DAY 2

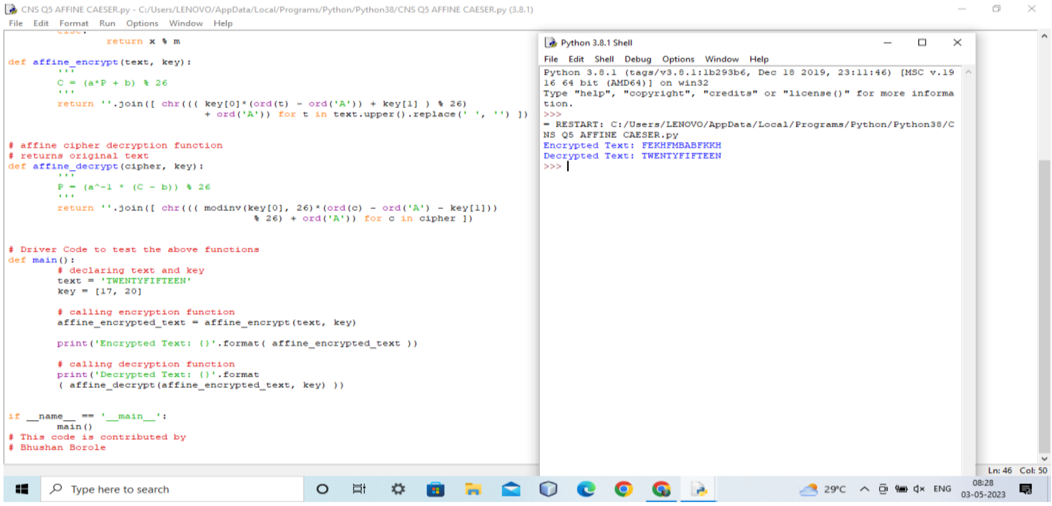
Q5. Write a C program for generalization of the Caesar cipher, known as the affine Caesar cipher, has the following form: For each plaintext letter p, substitute the ciphertext letter C: C = E([a, b], p) = (ap + b) mod 26 A basic requirement of any encryption algorithm is that it be one-to-one. That 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:

```
def egcd(a, b):

    x,y, u,v = 0,1, 1,0

    while a != 0:

        q, r = b//a, b%a

        m, n = x-u*q, y-v*q

        b,a, x,y, u,v = a,r, u,v, m,n

    gcd = b

    return gcd, x, y

def modinv(a, m):

    gcd, x, y = egcd(a, m)

    if gcd != 1:

        return None

else:

        return x % m

def affine_encrypt(text, key):

    return ''.join([ chr((( key[0]*(ord(t) - ord('A')) + key[1] ) % 26)

                + ord('A')) for t in text.upper().replace(' ', '') ])
```

```
def affine_decrypt(cipher, key):

        return ''.join([ chr((( modinv(key[0], 26)*(ord(c) - ord('A') - key[1]))

                % 26) + ord('A')) for c in cipher ])

def main():

        text = 'TWENTYFIFTEEN'

        key = [17, 20]

        affine_encrypted_text = affine_encrypt(text, key)

        print('Encrypted Text: {}'.format( affine_encrypted_text ))

        print('Decrypted Text: {}'.format

        ( affine_decrypt(affine_encrypted_text, key) ))

if __name__ == '__main__':

        main()
```

RESULT:



CNS Q5 AFFINE CAESER.py - C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/CNS Q5 AFFINE CAESER.py (3.8.1)
File Edit Format Run Options Window Help
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Ln: 46 Col: 50
Type here to search
29°C
ENG
08:28
03-05-2023

```
        else:
                return x % m

def affine_encrypt(text, key):
        '''
        C = (a*P + b) % 26
        '''
        return ''.join([ chr((( key[0]*(ord(t) - ord('A')) + key[1] ) % 26)
                                + ord('A')) for t in text.upper().replace(' ', '') ])


# affine cipher decryption function
# returns original text
def affine_decrypt(cipher, key):
        '''
        P = (a^-1 * (C - b)) % 26
        '''
        return ''.join([ chr((( modinv(key[0], 26)*(ord(c) - ord('A') - key[1]))
                                        % 26) + ord('A')) for c in cipher ])


# Driver Code to test the above functions
def main():
        # declaring text and key
        text = 'TWENTYFIFTEEN'
        key = [17, 20]

        # calling encryption function
        affine_encrypted_text = affine_encrypt(text, key)

        print('Encrypted Text: {}'.format( affine_encrypted_text ))

        # calling decryption function
        print('Decrypted Text: {}'.format
        ( affine_decrypt(affine_encrypted_text, key) ))


if __name__ == '__main__':
        main()
# This code is contributed by
# Bhushan Borole
```

