1.Caesar cipher

```
def encrypt(text, s):
    result = ""

    for i in range(len(text)):
        char = text[i]

        if char.isupper():
            result += chr((ord(char) + s - 65) % 26 + 65)
        elif char.islower():
            result += chr((ord(char) + s - 97) % 26 + 97)
        else:
            result += ""

    return result


def decrypt(text, s):
    result = ""

    for i in range(len(text)):
        char = text[i]

        if char.isupper():
            result += chr((ord(char) - s - 65) % 26 + 65)
        elif char.islower():
            result += chr((ord(char) - s - 97) % 26 + 97)
        else:
            result += ""

    return result


while True:
    print("Enter a choice (1-3) ")
    choice = int(input("1.Encrypt\n2.Decrypt\n3.Exit :"))

    if choice == 1:
        text = input("Enter a text :")
        key = int(input("Enter a key :"))
        print("Encrypting...")

        ciphertext = encrypt(text, key)

        print("Cipher text : ", ciphertext)
```

```
    elif choice == 2:
        print("Decrypting...")
        print("Plain text : ", decrypt(ciphertext, key))
    elif choice == 3:
        break
    else:
        print("Invalid choice ")
```

2.Monoalphabetic cipher]

```
l = []

def encrypt(a,dict1):
    for x in a:
        y = dict1.get(x)
        l.append(y)

    return "".join(l)


def decrypt(dict1):
    keyList = list(dict1.keys())
    valueList = list(dict1.values())

    print("Decrypted value...")

    for i in l:
        position = valueList.index(i)
        print(keyList[position],end="")


dict2 = {
'A':'Z',
'B':'Y',
'C':'X',
'D':'W',
'E':'V',
'F':'U',
'G':'T',
'H':'S',
'I':'R',
'J':'Q',
'K':'P',
'L':'O',
```

```
'M':'N',
'N':'M',
'O':'L',
'P':'K',
'Q':'J',
'R':'I',
'S':'H',
'T':'G',
'U':'F',
'V':'E',
'W':'D',
'X':'C',
'Y':'B',
'Z':'A',
'a':'z',
'b':'y',
'c':'x',
'd':'w',
'e':'v',
'f':'u',
'g':'t',
'h':'s',
'i':'r',
'j':'q',
'k':'p',
'l':'o',
'm':'n',
'n':'m',
'o':'l',
'p':'k',
'q':'j',
'r':'i',
's':'h',
't':'g',
'u':'f',
'v':'e',
'w':'d',
'x':'c',
'y':'b',
'z':'a'

}

text = input("Enter a text : ")
encrypt = encrypt(text,dict2)

print("Encrypting...")
```

```
print("Cipher text : ",encrypt)

print("Decrypting...")
decrypt(dict2)
```

3.Diffie-Helman

```
import math

q = int(input("Enter a prime number : "))
a = int(input("Enter a primitive root :"))


Xa = int(input("Enter the private key of A :"))
Xb = int(input("Enter the private key of B : "))


Ya = math.pow(a, Xa) % q
Yb = math.pow(a, Xb) % q


print("Public key of  A  : ",Ya)
print("Public key of B : ",Yb)

Ka = math.pow(Yb,Xa)%q
Kb = math.pow(Ya,Xb)%q

print("Shared key for A : ",Ka)
print("Shared key for B : ",Kb)```
```

