N.KARTHIK REDDY-192111167

DAY - 2 LAB EXPERIMENTS

QUESTION 1 :

Write a High level codefor 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 fromwhich 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:

plain : abcdefghijklmnopqrstuvwxyz

cipher : CIPHERABDFGJKLMNOQSTUVWXYZ

PROGRAM :

#include<stdio.h>

char monocipher\_encr(char);

char alpha[27][3] = { { 'a', 'C' }, { 'b', 'I' }, { 'c', 'P' }, { 'd', 'H' }, {

'e', 'E' }, { 'f', 'R' }, { 'g', 'A' }, { 'h', 'B' }, { 'i', 'D' }, {

'j', 'F' }, { 'k', 'G' }, { 'l', 'J' }, { 'm', 'K' }, { 'n', 'L' }, {

'o', 'M' }, { 'p', 'N' }, { 'q', 'O' }, { 'r', 'Q' }, { 's', 'S' }, {

't', 'T' }, { 'u', 'U' }, { 'v', 'V' }, { 'w', 'W' }, { 'x', 'X' }, {

'y', 'Y' }, { 'z', 'Z' } };

char str[20];

int main() {

char str[20], str2[20];

int i;

printf("\n Enter String..");

gets(str);

for (i = 0; str[i]; i++) {

str2[i] = monocipher\_encr(str[i]);

}

str2[i] = '\0';

printf("\n Before Decryption..%s", str);

printf("\n After Decryption..%s\n", str2);

}

char monocipher\_encr(char a) {

int i;

for (i = 0; i < 27; i++) {

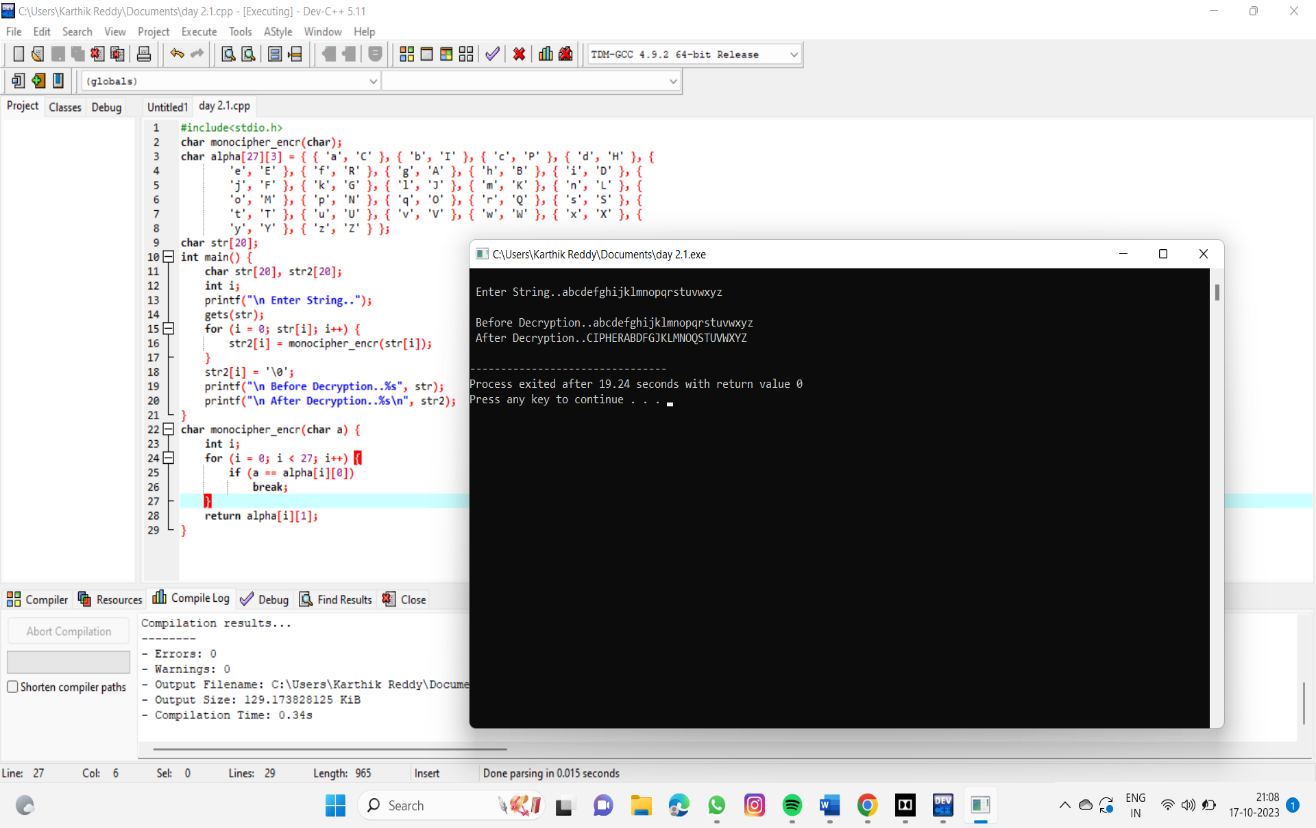
if (a == alpha[i][0])

