CS800 Number Theory and Cryptography

Lab Report

Vidit Prashant Gala (252IS012)

I SEMESTER 2025-26

submitted to Mr. M.P. Singh sir

Department of Computer Science and Engineering

National Institute of Technology Karnataka, Surathkal
PO Srinivasnagar, Mangaluru-575025, Karnataka

2025-26

Index

EXPERIMENT 1

A. Basics (Data types, array length)

```
for i in range(0,6):
    print(i)

z = int(2)
y = int(3.5)
x = float(2)
print(x,y,z)

a = "Vidit Gala"
print(len(a))
print(a[0:3])
print(type(a))

b = 5
print(type(b))
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/basics.py
0
1
2
3
4
5
2.0 3 2
10
Vid
<class 'str'>
<class 'int'>
PS C:\Users\vidit>
```

B. Switch case for GCD and Calculator

```
def gcd_recursive(a,b):
    print("Greatest Common Divisor!")
    if b == 0:
            return a
    else:
            return gcd_recursive(b, a % b)
```

```
def Calculator():
    print("Welcome to Calculator")
    a = int(input("Enter the 1st number: "))
    b = int(input("Enter the 2nd number: "))
    opt = 0
    while(opt!=5):
        opt = int(input("Enter 1 to perform add, 2 for subtract, 3 for
multiply, 4 for division, and 5 to exit: "))
        match opt:
            case 1: print(a+b)
            case 2: print(a-b)
            case 3: print(a*b)
            case 4: print(a/b)
    print("Thank You!")


choice = int(input("Enter 1 for GCD, Enter 2 for Calculator: "))
match choice:
    case 1:
            x = int(input("Enter the 1st number: "))
            y = int(input("Enter the 2nd number: "))
            res = gcd_recursive(x,y)
            print(res)
    case 2:
            Calculator()
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/basics.py
Enter 1 for GCD, Enter 2 for Calculator: 1
Enter the 1st number: 23
Enter the 2nd number: 345
Greatest Common Divisor!
Greatest Common Divisor!
Greatest Common Divisor!
23
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/basics.py
Enter 1 for GCD, Enter 2 for Calculator: 2
Welcome to Calculator
Enter the 1st number: 23
Enter the 2nd number: 12
Enter 1 to perform add, 2 for subtract, 3 for multiply, 4 for division, and 5 to exit: 1
35
Enter 1 to perform add, 2 for subtract, 3 for multiply, 4 for division, and 5 to exit: 2
11
Enter 1 to perform add, 2 for subtract, 3 for multiply, 4 for division, and 5 to exit: 3
276
Enter 1 to perform add, 2 for subtract, 3 for multiply, 4 for division, and 5 to exit: 4
1.9166666666666667
Enter 1 to perform add, 2 for subtract, 3 for multiply, 4 for division, and 5 to exit: 5
Thank You!
PS C:\Users\vidit>
```

C. Star Pattern

```
n = int(input("Enter the value of n: "))
for i in range(1,n+1):
      for j in range(0,i):
        print("*", end = "")
      print()
for i in range(n-1,0,-1):
      for j in range(i,0,-1):
        print("*", end = "")
      print()
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/basics.py
Enter the value of n: 5
*
**
***
****
*****
****
***
**
*
PS C:\Users\vidit>
```

D. Finding a number in an array, then calculating its square

```
a = []
for i in range(5):
    b = int(input("Enter a value: "))
    a.append(b)

print(a)

num = int(input("Enter the number to find: "))

if num in a:
        print("Number found!")
        print("Square of the number is", num*num)
else:
        print("number not found!")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/basics.py
Enter a value: 76
Enter a value: 12
Enter a value: 13
Enter a value: 18
Enter a value: 5
[76, 12, 13, 18, 5]
Enter the number to find: 14
number not found!
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/basics.py
Enter a value: 1
Enter a value: 2
Enter a value: 3
Enter a value: 4
Enter a value: 5
[1, 2, 3, 4, 5]
Enter the number to find: 3
Number found!
Square of the number is 9
PS C:\Users\vidit>
```