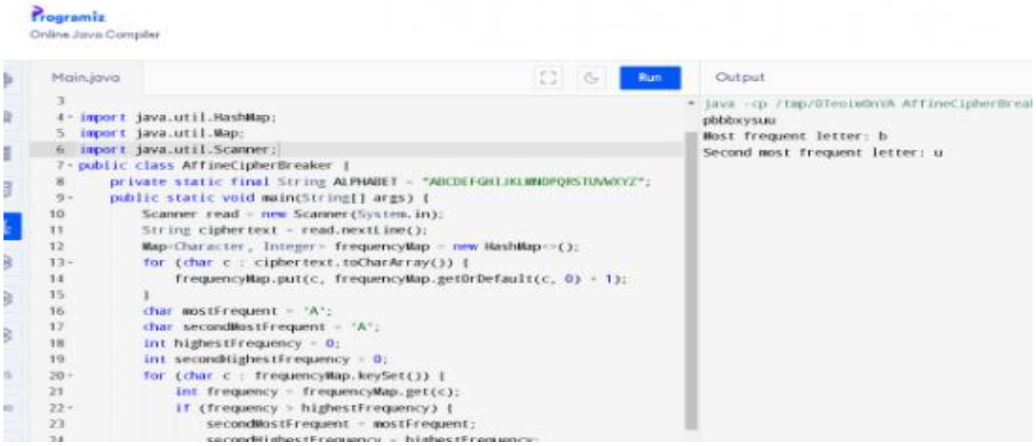
```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.19
16 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more informa
tion.
>>>
= RESTART: C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/C
NS Q5 AFFINE CAESER.py
Encrypted Text: FEKHFMBABFKKH
Decrypted Text: TWENTYFIFTEEN
>>> |
```

Q6. Write a High level code 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:

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
public class AffineCipherBreaker {
    private static final String ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    public static void main(String[] args) {
        Scanner read = new Scanner(System.in);
        String ciphertext = read.nextLine();
        Map<Character, Integer> frequencyMap = new HashMap<>();
        for (char c : ciphertext.toCharArray()) {
            frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);
        }
        char mostFrequent = 'A';
        char secondMostFrequent = 'A';
        int highestFrequency = 0;
        int secondHighestFrequency = 0;
        for (char c : frequencyMap.keySet()) {
            int frequency = frequencyMap.get(c);
            if (frequency > highestFrequency) {
                secondMostFrequent = mostFrequent;
                secondHighestFrequency = highestFrequency;
                mostFrequent = c;
                highestFrequency = frequency;
            } else if (frequency > secondHighestFrequency) {
                secondMostFrequent = c;
                secondHighestFrequency = frequency;
            }
        }
        System.out.println("Most frequent letter: " + mostFrequent);
        System.out.println("Second most frequent letter: " + secondMostFrequent);
    }
}
```

RESULT:



```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
public class AffineCipherBreaker {
    private static final String ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    public static void main(String[] args) {
        Scanner read = new Scanner(System.in);
        String ciphertext = read.nextLine();
        Map<Character, Integer> frequencyMap = new HashMap<>();
        for (char c : ciphertext.toCharArray()) {
            frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);
        }
        char mostFrequent = 'A';
        char secondMostFrequent = 'A';
        int highestFrequency = 0;
        int secondHighestFrequency = 0;
        for (char c : frequencyMap.keySet()) {
            int frequency = frequencyMap.get(c);
            if (frequency > highestFrequency) {
                secondMostFrequent = mostFrequent;
                secondHighestFrequency = highestFrequency;
```