4.ECC

```
import tinyec
from tinyec import registry
import secrets

curve = registry.get_curve("brainpoolP256r1")


def compress_point(point):
    return hex(point.x) + hex(point.y % 2)[2:]


def getEnKey(pubKey):
    ciPrivateKey = secrets.randbelow(curve.field.n)
    ciPublicKey = ciPrivateKey * curve.g
```

```
    enKey = ciPublicKey * ciPrivateKey
    return (enKey, ciPublicKey)


senderPrivateKey = secrets.randbelow(curve.field.n)
senderPublicKey = senderPrivateKey * curve.g


print("Sender's private key : ", hex(senderPrivateKey))
print("Sender's public key : ", compress_point(senderPublicKey))

print("\n")
(enKeySender, ciPublicKeySender) = getEnKey(senderPublicKey)

print("Sender's ciphertext public key : ", compress_point(ciPublicKeySender))
print("Sender's encryption key : ", compress_point(enKeySender))
print("\n")

receiverPrivateKey = secrets.randbelow(curve.field.n)
receiverPublicKey = receiverPrivateKey * curve.g


print("Receiver's private key : ",hex(receiverPrivateKey))
print("Receiver's public key :",compress_point(receiverPublicKey))

print("\n")
(enKeyReceiver,ciPublicKeyReceiver) = getEnKey(receiverPublicKey)

print("Receiver's ciphertext public key : ",compress_point(ciPublicKeyReceiver))
print("Receiver encryption key : ",compress_point(enKeyReceiver))

5.Vigenere cipher




def vignere(key,message):

    message = message.lower()
    message = message.replace(' ','')
    m = len(key)

    cipherText = ""

    for i in range(len(message)):
        letter = message[i]
        k = key[i%m]
```

```
        cipherText+=chr((ord(letter)+k-97)%26+97)

    return cipherText


if __name__ == "__main__":
    print("Encrypting....")
    key = input("Enter a keystream : ")
    key = [ord(letter) - 97 for letter in key]

    message = input("Enter a message : ")
    cipherText = vignere(key,message)
    print("CipherText : ",cipherText)

    print("Decrypting...")

    key = [-1*k for k in key]
    plainText = vignere(key,cipherText)
    print("Plain text : ",plainText)
```

PART-B

1.Feistel cipher

```
# Fiestel cipher


s = input("Enter a string : ")


# This will convert string to ASCII⟶ then to 8-bit binary
result = "".join(format(ord(i),'08b') for i in s)


print("Result : ",result)

l = int(len(result)/2)

left = result[:l]
right = result[l:]


k = input("Enter a key : ")
key = "".join(format(ord(i),'08b') for i in k)
s = bin(int(right,2)+int(key,2))
```

```
answer = bin(int(s[2:],2)^int(left,2))

newr= answer[2:]
newl = right


newr,newl = newl,newr

s= bin(int(newr,2)+int(key,2))
ans = bin(int(s[2:],2) ^ int(newl,2))
nl = newr

nr = ans[2:]
nl,nr = nr,nl
cipher = nl+nr

if(len(cipher)≠len(result)):
    while(len(cipher)≠len(result)):
        cipher="0"+cipher

print(cipher)

plainText = ""
for i in range(0,len(cipher),8):
    temp = cipher[i:i+8]
    d = int(temp,2)
    plainText=plainText+chr(d)

print(plainText)

# Enter a string :helloworld
# Result :
0110100001100101011011000110110001101111011101110110111101110010011011000110010
# Enter a key :hello
# Cipher :
0110100001100101011011000110110001101111011101110110111101110010011011000110010
# Plaintext :  helloworld


2.Hill cipher


import numpy as np


l = list(map(int,input("Enter the key matrix : ").split()))
keyMatrix = np.array(l).reshape(2,2)
```

```
det = keyMatrix[0,0]*keyMatrix[1,1] - keyMatrix[0,1]*keyMatrix[1,0]


det = pow(int(det),1,26)

detInverse = pow(det,-1,26)

keyInverse  =
np.array([[keyMatrix[1,1],-keyMatrix[0,1]],[-keyMatrix[1,0],keyMatrix[0,0]]])

keyInverse = (detInverse*keyInverse)%26


def text_to_num(text):
    return [ord(char)-ord('A') for char in text]

def num_to_text(nums):
    return "".join([chr(num+ord('A')) for num in nums])



def encrypt(plainText):
    plainText = plainText.upper().replace(' ','')

    if(len(plainText)%2≠0):
        plainText+='X'


    cipherText = ""


    for i in range(0,len(plainText),2):
        block = np.array(text_to_num(plainText[i:i+2]))
        encryptedBlock = np.dot(keyMatrix,block)%26
        cipherText+=num_to_text(encryptedBlock)
    return cipherText



def decrypt(cipherText):
    cipherText = cipherText.upper().replace(' ','')
    plainText = ""

    for i in range(0,len(cipherText),2):
        block = np.array(text_to_num(cipherText[i:i+2]))
        decryptedBlock = np.dot(keyInverse,block)%26
        plainText+=num_to_text(decryptedBlock)
```

```
    return plainText


plainText = input("Enter a message :")
cipherText = encrypt(plainText)
print("Encrypted : ",cipherText)
print("Decrypted : ",decrypt(cipherText))


3.Playfair cipher

key = input("Enter a key :")
key = key.upper()

keysAlready = []

mapper = {}

matrix = [[0,0,0,0,0],
          [0,0,0,0,0],
          [0,0,0,0,0],
          [0,0,0,0,0],
          [0,0,0,0,0]]


i = 0
j = 0

#Key matrix creation

for ch in key:
    if(ch not in keysAlready):
        matrix[i][j] = ch
        mapper[ch] = (i,j)
        keysAlready.append(ch)
        j = (j+1)%5
        if(j == 0):
            i+=1


for ascii in range(65,91):
    ch = chr(ascii)
    if(ch not in keysAlready and ch≠"J"):
        matrix[i][j] = ch
        mapper[ch] = (i,j)
        keysAlready.append(ch)
        j = (j+1)%5
```

```
        if(j==0):
            i+=1


print(matrix)



# Plaintext modification

plainText = input("Enter a plaintext :")

plainText = plainText.upper()

for i in range(0,len(plainText)-1,2):
    if(plainText[i] == plainText[i+1]):
        plainText = plainText[:i+1]+"X"+plainText[i+1:]
if(len(plainText)%2≠0):
    plainText+="X"

print("Plain text : ",plainText)


# Encryption

cipherText = ""

for i in range(0,len(plainText)-1,2):
    char1 = plainText[i]
    char2 = plainText[i+1]

    (row1,col1) = mapper[char1]
    (row2,col2) = mapper[char2]

    if(row1 == row2):
        row = row1 = row2
        cipherText+=matrix[row][(col1+1)%5]
        cipherText+=matrix[row][(col2+1)%5]

    elif(col1 == col2):
        col = col1 =col2
        cipherText+=matrix[(row1+1)%5][col]
        cipherText+=matrix[(row2+1)%5][col]
    else:
        cipherText+=matrix[row1][col2]
        cipherText+=matrix[row2][col1]
```

```
print("Ciphertext : ",cipherText)

# Decryption

plainText = ""

for i in range(0,len(cipherText)-1,2):
    char1 = cipherText[i]
    char2 = cipherText[i+1]

    (row1,col1) = mapper[char1]
    (row2,col2) = mapper[char2]

    if(row1 == row2):
        row = row1 = row2
        plainText+=matrix[row][(col1-1)%5]
        plainText+=matrix[row][(col2-1)%5]

    elif(col1 == col2):
        col = col1 =col2
        plainText+=matrix[(row1-1)%5][col]
        plainText+=matrix[(row2-1)%5][col]
    else:
        plainText+=matrix[row1][col2]
        plainText+=matrix[row2][col1]

print("Plaintext  : ",plainText)
```

