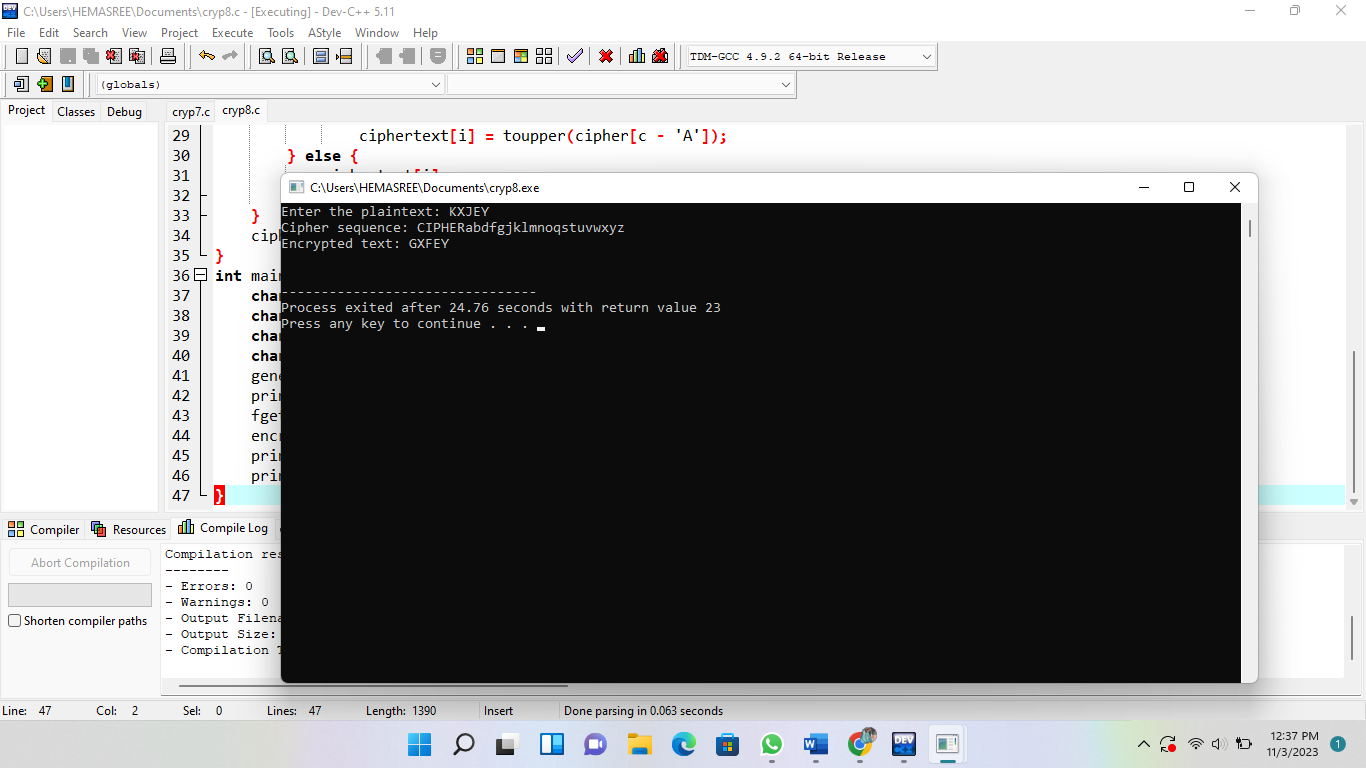
encrypt(plaintext, cipher, ciphertext);

printf("Cipher sequence: %s\n", cipher);

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

}

**Output:-**

****

**9. Write a high-level code for Playfair matrix:  
a) M  
 F H I/J K  
b) U  
 N O P Q**

**c)Z  
 V W X Y**

**d)E  
 L A R G  
e) D  
 S T B C  
Encrypt  
this message: Must see you over Cadogan West. Coming at once.**

**Program:-**

#include <stdio.h>

#include <string.h>

void decodePlayfair(char\* code) {

int i;