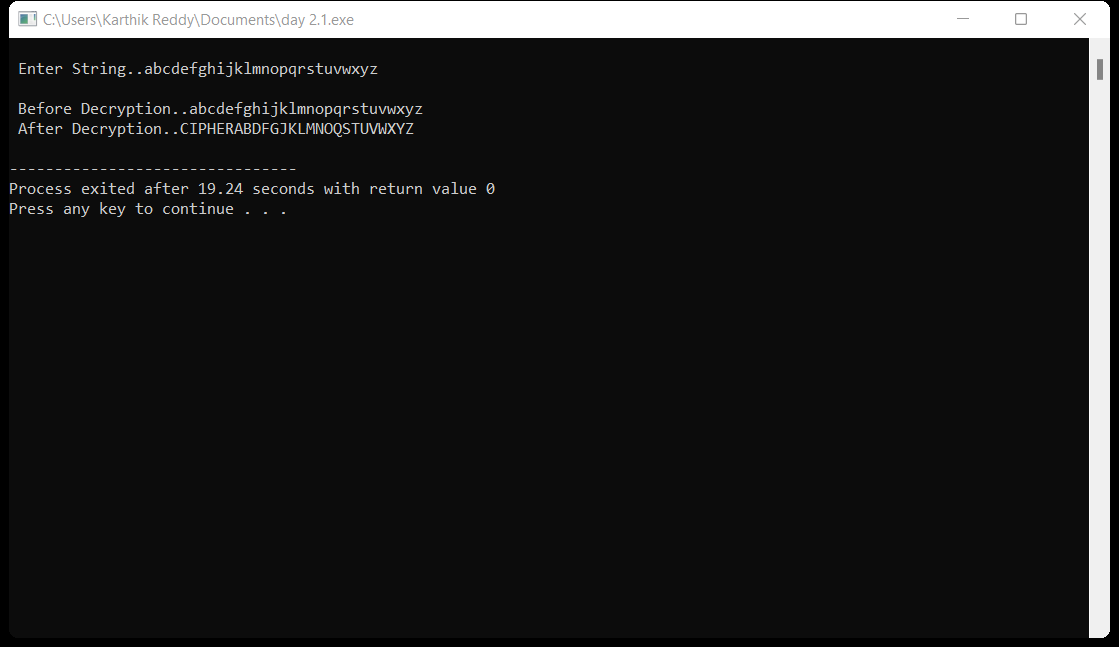
break;

}

return alpha[i][1];

}

OUTPUT :



QUESTION 2: Write a High level codefor PT-109 American patrol boat, under the command of Lieutenant John F.Kennedy, was sink by a Japanese destroyer, a message was received at an Australian wireless station in

Playfair code: KXJEY UREBE ZWEHE WRYTU HEYFS

KREHE GOYFI WTTTU OLKSY CAJPO BOTEI ZONTX

BYBNT GONEY CUZWRGDSON SXBOU YWRHE BAAHY USEDQ

PROGRAM :

import string

def generate\_cipher\_sequence(keyword):

"""

Given a keyword, generates the cipher sequence by appending the unused letters in normal order

to the keyword and returning the resulting string.

