1. Write a C program for Caesar cipher involves replacing each letter of the alphabet with the letter standing k places further down the alphabet, for k in the range 1 through 25.

Code (.py)->>

```
def cc(txt,k):
    res = ""
    for char in txt:
        if char.isalpha():
          s=ord('A') if char.isupper() else ord('a')
           res+= chr((ord(char)-s+k) % 26+s)
        else:
           res+=char
    return res
txt = input("Entertext to encrypt: ")
k = int(input("Enter key : "))
etxt = cc(txt, k)
print("Encrypted text:", etxt)
```

output:

```
IDLE Shell 3.12.4
File  Edit  Shell  Debug  Options  Window  Help
    Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v
    AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more inform
>>>
    = RESTART: D:\CNS\CNS LAB\1.py
    Entertext to encrypt: meet me at the party
    Enter key : 3
    Encrypted text: phhw ph dw wkh sduwb
>>>
```

2. Write a C program for monoalphabetic substitution cipher maps a plaintext alphabet to a ciphertext alphabet, so that each letter of the plaintext alphabet maps to a single unique letter of the ciphertext alphabet.

Code:

```
import string


def gkey(word):
    word = "".join(dict.fromkeys(word))  # Remove duplicate letters
    rem = [c for c in string.ascii_lowercase if c not in word]  # Remaining letters
    return word + "".join(rem)  # Concatenate word and remaining letters


def enc(pt, key):
    ct = ""
    for c in pt:
        if c.isalpha():
            c = c.lower()
            idx = ord(c) - ord('a')
            ct += key[idx]
        else:
            ct += c
    return ct


def dec(ct, key):
    pt = ""
    for c in ct:
        if c.isalpha():
            c = c.lower()
            idx = key.index(c)
```

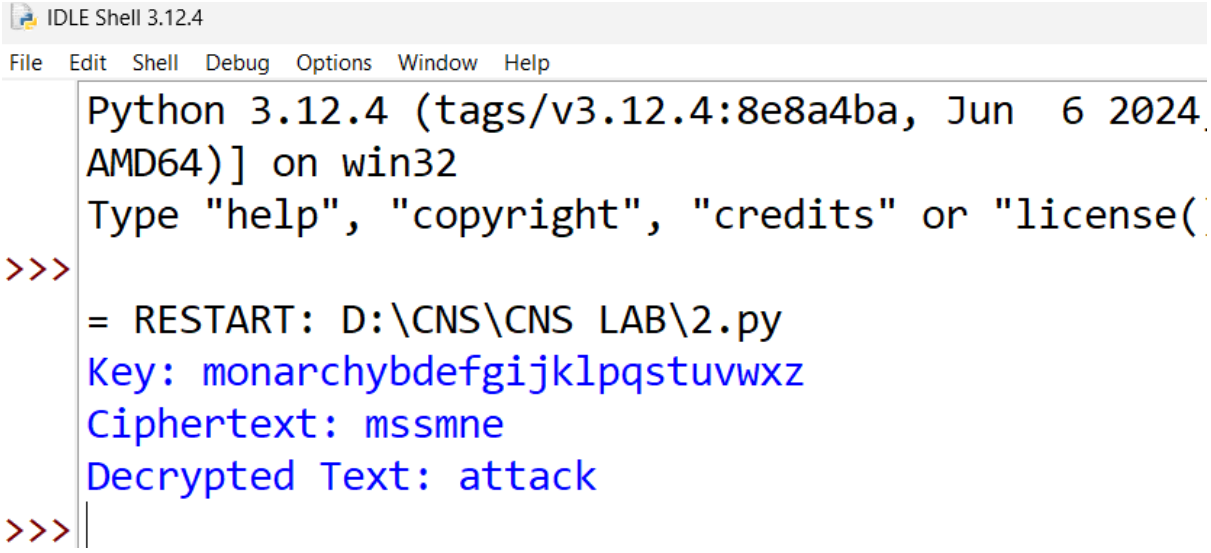
```
            pt += chr(idx + ord('a'))
        else:
            pt += c
    return pt


w = "monarchy"  # Shortened variable name for word
k = gkey(w)  # Shortened variable name for key
print(f"Key: {k}")


pt = "attack"
ct = enc(pt, k)
print(f"Ciphertext: {ct}")


dec_txt = dec(ct, k)
print(f"Decrypted Text: {dec_txt}")
```

output:



3. Write a C program for Playfair algorithm is based on the use of a 5 X 5 matrix of letters constructed using a keyword. Plaintext is encrypted two letters at a time using this matrix.