EXPERIMENT 2

A. **Students with the highest marks to find in dictionary**

```
scores = {"Alice": 88, "Bob": 95, "Charlie": 70, "David": 95, "Eve": 88}  # students with highest marks to find
high = max(scores.values())
print(high)
st = [name for name, sc in scores.items() if sc == high]
# if(scores.values == high):
print(st)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp2.py
95
['Bob', 'David']
```

B. **Check if list has duplicate elements only using a set**

```
nums = [1,2,3,4,2,5,6,3]  # check if list has duplicate elements only using a set
s = set()
for i in range(len(nums)):
    s.add(nums[i])
print(s)
if len(s)<len(nums):
    check = True
else:
    check = False
print(check)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp2.py
{1, 2, 3, 4, 5, 6}
True
```

C. Symmetric diff

```
a = {1,2,3,4,5}
b = {4,5,6,7,8}

diff = a.union(b) - a.intersection(b)
print(diff)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp2.py
{1, 2, 3, 6, 7, 8}
```

D. Top 2 most frequent elements

```
list_nums = [2,3,5,2,3,7,11,5,2]  # top 2 most freq elems
maxim = list_nums.count(list_nums[0])
maxis = 0
for i in range(len(list_nums)):
    n = list_nums.count(list_nums[i])
    if n>maxim:
        maxim = n
        ind = i
    elif n<maxim and n>maxis:
        maxis = n
        index = i
print(f"1st Frequency->", maxim, "1st Elem->", list_nums[i])
print(f"2nd Frequency->", maxis, "2nd Elem->", list_nums[index])
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp2.py
{1, 2, 3, 6, 7, 8}
1st Frequency-> 3 1st Elem-> 2
2nd Frequency-> 2 2nd Elem-> 3
```

E. Flattening a list

```
flat_nums = [1,2,3,[4,5],(6,7),[8,[9,10]]]
lit = []

# print(len(flat_nums[5]))
def flatt(flat_nums):
    for i in range(len(flat_nums)):
        if type(flat_nums[i]) == list or type(flat_nums[i]) == tuple:
            flatt(flat_nums[i])
        else:
            lit.append(flat_nums[i])
flatt(flat_nums)
print(lit)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp2.py
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

F. Function inside a function

```
flat_nums = [1,2,3,4,5,6,7,8,9,10]
lit = []

def sum(flat_nums):
```

```
    sum=0
    for i in range(len(flat_nums)):
        sum = sum + flat_nums[i]
    return sum

# print(len(flat_nums[5]))
def flatt(flat_nums):
    s = sum(flat_nums)
    print(s)

flatt(flat_nums)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp2.py
55
```

G. Tuple

```
li = [(1,2),(3,4),(5,6)]
tu = ()
new = []
# print(len(li))
for i in range(len(li)):
    s = sum(li[i])
    # print(s)
    new.append(s)

# print(new)
tu = tuple(new)
print(tu)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp2.py
(3, 7, 11)
```

H. Prefix finding

```
words = ["flower", "flow", "flight", "app", "banana"]
new_li = []
for i in range(len(words)):
    for j in range(i+1, len(words)):
        if set(words[i]) & set(words[j]):
            print(words[i], words[j])
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp2.py
flower flow
flower flight
flow flight
app banana
```

I. Sum of list

```
list_int = [1,2,3,4,5,1,2]
li=[]
s = set()
def exact(list_int):
    for i in range(len(list_int)):
        if list_int.count(list_int[i]) == 2:
            s.add(list_int[i])

# exact(list_int)
print(s)
li = list(s)
print(li)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp2.py
{1, 2, 3}
[1, 2, 3]
```

J. Tuple

```
import numpy as np

arr= np.random.rand(5, 5)
arr_int = np.random.randint(0, 10, size=(5, 5))
print(arr_int)
sum=0
# s = sum(arr_int)
# print(s)
for i in range(5):
    for j in range(5):
        sum = sum + arr_int[i][j]

print(sum/25)
for i in range(5):
    for j in range(5):
        if arr_int[i][j] > (sum/25):
```