"""

keyword = keyword.upper()

unused\_letters = ''.join([letter for letter in string.ascii\_uppercase if letter not in keyword])

return keyword + unused\_letters

def encrypt(plaintext, keyword):

"""

Encrypts a plaintext message using the given keyword and the generated cipher sequence.

"""

cipher\_sequence = generate\_cipher\_sequence(keyword)

plaintext = plaintext.upper()

ciphertext = ''

for letter in plaintext:

if letter in string.ascii\_uppercase:

index = string.ascii\_uppercase.index(letter)

ciphertext += cipher\_sequence[index]

else:

ciphertext += letter

return ciphertext

def decrypt(ciphertext, keyword):

"""

Decrypts a ciphertext message using the given keyword and the generated cipher sequence.

"""

cipher\_sequence = generate\_cipher\_sequence(keyword)

plaintext = ''

for letter in ciphertext:

if letter in string.ascii\_uppercase:

index = cipher\_sequence.index(letter)

plaintext += string.ascii\_uppercase[index]

else:

plaintext += letter

return plaintext

# Example usage

plaintext ="AMERICAN PATROL BOAT, UNDER THE COMMAND OF LIEUTENANT JOHN F.KENNEDY, WAS SUNK BY A JAPANESE DESTROYER, A MESSAGE WAS RECEIVED A AN AUSTRALIAN STATION IN PLAY FAIR CODE"

keyword = "CIPHER"

# Encrypt the plaintext

ciphertext = encrypt(plaintext, keyword)

print("Ciphertext:")

print(ciphertext)

# Decrypt the ciphertext

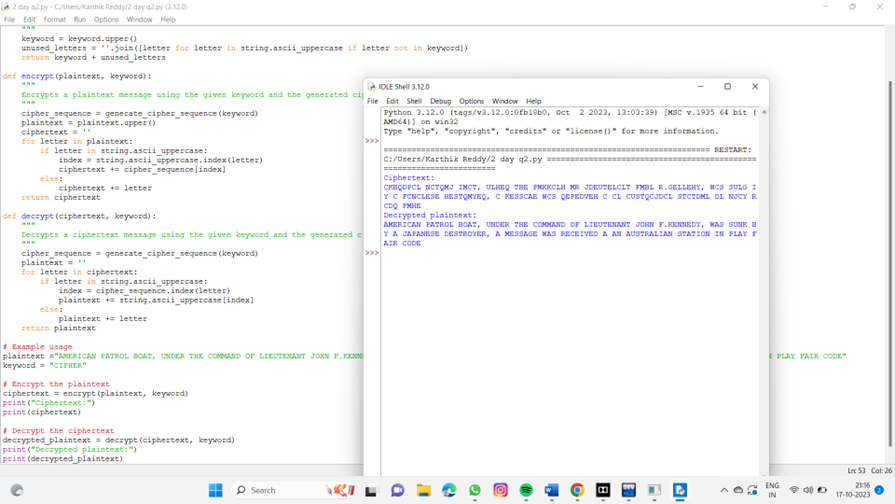
decrypted\_plaintext = decrypt(ciphertext, keyword)

print("Decrypted plaintext:")

print(decrypted\_plaintext)

oo

OUTPUT :



QUESTION 3 :

Write a High level code for Playfair matrix:

MFHIJK UNOPQ

ZV W X Y ELARG

DSTBC

Encrypt this message: Must see you over Cadogan West. Coming at once.

PROGRAM :

#include <stdio.h>

#include <string.h>

// Define Playfair matrix

char playfairMatrix[5][5] = {

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

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

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

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

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

};

// Function to find the coordinates of a character in the Playfair matrix

void findPosition(char ch, int \*row, int \*col) {

int i, j;

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

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

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

\*row = i;

\*col = j;

return;

}

}

}

}

// Function to encrypt a pair of characters

void encryptPair(char a, char b) {

int row1, col1, row2, col2;

findPosition(a, &row1, &col1);

findPosition(b, &row2, &col2);

if (row1 == row2) {

printf("%c%c", playfairMatrix[row1][(col1 + 1) % 5], playfairMatrix[row2][(col2 + 1) % 5]);