Code:

```
def ptxt(txt):
    txt = txt.upper().replace("J", "I").replace(" ", "")
    new_txt = ""
    i = 0
    while i < len(txt):
        if i == len(txt) - 1 or txt[i] == txt[i + 1]:
            new_txt += txt[i] + "X"
            i += 1
        else:
            new_txt += txt[i] + txt[i + 1]
            i += 2
    return new_txt

def gmat(key):
    key = key.upper().replace("J", "I")
    mat, used = [], set()

    for c in key + "ABCDEFGHIKLMNOPQRSTUVWXYZ":
        if c not in used:
            mat.append(c)
            used.add(c)

    return [mat[i:i+5] for i in range(0, 25, 5)]
```

```
def pos(mat, c):
    for r in range(5):
        if c in mat[r]:
            return r, mat[r].index(c)
    return None, None

def e(txt, mat):
    txt = ptxt(txt)
    etxt = ""

    for i in range(0, len(txt), 2):
        r1, c1 = pos(mat, txt[i])
        r2, c2 = pos(mat, txt[i+1])

        if r1 == r2:
            etxt += mat[r1][(c1+1) % 5] + mat[r2][(c2+1) % 5]
        elif c1 == c2:
            etxt += mat[(r1+1) % 5][c1] + mat[(r2+1) % 5][c2]
        else:
            etxt += mat[r1][c2] + mat[r2][c1]

    return etxt

key = "keyword"
txt = "hello world"
mat = gmat(key)
etxt = e(txt, mat)
```

print("Encrypted:", etxt)

Output:

```
IDLE Shell 3.12.4
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 202
AMD64)] on win32
Type "help", "copyright", "credits" or "license
>>>
= RESTART: D:\CNS\CNS LAB\3.py
Encrypted: GYIZSCOKCFBU
>>>
```

4. As you know, the most frequently occurring letter in English is e. Therefore, the first or second (or perhaps third?) most common character in the message is likely to stand for e. Also, e is often seen in pairs (e.g., meet, fleet, speed, seen, been, agree, etc.). Try to find a character in the ciphertext that decodes to e.

2.The most common word in English is “the.” Use this fact to guess the characters that stand for t and h. 3. Decipher the rest of the message by deducing additional words.

Code:

```
def cf(ct):
    f = {}
    for c in ct:
        if c.isalpha():
            f[c] = f.get(c, 0) + 1
    return sorted(f.items(), key=lambda x: x[1], reverse=True)


def fp(ct):
    p = {}
    for i in range(len(ct) - 1):
```

```
        if ct[i] == ct[i + 1]:
            k = ct[i] + ct[i + 1]
            p[k] = p.get(k, 0) + 1
    return sorted(p.items(), key=lambda x: x[1], reverse=True)

def gcl(fl):
    g = {}
    if len(fl) > 0:
        g[fl[0][0]] = 'E'
    if len(fl) > 1:
        g[fl[1][0]] = 'T'
    if len(fl) > 2:
        g[fl[2][0]] = 'H'
    return g

def sub(ct, g):
    d = ""
    for c in ct:
        d += g[c] if c in g else "_"
    return d

ct = "GDKKNXVNQKCGDKKNXVNQKC"

fl = cf(ct)
print("Top Letters:", fl)

pl = fp(ct)
print("Top Pairs:", pl)
```

```
g = gcl(fl)

print("Guessed Letters:", g)



dct = sub(ct, g)

print("Decoded Text:", dct)
```

Output:

```
IDLE Shell 3.12.4
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\CNS\CNS LAB\4.py
Top Letters: [('K', 6), ('N', 4), ('G', 2), ('D', 2), ('X', 2), ('V', 2), ('Q',
2), ('C', 2)]
Top Pairs: [('KK', 2)]
Guessed Letters: {'K': 'E', 'N': 'T', 'G': 'H'}
Decoded Text: H_EET__T_E_H_EET__T_E_
>>>
```

