# DAA CODE

## RANDOMIZED QUICH SHORT:

import random

def Ran\_quick\_sort(arr):

    n=len(arr)

    if n<=1:

        return arr

    pivot = random.choice(arr)

    left = []

    right = []

    pivot\_List=[]

    for i in arr:

        if(i < pivot):

            left.append(i)

        elif i<pivot:

            right.append(i)

        else:

            pivot\_List.append(i)

    return Ran\_quick\_sort(left) +pivot\_List+ Ran\_quick\_sort(right)

n=int(input("Enter the number of elements in array:"))

arr=[]

for i in range(0,n):

    a=int(input("Enter the element or array:"))

    arr.append(a)

print("The original Array or List:\n",arr)

r=Ran\_quick\_sort(arr)

print("Arral after shorted",r)

## RSA:

import random

def gcd(a,b):

    while b!=0:

        a=b

        b=a%b

    return a

def is\_prime(num):

    if num>1:

        for i in range(2,(num\*\*(1//2))+1):

            if(num% i==0):

                return False

    else:

        return False

    return True

def pr\_gen():

    while True:

        num=random.randint(2,100)

        if is\_prime(num):

            return num

def gcd(n1,n2):

    while(n2!=0):

        n1,n2=n2,n1%n2

    return n1

def co\_prim(a):

    while True:

        num=random.randint(2,a)

        if gcd(a,num)==1:

            return num

def invers\_mod(a,de):

    while True:

        # num=random.randint(2,a)

        for i in range(1,de):

            if(a\*i)%de==1:

                return i

        return None

def str\_dec(msg):

    int\_m=[]

    print("string to decimal")

    for i in msg:

        int\_m.append(ord(i))

    print(int\_m)

    return int\_m

def enc(msg,e,n):

    en=str\_dec(msg)

    new\_msg=[]

    for i in en:

        new\_msg.append((i\*\*e)%n)

    # en=[chr(i) for i in en]

    return new\_msg

def dec(new\_msg,d,n):

    omsg=[]

    for i in new\_msg:

        omsg.append((i\*\*d)%n)

    print("new string list is:",omsg)

    return ''.join(chr(i) for i in omsg)

msg=input("enter the input string:\n")

print("this is original message:", msg)

p=pr\_gen()

q=pr\_gen()

while p==q:

    q=pr\_gen()

n=p\*q

del\_n = (p-1)\*(q-1)

e = co\_prim(del\_n)

d = invers\_mod(e,del\_n)

new\_msg=[]

new\_msg=enc(msg,e,n)

print("encripted Message",new\_msg)

or\_msg=[]

or\_msg=dec(new\_msg,d,n)

# print(or\_msg)

print(msg)

## BAYER-MOOR ALGORITHM:

## RSA:

import random

def gcd(a,b):

    while b!=0:

        a=b

        b=a%b

    return a

def is\_prime(num):

    if num>1:

        for i in range(2,(num\*\*(1//2))+1):

            if(num% i==0):

                return False

    else:

        return False

    return True

def pr\_gen():

    while True:

        num=random.randint(2,100)

        if is\_prime(num):

            return num

def gcd(n1,n2):

    while(n2!=0):

        n1,n2=n2,n1%n2

    return n1

def co\_prim(a):

    while True:

        num=random.randint(2,a)

        if gcd(a,num)==1:

            return num

def invers\_mod(a,de):

    while True:

        # num=random.randint(2,a)

        for i in range(1,de):

            if(a\*i)%de==1:

                return i

        return None

def str\_dec(msg):

    int\_m=[]

    print("string to decimal")

    for i in msg:

        int\_m.append(ord(i))

    print(int\_m)

    return int\_m

def enc(msg,e,n):

    en=str\_dec(msg)

    new\_msg=[]

    for i in en:

        new\_msg.append((i\*\*e)%n)

    # en=[chr(i) for i in en]

    return new\_msg

def dec(new\_msg,d,n):

    omsg=[]

    for i in new\_msg:

        omsg.append((i\*\*d)%n)

    print("new string list is:",omsg)

    return ''.join(chr(i) for i in omsg)

msg=input("enter the input string:\n")

print("this is original message:", msg)

p=pr\_gen()

q=pr\_gen()

while p==q:

    q=pr\_gen()

n=p\*q

del\_n = (p-1)\*(q-1)

e = co\_prim(del\_n)

d = invers\_mod(e,del\_n)

new\_msg=[]

new\_msg=enc(msg,e,n)

print("encripted Message",new\_msg)

or\_msg=[]

or\_msg=dec(new\_msg,d,n)

# print(or\_msg)

print(msg)