```
            arr_int[i][j] = 1
        else:
            arr_int[i][j] = 0

print(arr_int)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp2.py
[[3 9 1 6 7]
 [8 5 7 1 0]
 [2 8 4 3 3]
 [4 5 7 9 9]
 [4 1 9 1 2]]
4.72
[[0 1 0 1 1]
 [1 1 1 0 0]
 [0 1 0 0 0]
 [0 1 1 1 1]
 [0 0 1 0 0]]
PS C:\Users\vidit>
```

EXPERIMENT 3

A. Print hello world

```
print("Question 2 ============")
print("Hello World")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp3-2.py
Question 2 ============
Hello World
```

B. Does b divide a?

```
print("\nQuestion 3 ============")
a = int(input("Enter a number (a): "))
b = int(input("Enter a number (b) and b should not be 0:"))
if b == 0:
    print("Enter the value again")
    b = int(input("Enter a number (b) and b should not be 0:"))

if a%b == 0:
    print("This is true -  b | a")
else:
    print("This is false - b | a")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp3-2.py

Question 3 ============
Enter a number (a): 31
Enter a number (b) and b should not be 0:2
This is false - b | a
```

C. Find gcd by d max

```
print("\nQuestion 4 ============")
n1 = int(input("Enter a: "))
n2 = int(input("Enter b: "))

minim = min(n1,n2)
maxim = 0
if n1 == 0:
    maxim = n2
if n2 == 0:
    maxim = n1
for i in range(1,minim+1):
```

```
    if n1%i == 0 and n2%i == 0:
        d = i
        maxim = max(d,maxim)

print(maxim)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp3-2.py

Question 4 ============
Enter a: 56
Enter b: 4
4
```

D. Find gcd by euclid

```
print("\nQuestion 5 ============")
def gcd_recursive(a,b):
    if b == 0:
        return a
    else:
        return gcd_recursive(b, a % b)

m1 = int(input("Enter a number (a): "))
m2 = int(input("Enter a number (b): "))
num = gcd_recursive(m1, m2)
print(num)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp3-2.py

Question 5 ============
Enter a number (a): 33
Enter a number (b): 3
3
PS C:\Users\vidit>
```

E. Recursive gcd

```
print("\nQuestion 5 ============")
def gcd_recursive(a,b):
    if b == 0:
        return a
    else:
        return gcd_recursive(b, a % b)

m1 = int(input("Enter a number (a): "))
m2 = int(input("Enter a number (b): "))
num = gcd_recursive(m1, m2)
```

```
print(num)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp3-22.py

Question 5 ============
Enter a number (a): 46
Enter a number (b): 23
23
```

F. Find d = ax + by, where d = (a, b)

```
print("\nQuestion 6 ============")
n1 = int(input("Enter a number (a): "))

n2 = int(input("Enter a number (b): "))
def gcd(n1,n2):
    if n1 == 0:
        return(n2,0,1)
    else:
        d,x1,y1 = gcd(n2%n1, n1)
        x = y1 - (n2//n1)*x1
        y = x1
    return(d,x,y)

d, x, y = gcd(n1,n2)
print(d,x,y)
print(n1,"*",x,"(x)","+",n2,"*",y,"(y)","=",d)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp3-22.py

Question 6 ============
Enter a number (a): 21
Enter a number (b): 17
1 -4 5
21 * -4 (x) + 17 * 5 (y) = 1
```

G. Find count of numbers that have k divisors in a range

```
print("\nQuestion 7 ============")
a = int(input("Enter a number (a): "))
b = int(input("Enter a number (b): "))
k = int(input("Enter a number (k): "))
ans = 0
for i in range(a+1, b+1):
    count = 0
    for j in range(1,i+1):
        if i%j == 0:
            count = count + 1
```

```
    if count == k:
        ans = ans + 1

print(ans)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp3-22.py

Question 7 ============
Enter a number (a): 3
Enter a number (b): 19
Enter a number (k): 2
6
```

H. Find count of numbers and print them that have k divisors in a range