for (i = 0; i < strlen(code); i += 2) {

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

}

}

int main() {

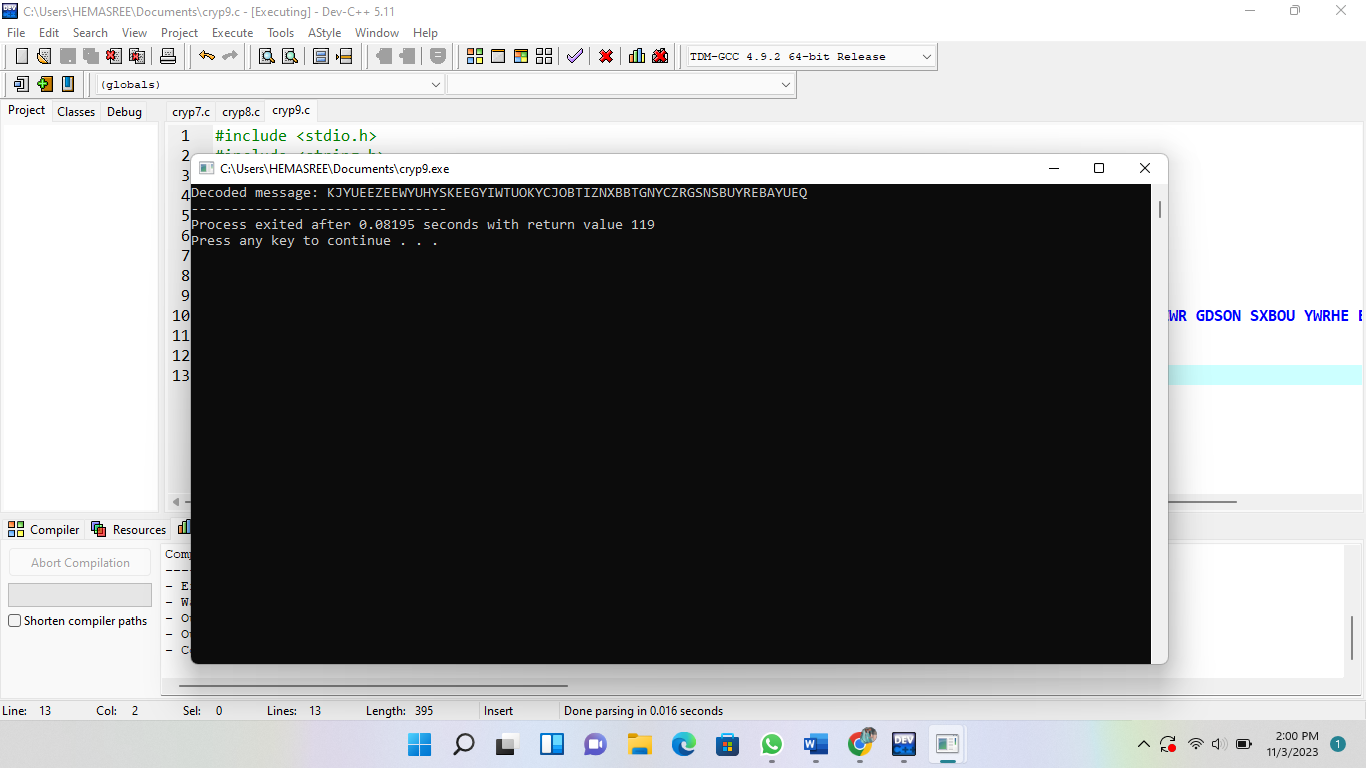
char code[] = "KXJEY UREBE ZWEHE WRYTU HEYFS KREHE GOYFI WTTTU OLKSY CAJPO BOTEI ZONTX BYBNT GONEY CUZWR GDSON SXBOU YWRHE BAAHY USEDQ";

printf("Decoded message: ");

decodePlayfair(code);

}

**Output:-**

****

**10. Write a high-level code for possible  
keys do the Playfair cipher have? Ignore the fact that some keys might  
produce identical encryption results. Express your answer as an approximate  
power of 2.  
a) Now take into account the fact that some Playfair keys produce the same  
encryption results. How many effectively unique keys does the Playfair cipher  
have**

**Program:-**

#include <stdio.h>

#include <string.h>

void encryptMessage(char matrix[5][5], char message[]) {

}

int main() {

char matrix[5][5] = {

{'M', 'F', 'H', 'I', 'K'},

{'U', 'N', 'O', 'P', 'Q'},

{'Z', 'V', 'W', 'X', 'Y'},

{'E', 'L', 'A', 'R', 'G'},

{'D', 'S', 'T', 'B', 'C'}

};