```
java -cp /tmp/OTeoiwOnVA AffineCipherBreal
pbbbxysuu
Most frequent letter: b
Second most frequent letter: u
```

Q7. Write a C program 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:

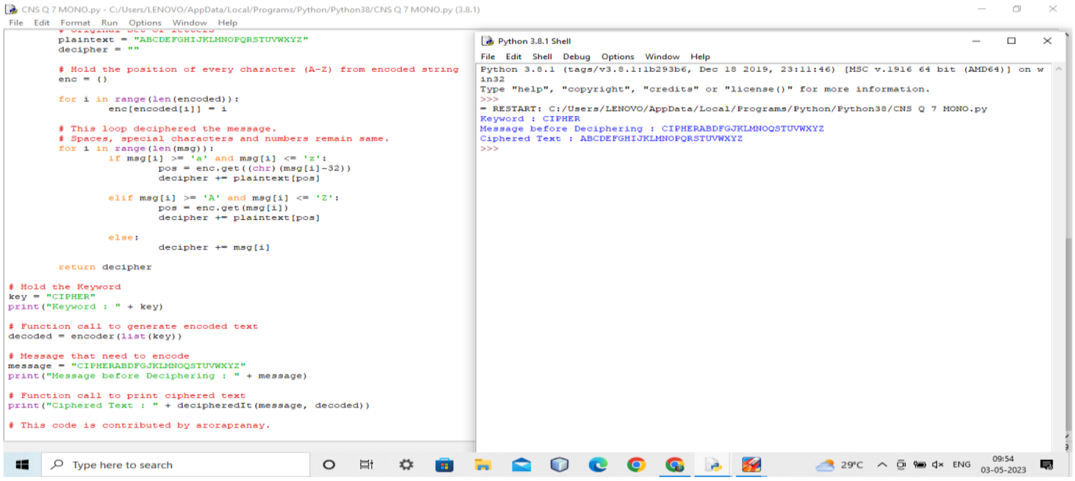
plain: a b c d e f g h i j k l m n o p q r s t u v w x y z

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:

```
import string
all_alphabets = list(string.ascii_uppercase)
def encoder(key):
    encoded = ""
    arr = [False]*26
    for i in range(len(key)):
        if key[i] >= 'A' and key[i] <= 'Z':
            if arr[ord(key[i]) - 65] == False:
                encoded += key[i]
                arr[ord(key[i]) - 65] = True
        elif key[i] >= 'a' and key[i] <= 'z':
            if arr[ord(key[i]) - 97] == False:
                encoded += chr(ord(key[i]) - 32)
                arr[ord(key[i]) - 97] = True
    for i in range(26):
        if arr[i] == False:
            arr[i] = True
            encoded += (chr(i + 65))
    return encoded
def decipheredIt(msg, encoded):
    plaintext = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    decipher = ""
    enc = {}

    for i in range(len(encoded)):
        enc[encoded[i]] = i
    for i in range(len(msg)):
        if msg[i] >= 'a' and msg[i] <= 'z':
            pos = enc.get((chr)(msg[i]-32))
            decipher += plaintext[pos]
        elif msg[i] >= 'A' and msg[i] <= 'Z':
            pos = enc.get(msg[i])
            decipher += plaintext[pos]
        else:
            decipher += msg[i]
    return decipher
key = "CIPHER"
print("Keyword : " + key)
decoded = encoder(list(key))
message = "CIPHERABDFGJKLMNOQSTUVWXYZ"
print("Message before Deciphering : " + message)
print("Ciphered Text : " + decipheredIt(message, decoded))
```

RESULT:


CNS Q 7 MONO.py - C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/CNS Q 7 MONO.py (3.8.1)
File Edit Format Run Options Window Help
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Type here to search
29°C
ENG
09:54
03-05-2023

```
        plaintext = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
        decipher = ""

        # Hold the position of every character (A-Z) from encoded string
        enc = {}

        for i in range(len(encoded)):
                enc[encoded[i]] = i

        # This loop deciphered the message.
        # Spaces, special characters and numbers remain same.
        for i in range(len(msg)):
                if msg[i] >= 'a' and msg[i] <= 'z':
                        pos = enc.get((chr)(msg[i]-32))
                        decipher += plaintext[pos]

                elif msg[i] >= 'A' and msg[i] <= 'Z':
                        pos = enc.get(msg[i])
                        decipher += plaintext[pos]

                else:
                        decipher += msg[i]

        return decipher

# Hold the Keyword
key = "CIPHER"
print("Keyword : " + key)

# Function call to generate encoded text
decoded = encoder(list(key))

# Message that need to encode
message = "CIPHERABDFGJKLMNOQSTUVWXYZ"
print("Message before Deciphering : " + message)

# Function call to print ciphered text
print("Ciphered Text : " + decipheredIt(message, decoded))

# This code is contributed by arorapranay.
```

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on w
in32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/CNS Q 7 MONO.py
Keyword : CIPHER
Message before Deciphering : CIPHERABDFGJKLMNOQSTUVWXYZ
Ciphered Text : ABCDEFGHIJKLMNOPQRSTUVWXYZ
>>>
```