} else if (col1 == col2) {

printf("%c%c", playfairMatrix[(row1 + 1) % 5][col1], playfairMatrix[(row2 + 1) % 5][col2]);

} else {

printf("%c%c", playfairMatrix[row1][col2], playfairMatrix[row2][col1]);

}

}

int main() {

char message[1000];

printf("Enter the message to encrypt: ");

fgets(message, sizeof(message), stdin);

// Preprocess the message

int len = strlen(message);

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

if (message[i] >= 'a' && message[i] <= 'z') {

message[i] = message[i] - 'a' + 'A'; // Convert to uppercase

}

if (message[i] >= 'A' && message[i] <= 'Z') {

if (message[i] == 'J') {

message[i] = 'I'; // Replace 'J' with 'I'

}

} else {

continue; // Skip non-alphabetic characters

}

}

// Handle double letters

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

if (message[i] == message[i + 1]) {

memmove(&message[i + 1], &message[i + 2], len - i - 2);

len--;

}

}

// Encrypt the message in pairs

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

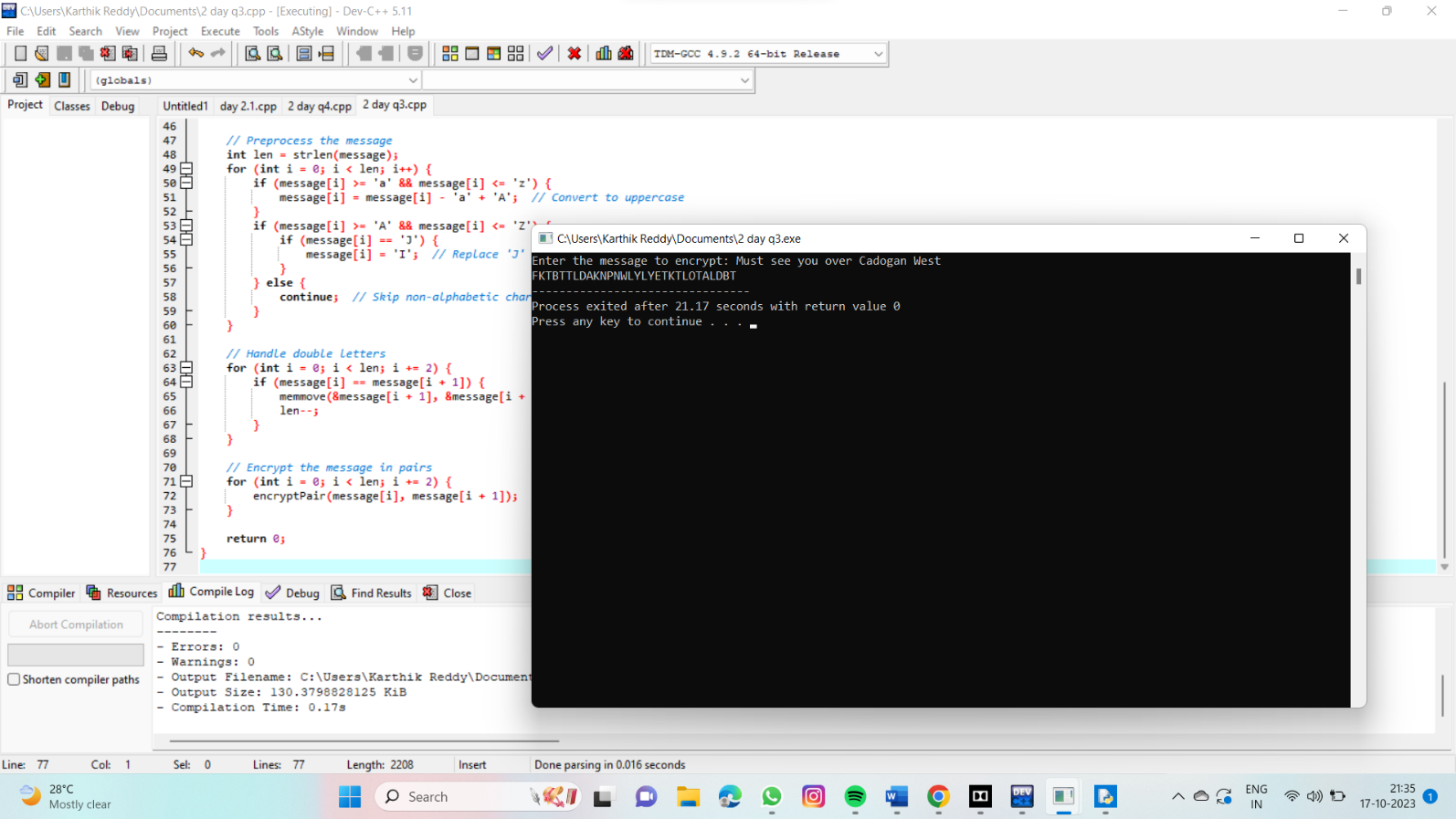
encryptPair(message[i], message[i + 1]);