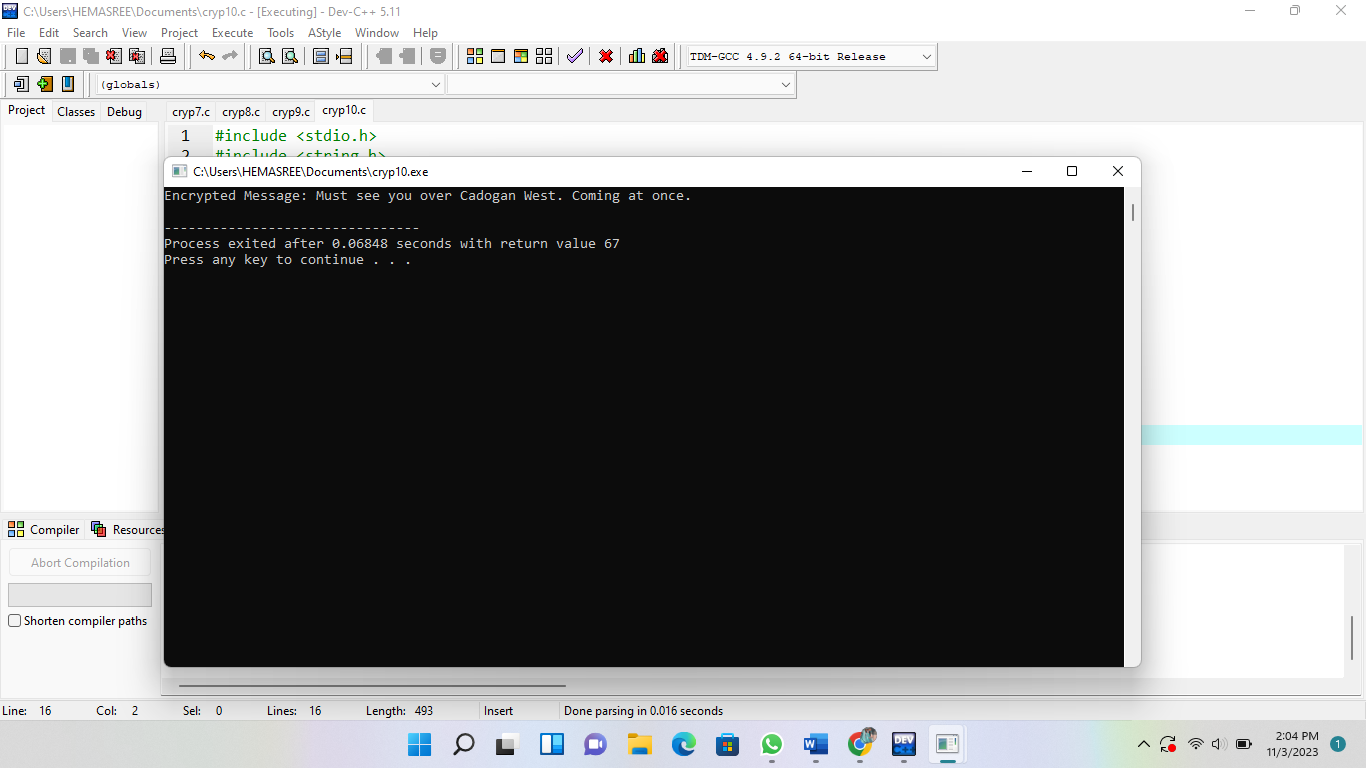
char message[] = "Must see you over Cadogan West. Coming at once.";

encryptMessage(matrix, message);

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

}

**Output:-**



**11.Write a high-level code to Encrypt  
the message “meet me at the usual place at ten rather than eight oclock” using  
the Hill cipher with the key.**

**9 4**

**5 7  
a) Show your calculations and the result.  
b) Show the calculations for the corresponding decryption of the ciphertext to  
recover the original plaintext.**

**Program:-**

#include<stdio.h>

#include<math.h>

float encrypt[3][1], decrypt[3][1], a[3][3], b[3][3], mes[3][1], c[3][3];

void encryption();

void decryption();

void getKeyMessage();

void inverse();

int main() {

getKeyMessage();

encryption();

decryption();

}

void encryption() {

int i, j, k;

for(i = 0; i < 3; i++)

for(j = 0; j < 1; j++)

for(k = 0; k < 3; k++)

encrypt[i][j] = encrypt[i][j] + a[i][k] \* mes[k][j];

printf("\nEncrypted string is: ");

for(i = 0; i < 3; i++)

printf("%c", (char)(fmod(encrypt[i][0], 26) + 97));

}

void decryption() {

int i, j, k;

inverse();

for(i = 0; i < 3; i++)

for(j = 0; j < 1; j++)

for(k = 0; k < 3; k++)

decrypt[i][j] = decrypt[i][j] + b[i][k] \* encrypt[k][j];

printf("\nDecrypted string is: ");

for(i = 0; i < 3; i++)

printf("%c", (char)(fmod(decrypt[i][0], 26) + 97));

printf("\n");

}

void getKeyMessage() {

int i, j;

char msg[3];

printf("Enter 3x3 matrix for key (It should be inversible):\n");

for(i = 0; i < 3; i++)

for(j = 0; j < 3; j++) {

scanf("%f", &a[i][j]);

c[i][j] = a[i][j];