Q8. Write a C program for Playfair matrix:

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 C

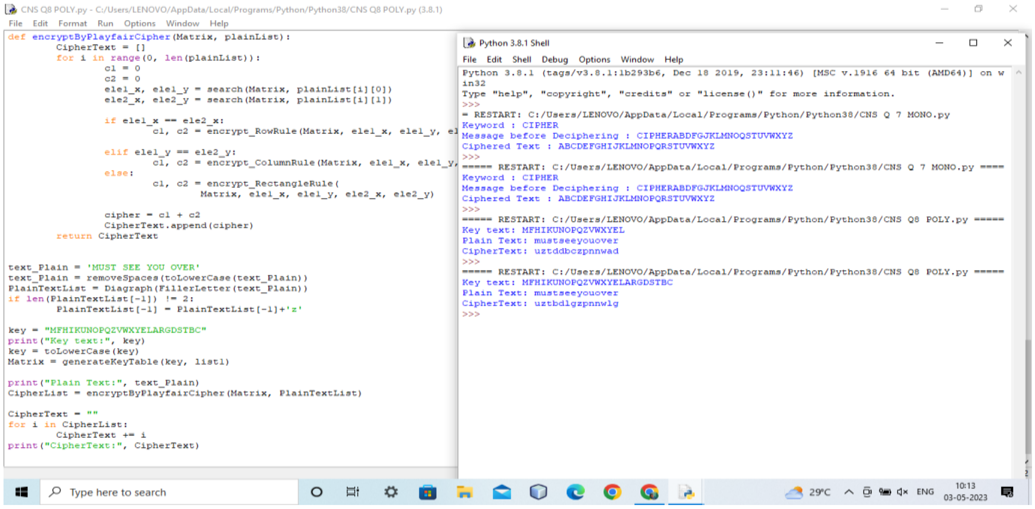
Encrypt this message: Must see you over

PROGRAM:

```
ef toLowerCase(text):

    return text.lower()

def removeSpaces(text):

    newText = ""

    for i in text:

        if i == " ":

            continue

        else:

            newText = newText + i

    return newText

def Diagraph(text):

    Diagraph = []

    group = 0

    for i in range(2, len(text), 2):

        Diagraph.append(text[group:i])


        group = i
    Diagraph.append(text[group:])

    return Diagraph

def FillerLetter(text):

    k = len(text)

    if k % 2 == 0:

        for i in range(0, k, 2):

            if text[i] == text[i+1]:

                new_word = text[0:i+1] + str('x') + text[i+1:]

                new_word = FillerLetter(new_word)

                break

            else:

                new_word = text

    else:

        for i in range(0, k-1, 2):

            if text[i] == text[i+1]:

                new_word = text[0:i+1] + str('x') + text[i+1:]

                new_word = FillerLetter(new_word)

                break

            else:

                new_word = text

    return new_word

list1 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm',

        'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
```

```
def generateKeyTable(word, list1):
    key_letters = []
    for i in word:
        if i not in key_letters:
            key_letters.append(i)
    compElements = []
    for i in key_letters:
        if i not in compElements:
            compElements.append(i)
    for i in list1:
        if i not in compElements:
            compElements.append(i)
    matrix = []
    while compElements != []:
        matrix.append(compElements[:5])
        compElements = compElements[5:]
    return matrix
def search(mat, element):
    for i in range(5):
        for j in range(5):
            if(mat[i][j] == element):
                return i, j
def encrypt_RowRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    if e1c == 4:
        char1 = matr[e1r][0]
    else:
        char1 = matr[e1r][e1c+1]

    char2 = ''
    if e2c == 4:
        char2 = matr[e2r][0]
    else:
        char2 = matr[e2r][e2c+1]
    return char1, char2
def encrypt_ColumnRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    if e1r == 4:
        char1 = matr[0][e1c]
    else:
        char1 = matr[e1r+1][e1c]
    char2 = ''
    if e2r == 4:
        char2 = matr[0][e2c]
    else:
        char2 = matr[e2r+1][e2c]
    return char1, char2
def encrypt_RectangleRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    char1 = matr[e1r][e2c]
    char2 = ''

    char2 = matr[e2r][e1c]

    return char1, char2

def encryptByPlayfairCipher(Matrix, plainList):

    CipherText = []

    for i in range(0, len(plainList)):

        c1 = 0

        c2 = 0

        ele1_x, ele1_y = search(Matrix, plainList[i][0])

        ele2_x, ele2_y = search(Matrix, plainList[i][1])

        if ele1_x == ele2_x:

            c1, c2 = encrypt_RowRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)

        elif ele1_y == ele2_y:

            c1, c2 = encrypt_ColumnRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)

        else:

            c1, c2 = encrypt_RectangleRule(

                Matrix, ele1_x, ele1_y, ele2_x, ele2_y)

        cipher = c1 + c2

        CipherText.append(cipher)

    return CipherText

text_Plain = 'MUST SEE YOU OVER'

text_Plain = removeSpaces(toLowerCase(text_Plain))

PlainTextList = Diagraph(FillerLetter(text_Plain))

if len(PlainTextList[-1]) != 2:
        PlainTextList[-1] = PlainTextList[-1]+'z'

key = "MFHIKUNOPQZVWXYELARGDSTBC"

print("Key text:", key)

key = toLowerCase(key)

Matrix = generateKeyTable(key, list1)

print("Plain Text:", text_Plain)

CipherList = encryptByPlayfairCipher(Matrix, PlainTextList)

CipherText = ""

for i in CipherList:

        CipherText += i

print("CipherText:", CipherText)
```