```
print("\nQuestion 8 ============")
a = int(input("Enter a number (a): "))
b = int(input("Enter a number (b): "))
n = int(input("Enter a number (n): "))
s = set()
ans = 0
for i in range(a+1, b+1):
    count = 0
    for j in range(1,i+1):
        if i%j == 0:
            count = count + 1
    if count == n:
        ans = ans + 1
        s.add(i)

print(ans)
print(s)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp3-22.py

Question 8 ============
Enter a number (a): 6
Enter a number (b): 26
Enter a number (n): 3
2
{9, 25}
```

I. Find d for (85, 289) and also x, y

```
print("\nQuestion 9 ============")
n1 = 85
n2 = 289
def gcd(n1,n2):
```

```
    if n1 == 0:
        return(n2,0,1)
    else:
        d,x1,y1 = gcd(n2%n1, n1)
        x = y1 - (n2//n1)*x1
        y = x1
    return(d,x,y)

d, x, y = gcd(n1,n2)
print(d,x,y)
print(n1,"*",x,"(x)","+",n2,"*",y,"(y)","=",d)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp3-22.py

Question 9 ============
17 7 -2
85 * 7 (x) + 289 * -2 (y) = 17
```

J. Find number of steps to find gcd

```
print("\nQuestion 10 ============")
def gcd_recurs(a,b,k):
    if b == 0:
        k = k+1
        return a, k
    else:
        k = k+1
        return gcd_recurs(b, a % b,k)

m1 = int(input("Enter a number (a): "))
m2 = int(input("Enter a number (b): "))
k = 0
num,ans = gcd_recurs(m1, m2,k)
print(num)
print("The number of steps are: ", ans)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp3-22.py

Question 10 ============
Enter a number (a): 4
Enter a number (b): 16
4
The number of steps are:  3
```

K. GCD of (m1+m2, m1-m2) = 1 or 2 iff gcd(m1, m2) = 1

```
print("\nQuestion 11 ============")
def gcd_recursive(a,b):
    if b == 0:
        return a
    else:
        return gcd_recursive(b, a % b)

m1 = int(input("Enter a number (a): "))
m2 = int(input("Enter a number (b): "))
num = gcd_recursive(m1, m2)
if num == 1:
    num1 = gcd_recursive(m1+m2,m1-m2)
    print("GCD of ", m1+m2, "and", m1-m2, "is: ")
    print(num1)
else:
    print("GCD IS NOT 1")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp3-22.py

Question 11 ============
Enter a number (a): 45
Enter a number (b): 23
GCD of  68 and 22 is:
2
```

EXPERIMENT 4

A. Solve diophantine equations

```
import math
def extended_gcd(a, b):
    if a == 0:
        return b, 0, 1
    gcd, x1, y1 = extended_gcd(b % a, a)
    x = y1 - (b // a) * x1
    y = x1
    return gcd, x, y
def solve_linear_diophantine(a, b, c):
    gcd, x0, y0 = extended_gcd(a, b)

    if c % gcd != 0:
        return "No integer solutions exist."

    x_1 = x0 * (c // gcd)
    y_1 = y0 * (c // gcd)

    b_p = b // gcd
    a_p = a // gcd

    def get_solution(k):
        x_k = x_1 + k * b_p
        y_k = y_1 - k * a_p
        return x_k, y_k

    return get_solution, gcd
a = 56
b = 72
c = 40
solution_generator, common_divisor = solve_linear_diophantine(a, b, c)
if isinstance(solution_generator, str):
    print(solution_generator)
else:
    print(f"GCD({a}, {b}) = {common_divisor}")
    print("General solutions (x, y) for 56x + 72y = 40:")
    for k in range(-10,10):
        x, y = solution_generator(k)
        print(f"k = {k}: x = {x}, y = {y}")
```