5. 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

code:

```
def rm_dup(kw):


    seen = set()

    return ''.join([c for c in kw if not (c in seen or seen.add(c))])
```

```
def gen_cipher(kw):

    kw = rm_dup(kw)
    alph = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    used = set(kw)
    cipher = list(kw)
    cipher.extend([l for l in alph if l not in used])
    return ''.join(cipher)

def enc(pt, cipher):

    alph = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    ct = []
    for c in pt:
        if c.isalpha():
            idx = alph.index(c.upper())
            ct.append(cipher[idx].lower() if c.islower() else cipher[idx])
        else:
            ct.append(c)
    return ''.join(ct)

def main():
    kw = "CIPHER"
    pt = input("Enter plaintext message: ")
    cipher = gen_cipher(kw)
    ct = enc(pt, cipher)
    print("Ciphertext:", ct)
```

```
if __name__ == "__main__":
    main()
```

output:

```
IDLE Shell 3.12.4
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\CNS\CNS LAB\5-1.py
Enter plaintext message: a b c d e f g h i j k l m n o p q r s t u v w x
Ciphertext: c i p h e r a b d f g j k l m n o q s t u v w x
>>>
```

5. 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 Cadogan West. Coming at once.

Code:

```
pf = [
    ['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']
]
```

```
def gp(ch):
    for r in range(len(pf)):
        for c in range(len(pf[r])):
            if pf[r][c] == ch:
                return r, c
    return None


def enc(txt):
    txt = txt.upper().replace(" ", "").replace("J", "I")
    txt = [ch for ch in txt if ch.isalpha()]


    p = []
    i = 0
    while i < len(txt):
        if i == len(txt) - 1 or txt[i] == txt[i + 1]:
            p.append((txt[i], 'X'))
            i += 1
        else:
            p.append((txt[i], txt[i + 1]))
            i += 2


    et = ""


    for a, b in p:
        r1, c1 = gp(a)
        r2, c2 = gp(b)
```

```
        if r1 == r2:
            et += pf[r1][(c1 + 1) % len(pf[r1])]
            et += pf[r2][(c2 + 1) % len(pf[r2])]
        elif c1 == c2:
            et += pf[(r1 + 1) % len(pf)][c1]
            et += pf[(r2 + 1) % len(pf)][c2]
        else:
            et += pf[r1][c2]
            et += pf[r2][c1]

    return et

m = "Must see you over Cadogan West. Coming at once."
em = enc(m)
print("Encrypted:", em)
```

output:

```
IDLE Shell 3.12.4
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [M
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more in
>>>
= RESTART: D:\CNS\CNS LAB\5-2.py
Encrypted: UZTBDLGZPNNWLGTGTUEROVLDBDUHFPERHWQSRZ
>>>
```

6. Write a C program 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

code:

```
MOD = 26


def inv(a, m):
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    return -1


def mul(k, blk):
    return [
        (k[0][0] * blk[0] + k[0][1] * blk[1]) % MOD,
        (k[1][0] * blk[0] + k[1][1] * blk[1]) % MOD
    ]


def txtToNum(txt):
    txt = txt.replace(" ", "")
    return [ord(c) - ord('a') for c in txt]


def numToTxt(nums):
    return ''.join(chr((n % MOD) + ord('a')) for n in nums)


def enc(txt, k):
    nums = txtToNum(txt)
    encTxt = []
    for i in range(0, len(nums), 2):
        blk = [nums[i], nums[i + 1]]
```

```
        encBlk = mul(k, blk)
        encTxt.extend(encBlk)
    return encTxt

def dec(ct, k):
    d = (k[0][0] * k[1][1] - k[0][1] * k[1][0]) % MOD
    dInv = inv(d, MOD)
    if dInv == -1:
        raise ValueError("Matrix is not invertible")
    kInv = [
        [k[1][1], -k[0][1]],
        [-k[1][0], k[0][0]]
    ]
    kInv = [
        [(dInv * kInv[0][0]) % MOD, (dInv * kInv[0][1]) % MOD],
        [(dInv * kInv[1][0]) % MOD, (dInv * kInv[1][1]) % MOD]
    ]
    decTxt = []
    for i in range(0, len(ct), 2):
        blk = [ct[i], ct[i + 1]]
        decBlk = mul(kInv, blk)
        decTxt.extend(decBlk)
    return decTxt

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

txt = "meet me at the usual place at ten rather than eight clock"
```

```
ctNums = enc(txt, k)

ct = numToTxt(ctNums)

print(f"Ciphertext: {ct}")


decNums = dec(ctNums, k)

decTxt = numToTxt(decNums)

print(f"Decrypted Text: {decTxt}")
```

output:

```
IDLE Shell 3.12.4
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\cgvij\AppData\Local\Programs\Python\Python312\6.py
Ciphertext: ukixukydromeiwszxwiokunukhxhroajroanqyebxfzxgc
Decrypted Text: meetmeattheusualplaceattenratherthaneightclock
>>>
```