RESULT:


CNS Q8 POLY.py - C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/CNS Q8 POLY.py (3.8.1)
File Edit Format Run Options Window Help
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Type here to search
29°C
ENG
10:13
03-05-2023

```
def encryptByPlayfairCipher(Matrix, plainList):
        CipherText = []
        for i in range(0, len(plainList)):
                c1 = 0
                c2 = 0
                ele1_x, ele1_y = search(Matrix, plainList[i][0])
                ele2_x, ele2_y = search(Matrix, plainList[i][1])

                if ele1_x == ele2_x:
                        c1, c2 = encrypt_RowRule(Matrix, ele1_x, ele1_y, el

                elif ele1_y == ele2_y:
                        c1, c2 = encrypt_ColumnRule(Matrix, ele1_x, ele1_y,
                else:
                        c1, c2 = encrypt_RectangleRule(
                                Matrix, ele1_x, ele1_y, ele2_x, ele2_y)

                cipher = c1 + c2
                CipherText.append(cipher)
        return CipherText


text_Plain = 'MUST SEE YOU OVER'
text_Plain = removeSpaces(toLowerCase(text_Plain))
PlainTextList = Diagraph(FillerLetter(text_Plain))
if len(PlainTextList[-1]) != 2:
        PlainTextList[-1] = PlainTextList[-1]+'z'

key = "MFHIKUNOPQZVWXYELARGDSTBC"
print("Key text:", key)
key = toLowerCase(key)
Matrix = generateKeyTable(key, list1)

print("Plain Text:", text_Plain)
CipherList = encryptByPlayfairCipher(Matrix, PlainTextList)

CipherText = ""
for i in CipherList:
        CipherText += i
print("CipherText:", CipherText)
```