```
a = 24
b = 138
c = 18
solution_generator, common_divisor = solve_linear_diophantine(a, b, c)
if isinstance(solution_generator, str):
    print(solution_generator)
else:
    print(f"GCD({a}, {b}) = {common_divisor}")
    print("General solutions (x, y) for 24x + 138y = 18:")
    for k in range(-10,10):
        x, y = solution_generator(k)
        print(f"k = {k}: x = {x}, y = {y}")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe
GCD(56, 72) = 8
General solutions (x, y) for 56x + 72y = 40:
k = -10: x = -70, y = 55
k = -9: x = -61, y = 48
k = -8: x = -52, y = 41
k = -7: x = -43, y = 34
k = -6: x = -34, y = 27
k = -5: x = -25, y = 20
k = -4: x = -16, y = 13
k = -3: x = -7, y = 6
k = -2: x = 2, y = -1
k = -1: x = 11, y = -8
k = 0: x = 20, y = -15
k = 1: x = 29, y = -22
k = 2: x = 38, y = -29
k = 3: x = 47, y = -36
k = 4: x = 56, y = -43
k = 5: x = 65, y = -50
k = 6: x = 74, y = -57
k = 7: x = 83, y = -64
k = 8: x = 92, y = -71
k = 9: x = 101, y = -78
```

```
GCD(24, 138) = 6
General solutions (x, y) for 24x + 138y = 18:
k = -10: x = -212, y = 37
k = -9: x = -189, y = 33
k = -8: x = -166, y = 29
k = -7: x = -143, y = 25
k = -6: x = -120, y = 21
k = -5: x = -97, y = 17
k = -4: x = -74, y = 13
k = -3: x = -51, y = 9
k = -2: x = -28, y = 5
k = -1: x = -5, y = 1
k = 0: x = 18, y = -3
k = 1: x = 41, y = -7
k = 2: x = 64, y = -11
k = 3: x = 87, y = -15
k = 4: x = 110, y = -19
k = 5: x = 133, y = -23
k = 6: x = 156, y = -27
k = 7: x = 179, y = -31
k = 8: x = 202, y = -35
k = 9: x = 225, y = -39
```

B. Recursive gcd

```
print("\nQuestion 2 ============")
def gcd_recursive(a,b):
    if b == 0:
        return a
    else:
        return gcd_recursive(b, a % b)

m1 = int(input("Enter a number (a): "))
m2 = int(input("Enter a number (b): "))
num = gcd_recursive(m1,m2)
print(num)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp4.py

Question 2 ============
Enter a number (a): 5
Enter a number (b): 13
1
```

C. Gcd of a list

```
print("\nQuestion 3 ============")
a = [6,3]
for i in range(4):
    x = int(input("Enter number:-"))
    a.append(x)
print(a)
def gcd_recursive(a,b):
    if b == 0:
        return a
    else:
        return gcd_recursive(b, a % b)
g = gcd_recursive(a[0],a[1])
d = g
for i in range(2,5):
    d = gcd_recursive(d,a[i])
print(d)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp4.py

Question 3 ============
Enter number:-18
Enter number:-27
Enter number:-30
Enter number:-12
[6, 3, 18, 27, 30, 12]
3
```

D. If array number is mod 10 or not?

```
print("\nQuestion 4 ============")
arr = []
n = int(input("Enter value of n: "))
for i in range(n):
    val = int(input("Enter array values: "))
    arr.append(val)
print(arr)
s = ""
for i in range(n):
    v = arr[i]%10
    s = s + str(v)
n = int(s)
print(n)
if n%10 == 0: print("TRUE")
else: print("False")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe

Question 4 ============
Enter value of n: 5
Enter array values: 12380
Enter array values: 23
Enter array values: 8
Enter array values: 9
Enter array values: 0
[12380, 23, 8, 9, 0]
3890
TRUE
```

E. Number of steps to convert a or b to 0mod3

```
print("\nQuestion 5 ============")
a = int(input("Enter the value of a: "))
b = int(input("Enter the value of b: "))
count = 0
while a%3!=0 and b%3!=0:
```