}

return 0;

}

OUTPUT :



QUESTION 4:

Write a High level code for possible keys does the Playlar cipher have? Ignore the fact that some keys might produce identical encryption results. Express your answer as an approximate power of 2.

1. Now take into account the fact that some Playfair keys produce the same encryption resuks. Howmany effectively unique keys does the Playlair cipher have ?

PROGRAM :

#include <stdio.h>

#include <math.h>

unsigned long long factorial(int n) {

if (n == 0 || n == 1) return 1;

return n \* factorial(n - 1);

}

int main() {

int n = 25;

unsigned long long totalKeys = factorial(n);

double approxPowerOf2 = log2(totalKeys);

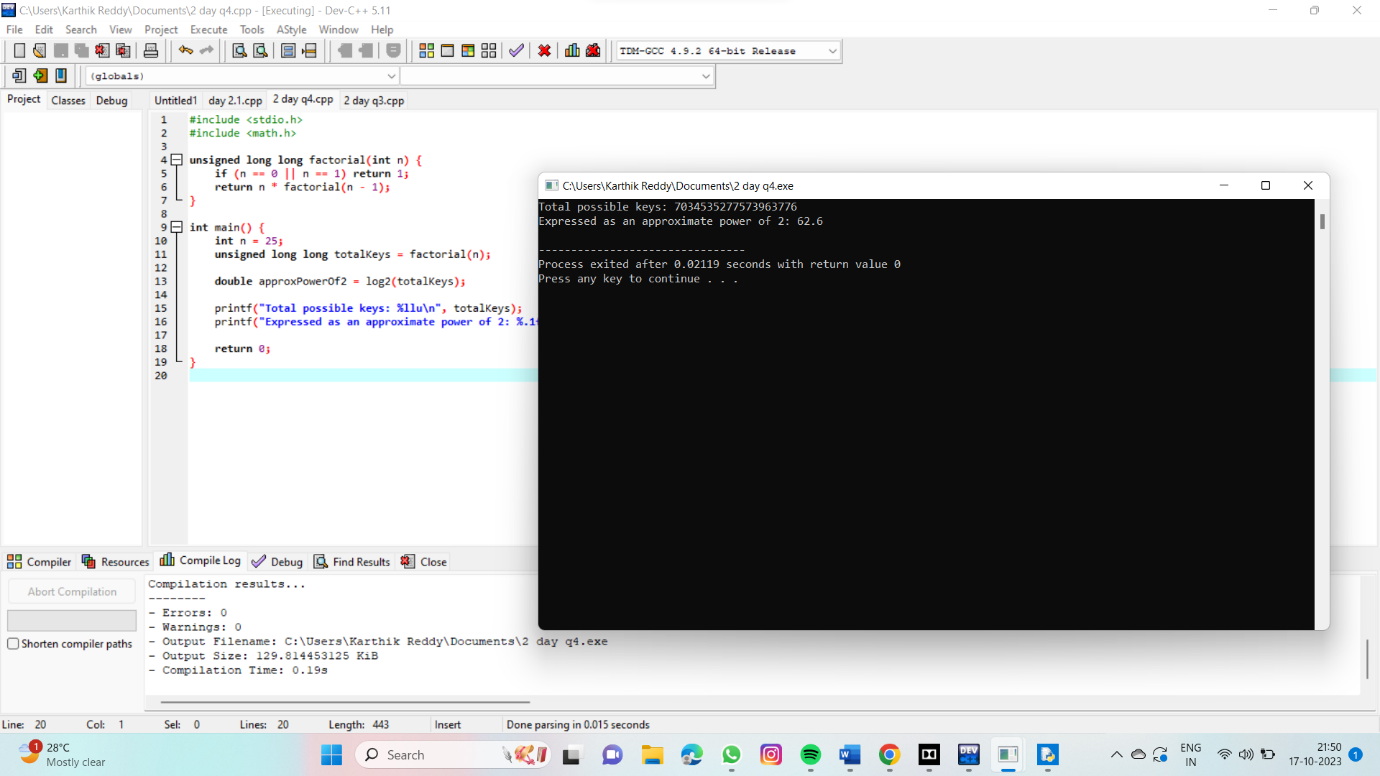
printf("Total possible keys: %llu\n", totalKeys);