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on w
in32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/CNS Q 7 MONO.py
Keyword : CIPHER
Message before Deciphering : CIPHERABDFGJKLMNOQSTUVWXYZ
Ciphered Text : ABCDEFGHIJKLMNOPQRSTUVWXYZ
>>>
===== RESTART: C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/CNS Q 7 MONO.py ====
Keyword : CIPHER
Message before Deciphering : CIPHERABDFGJKLMNOQSTUVWXYZ
Ciphered Text : ABCDEFGHIJKLMNOPQRSTUVWXYZ
>>>
===== RESTART: C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/CNS Q8 POLY.py =====
Key text: MFHIKUNOPQZVWXYEL
Plain Text: mustseeyouover
CipherText: uztddbczpnnwad
>>>
===== RESTART: C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/CNS Q8 POLY.py =====
Key text: MFHIKUNOPQZVWXYELARGDSTBC
Plain Text: mustseeyouover
CipherText: uztbdlgzpnnwlg
>>>
```

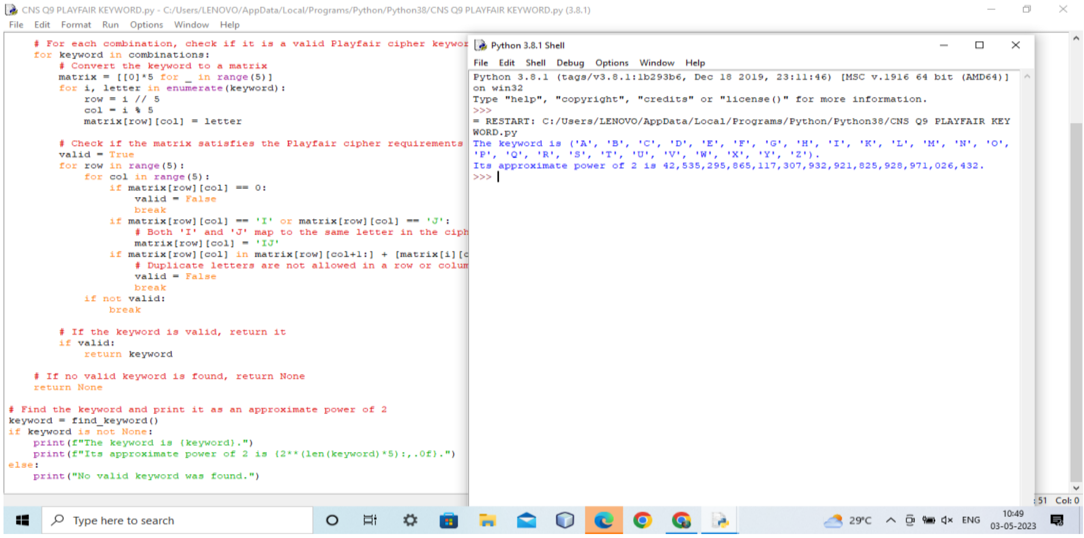
Q9. Write a high-level code for possible keys does the Playfair cipher have? Ignore the fact that some keys might produce identical encryption results. Express your answer as an approximate power of 2.

PROGRAM:

import itertools

```
def find_keyword():
    alphabet = 'ABCDEFGHIKLMNOPQRSTUVWXYZ'
    combinations = itertools.combinations(alphabet, 25)
    for keyword in combinations:
        matrix = [[0]*5 for _ in range(5)]
        for i, letter in enumerate(keyword):
            row = i // 5
            col = i % 5
            matrix[row][col] = letter
        valid = True
        for row in range(5):
            for col in range(5):
                if matrix[row][col] == 0:
                    valid = False
                    break
                if matrix[row][col] == 'I' or matrix[row][col] == 'J':

                    matrix[row][col] = 'IJ'
                if matrix[row][col] in matrix[row][col+1:] + [matrix[i][col] for i in range(row+1, 5)]:

                    valid = False
                    break
            if not valid:
                break
        if valid:
            return keyword

    return None

keyword = find_keyword()

if keyword is not None:

    print(f"The keyword is {keyword}.")

    print(f"Its approximate power of 2 is {2**(len(keyword)*5):,.0f}.")

else:

    print("No valid keyword was found.")
```

RESULT:



CNS Q9 PLAYFAIR KEYWORD.py - C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/CNS Q9 PLAYFAIR KEYWORD.py (3.8.1)
File Edit Format Run Options Window Help
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Type here to search
29°C
ENG
10:49
03-05-2023
51 Col: 0

```
    # For each combination, check if it is a valid Playfair cipher keywor
    for keyword in combinations:
        # Convert the keyword to a matrix
        matrix = [[0]*5 for _ in range(5)]
        for i, letter in enumerate(keyword):
            row = i // 5
            col = i % 5
            matrix[row][col] = letter

        # Check if the matrix satisfies the Playfair cipher requirements
        valid = True
        for row in range(5):
            for col in range(5):
                if matrix[row][col] == 0:
                    valid = False
                    break
                if matrix[row][col] == 'I' or matrix[row][col] == 'J':
                    # Both 'I' and 'J' map to the same letter in the ciph
                    matrix[row][col] = 'IJ'
                if matrix[row][col] in matrix[row][col+1:] + [matrix[i][c
                    # Duplicate letters are not allowed in a row or colum
                    valid = False
                    break
            if not valid:
                break