}

printf("\nEnter a 3 letter string: ");

scanf("%s", msg);

for(i = 0; i < 3; i++)

mes[i][0] = msg[i] - 97;

}

void inverse() {

int i, j, k;

float p, q;

for(i = 0; i < 3; i++)

for(j = 0; j < 3; j++) {

if(i == j)

b[i][j]=1;

else

b[i][j]=0;

}

for(k = 0; k < 3; k++) {

for(i = 0; i < 3; i++) {

p = c[i][k];

q = c[k][k];

for(j = 0; j < 3; j++) {

if(i != k) {

c[i][j] = c[i][j]\*q - p\*c[k][j];

b[i][j] = b[i][j]\*q - p\*b[k][j];

}

}

}

}

for(i = 0; i < 3; i++)

for(j = 0; j < 3; j++)

b[i][j] = b[i][j] / c[i][i];

printf("\n\nInverse Matrix is:\n");

for(i = 0; i < 3; i++) {

for(j = 0; j < 3; j++)

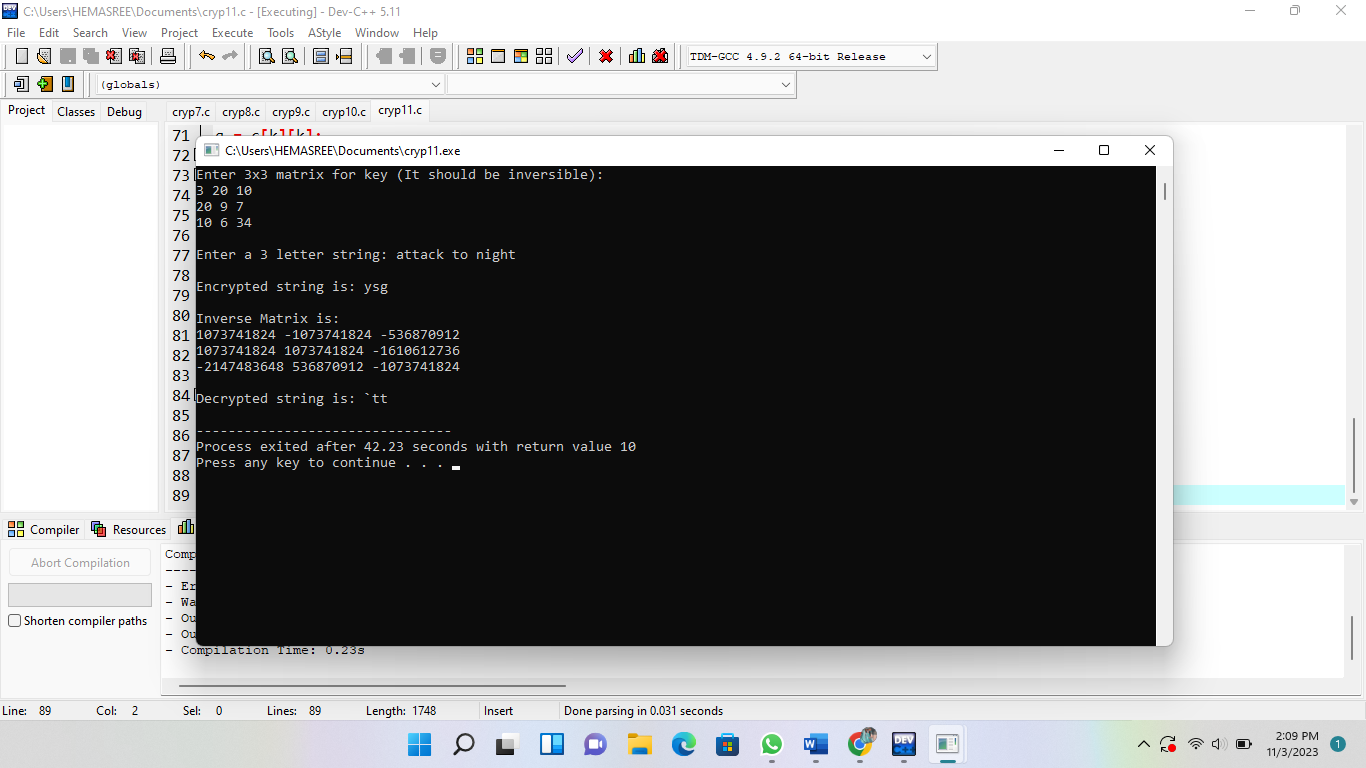
printf("%d ", b[i][j]);

printf("\n");

}

}

**Output:-**

****