4.DES

```
import random

s = input("Enter a string : ")

result = ''.join(format(ord(i),'08b') for i in s)
answer = ""


for i in range(len(result)):
    if(i%8!=0):
        answer+=result[i]


l = int(len(answer)/2)
left = answer[:l]
right = answer[l:]

lt = [2,3,6,7,1,6,5,9]
```

```
keys = []

for i in range(0,8):
    newKey = ""
    newAnswer = ""

    nl=int(left,2)
    nl=bin(nl<<lt[i])
    num=2+lt[i]

    nr = int(right,2)
    nr = bin(nr<<lt[i])
    num=2+lt[i]

    newKey = nr[num:]+nl[num:]
    rm =[]

    while(len(rm)≠8):
        r = random.randint(0,len(newKey)-1)
        if(r not in rm):
            rm.append(r)

    for i in range(len(newKey)):
        if(i not in rm):
            newAnswer+=newKey[i]

    keys.append(newAnswer)

for i in range(0,len(keys)):
    print("Key ",i+1," = ",keys[i])
```

5.RSA

```
def gcd(a,b):
    while b:
        a,b = b,a%b
    return a


def RSA(p,q,msg):
    n = p*q
    phi = (p-1)*(q-1)

    for i in range(2,phi):
```

```
        if(gcd(i,phi)==1):
            e = i
            break

    j = 0

    while True:
        if(j*e%phi) == 1:
            d = j
            break
        j+=1

    c = (msg**e)%n

    print("Encrypted message : ",c)

    d = (c**d)%n

    print("Decrypted message : ",d)


p = int(input("Enter the value of p :"))
q = int(input("Enter the value of q :"))
msg = int(input("Emter a message :"))

RSA(p,q,msg)
```