        # If the keyword is valid, return it
        if valid:
            return keyword

    # If no valid keyword is found, return None
    return None

# Find the keyword and print it as an approximate power of 2
keyword = find_keyword()
if keyword is not None:
    print(f"The keyword is {keyword}.")
    print(f"Its approximate power of 2 is {2**(len(keyword)*5):,.0f}.")
else:
    print("No valid keyword was found.")
```

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/CNS Q9 PLAYFAIR KEY
WORD.py
The keyword is ('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'O',
'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z').
Its approximate power of 2 is 42,535,295,865,117,307,932,921,825,928,971,026,432.
>>> 
```

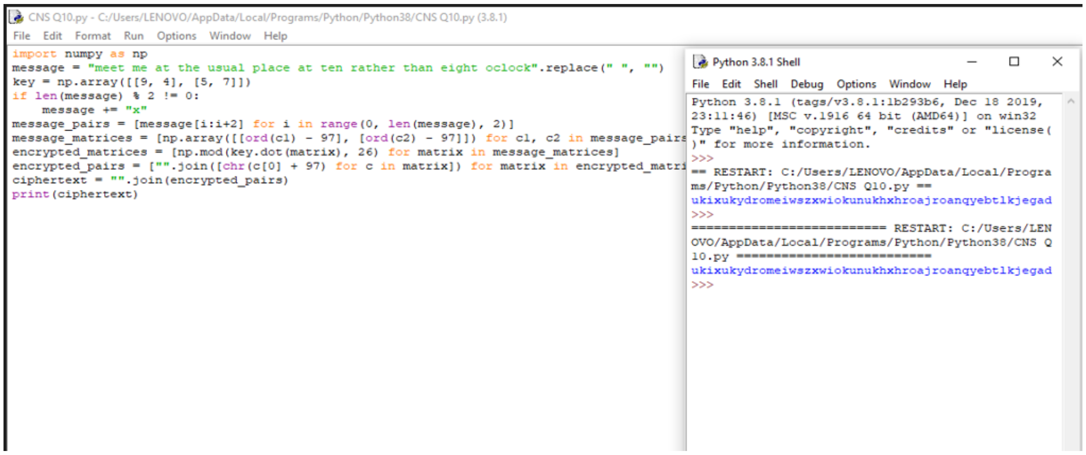
Q10. Write a high level code to Encrypt the message “meet me at the usual place at ten rather than eight oclock” using the Hill cipher with the key.9 4   5 7 a. Show your calculations and the result. b. Show the calculations for the corresponding decryption of the ciphertext to recover the original plaintext.

PROGRAM:

```
import numpy as np

message = "meet me at the usual place at ten rather than eight oclock".replace(" ", "")

key = np.array([[9, 4], [5, 7]])
```

```
if len(message) % 2 != 0:

   message += "x"

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

message_matrices = [np.array([[ord(c1) - 97], [ord(c2) - 97]]) for c1, c2 in message_pairs]

encrypted_matrices = [np.mod(key.dot(matrix), 26) for matrix in message_matrices]

encrypted_pairs = ["".join([chr(c[0] + 97) for c in matrix]) for matrix in encrypted_matrices]

ciphertext = "".join(encrypted_pairs)

print(ciphertext)
```

RESULT:


CNS Q10.py - C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/CNS Q10.py (3.8.1)
File Edit Format Run Options Window Help
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

```
import numpy as np
message = "meet me at the usual place at ten rather than eight oclock".replace(" ", "")
key = np.array([[9, 4], [5, 7]])
if len(message) % 2 != 0:
    message += "x"
message_pairs = [message[i:i+2] for i in range(0, len(message), 2)]
message_matrices = [np.array([[ord(c1) - 97], [ord(c2) - 97]]) for c1, c2 in message_pairs
encrypted_matrices = [np.mod(key.dot(matrix), 26) for matrix in message_matrices]
encrypted_pairs = ["".join([chr(c[0] + 97) for c in matrix]) for matrix in encrypted_matri
ciphertext = "".join(encrypted_pairs)
print(ciphertext)
```