printf("Expressed as an approximate power of 2: %.1f\n", approxPowerOf2);

return 0;

}

OUTPUT :



QUESTION 5 : 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 57

PROGRAM :

def hill\_encrypt(message, key):

message = message.replace(" ", "").lower()

if len(message) % 2 != 0:

message += 'x'

pairs = [message[i:i+2] for i in range(0, len(message), 2)]

numerical\_values = [ord(ch) - ord('a') for pair in pairs for ch in pair]

encrypted\_values = []

for i in range(0, len(numerical\_values), 2):

x = numerical\_values[i]

y = numerical\_values[i+1]

encrypted\_x = (key[0][0] \* x + key[0][1] \* y) % 26

encrypted\_y = (key[1][0] \* x + key[1][1] \* y) % 26

encrypted\_values.extend([encrypted\_x, encrypted\_y])

encrypted\_message = ''.join(chr(value + ord('a')) for value in encrypted\_values)

return encrypted\_message

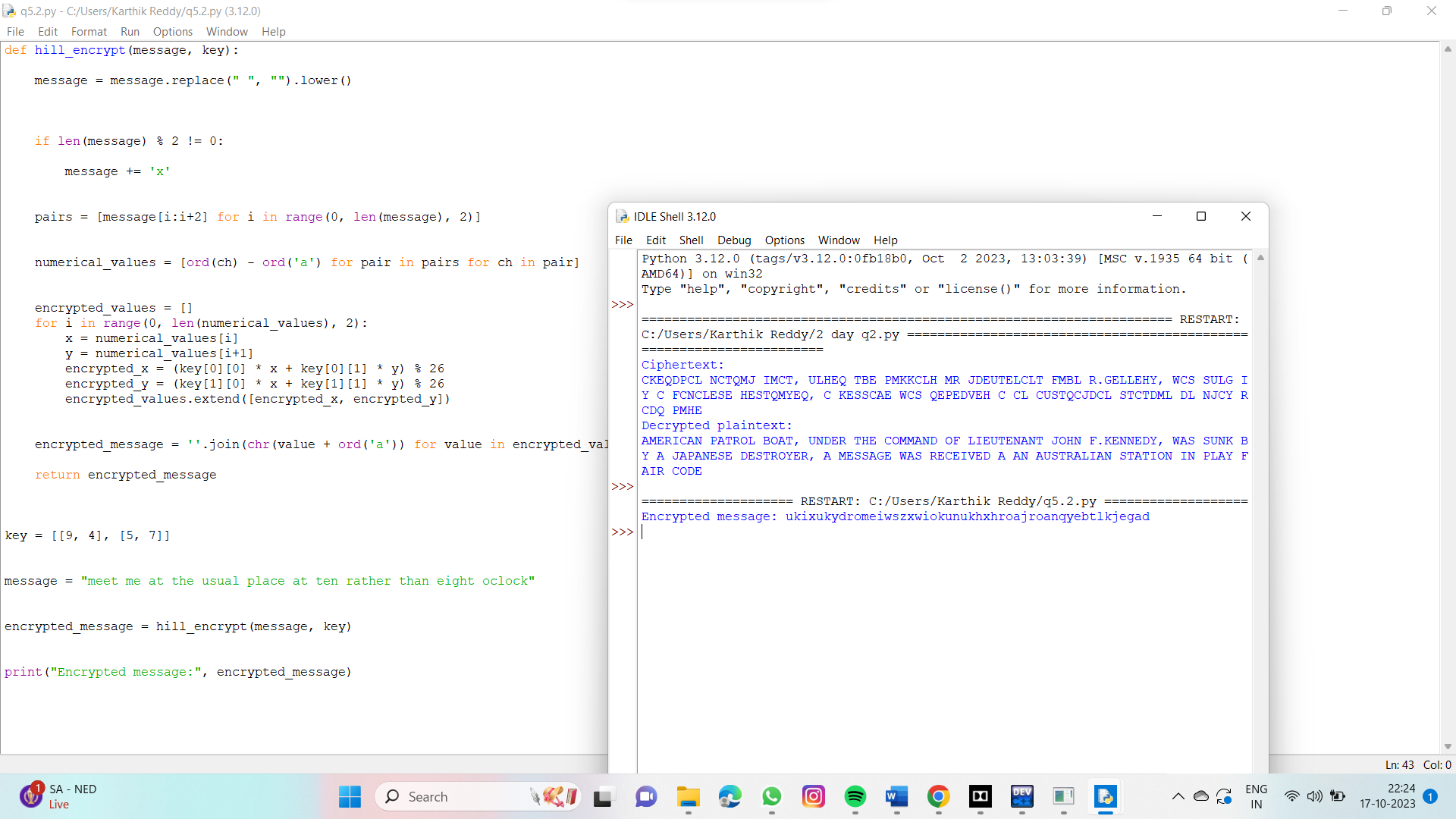
key = [[9, 4], [5, 7]]