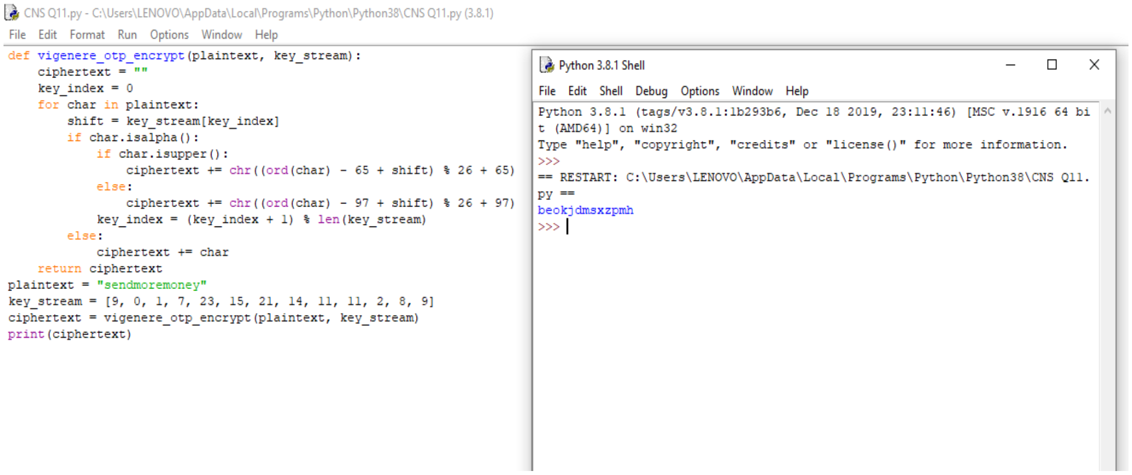
```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019,
23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license(
)" for more information.
>>>
== RESTART: C:/Users/LENOVO/AppData/Local/Progra
ms/Python/Python38/CNS Q10.py ==
ukixukydromeiwszxwiokunukhxhroajroanqyebtlkjegad
>>>
========================== RESTART: C:/Users/LEN
OVO/AppData/Local/Programs/Python/Python38/CNS Q
10.py ==========================
ukixukydromeiwszxwiokunukhxhroajroanqyebtlkjegad
>>>
```

Q11. Write a high level language program for one-time pad version of the Vigenère cipher. In this scheme, the key is a stream of random numbers between 1 and 26. For example, if the key is 3 19 5 . . . , then the first letter of the plaintext is encrypted with a shift of 3 letters, the second with a shift of 19 letters, the third with a shift of 5 letters, and so on.

PROGRAM:

```
def vigenere_otp_encrypt(plaintext, key_stream):

   ciphertext = ""

   key_index = 0

   for char in plaintext:
    shift = key_stream[key_index]
    if char.isalpha():
      if char.isupper():
        ciphertext += chr((ord(char) - 65 + shift) % 26 + 65)
      else:
        ciphertext += chr((ord(char) - 97 + shift) % 26 + 97)
      key_index = (key_index + 1) % len(key_stream)
    else:
      ciphertext += char
  return ciphertext
plaintext = "sendmoremoney"
key_stream = [9, 0, 1, 7, 23, 15, 21, 14, 11, 11, 2, 8, 9]
ciphertext = vigenere_otp_encrypt(plaintext, key_stream)
print(ciphertext)
```

RESULT:


CNS Q11.py - C:\Users\LENOVO\AppData\Local\Programs\Python\Python38\CNS Q11.py (3.8.1)
File Edit Format Run Options Window Help
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

```
def vigenere_otp_encrypt(plaintext, key_stream):
    ciphertext = ""
    key_index = 0
    for char in plaintext:
        shift = key_stream[key_index]
        if char.isalpha():
            if char.isupper():
                ciphertext += chr((ord(char) - 65 + shift) % 26 + 65)
            else:
                ciphertext += chr((ord(char) - 97 + shift) % 26 + 97)
            key_index = (key_index + 1) % len(key_stream)
        else:
            ciphertext += char
    return ciphertext
plaintext = "sendmoremoney"
key_stream = [9, 0, 1, 7, 23, 15, 21, 14, 11, 11, 2, 8, 9]
ciphertext = vigenere_otp_encrypt(plaintext, key_stream)
print(ciphertext)
```

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bi
t (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:\Users\LENOVO\AppData\Local\Programs\Python\Python38\CNS Q11.
py ==
beokjdmsxzpmh
>>> 
```