message = "meet me at the usual place at ten rather than eight oclock"

encrypted\_message = hill\_encrypt(message, key)

print("Encrypted message:", encrypted\_message)

OUTPUT :



QUESTION 6: . Write a C program for Hill cipher succumbs to a known plaintext attack if sufficient plaintext-ciphertext pairs are provided. It is even easier to solve the Hill cipher if a chosen plaintext attack can be mounted.

PROGRAM :

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void hillCipherEncrypt(char \*plainText, char \*keyMatrix) {

int i, j, k, len = strlen(plainText);

int key[2][2], plain[2], cipher[2];

for (i = 0, k = 0; i < 2; i++) {

for (j = 0; j < 2; j++, k++) {

key[i][j] = keyMatrix[k] - 'A';

}

}

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

plain[0] = plainText[i] - 'A';

plain[1] = plainText[i + 1] - 'A';

cipher[0] = key[0][0] \* plain[0] + key[0][1] \* plain[1];

cipher[1] = key[1][0] \* plain[0] + key[1][1] \* plain[1];

cipher[0] %= 26;

cipher[1] %= 26;

printf("%c%c", cipher[0] + 'A', cipher[1] + 'A');

}

}

int main() {

char plainText[100], keyMatrix[5];

printf("Enter plaintext (uppercase alphabets only): ");

scanf("%s", plainText);

printf("Enter 2x2 key matrix (uppercase alphabets only): ");

scanf("%s", keyMatrix);

if (strlen(plainText) % 2 != 0 || strlen(keyMatrix) != 4) {

printf("Plaintext and key matrix lengths must be even and 4 characters, respectively.\n");

return 1;

}

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

if (!isupper(plainText[i])) {

printf("Invalid characters in plaintext. Use uppercase alphabets only.\n");

return 1;

}

}

for (int i = 0; i < 4; i++) {

if (!isupper(keyMatrix[i])) {

printf("Invalid characters in the key matrix. Use uppercase alphabets only.\n");

return 1;

}

}

printf("Ciphertext: ");

hillCipherEncrypt(plainText, keyMatrix);

printf("\n");

return 0;

}

OUTPUT :