```
    if a>b:
        a = abs(a-b)
    else:
        b = abs(a-b)
    count = count+1

print("No.of steps = ", count)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp4.py

Question 5 ============
Enter the value of a: 7
Enter the value of b: 19
No.of steps =  1
```

F. Count of numbers divisible by 2 or 3 or 5 in a range

```
print("\nQuestion 6 ============")
a = int(input("Enter the value of a: "))
b = int(input("Enter the value of b: "))
arr = []
for i in range(a,b+1):
    if i%2==0 or i%3==0 or i%5==0:
        arr.append(i)

print(arr)
print(len(arr))
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Exp4.py

Question 6 ============
Enter the value of a: 2
Enter the value of b: 33
[2, 3, 4, 5, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25, 26, 27, 28, 30, 32, 33]
24
PS C:\Users\vidit>
```

EXPERIMENT 5

A. Find 2 primes whose sum = n

```
a = int(input("Enter a number(>2): "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []
for i in range(2,a):
    if(isPrime(i)):
        li.append(i)

print(li)
check = False
for i in range(len(li)):
    for j in range(i+1,len(li)):
        if li[i] + li[j] == a:
            check = True
            print(li[i], li[j])
            break
        else:
            check = False
    if check==True:
        break

if(check==False): print(-1)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-1.py
Enter a number(>2): 18
[2, 3, 5, 7, 11, 13, 17]
5 13
PS C:\Users\vidit>
```

B. Prove that for even n, there exist primes p and q (n-p, n-q) = 1.

```
n = int(input("Enter a number(>6): "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True
```

```
def gcd_recursive(a,b):
    if b == 0:
        return a
    else:
        return gcd_recursive(b, a % b)

li = []
for i in range(2,n):
    if(isPrime(i)):
        li.append(i)

for i in range(len(li)-2):
    num = gcd_recursive(n-li[i],n-li[i+2])
    print(num, "p and q are: ", li[i], li[i+2])
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-2.py
Enter a number(>6): 25
1 p and q are:  2 5
2 p and q are:  3 7
2 p and q are:  5 11
6 p and q are:  7 13
2 p and q are:  11 17
6 p and q are:  13 19
2 p and q are:  17 23
PS C:\Users\vidit>
```

C. Find all primes as sums and differences of two primes

```
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []
for i in range(2,n):
    if(isPrime(i)):
        li.append(i)
print(li)
count = 0
for i in range(2,n):
    if(isPrime(i)):
        for j in range(len(li)):
            for k in range(j+1,len(li)):
                if i == li[j] + li[k]:
                    count = count + 1
```

```
                    ans = i
                    # print(i,li[j],li[k])
        for j in range(len(li)):
            for k in range(j+1,len(li)):
                if i == li[k] - li[j]:
                    count = count + 1
                    ans = i
                    # print(i,li[j],li[k])
        if count == 2: print(i)
        else: count = 0
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-3.py
Enter a number: 45
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43]
5
PS C:\Users\vidit>
```

D. Find three least positive integers n such that there are no primes between n and n + 10, and three least positive integers m such that there are no primes between 10m and 10(m+1)

```
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []
for i in range(2,n):
    if not any(isPrime(j) for j in range(i,i+11)):
            print(i)
            li.append(i)
            if(len(li) == 3):
                break
    if(len(li) == 3):
        break
lm = []
for i in range(2,n):
    if not any(isPrime(j) for j in range(10*i,10*i + 11)):
            print(i)
            lm.append(i)
            if(len(lm) == 3):
                break
    if(len(lm) == 3):
        break
```

```
print(li)
print(lm)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-4.py
Enter a number: 400
114
115
116
20
32
51
[114, 115, 116]
[20, 32, 51]
PS C:\Users\vidit>
```

E. Find four solutions of the equation p^2+ 1 = q^2+r^2 primes

```
# p^2 + 1 = q^2 + r^2    p,q,r are primes
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

count = 0
li = []
for i in range(2,n):
    if(isPrime(i)):
        for j in range(2,n):
            if(isPrime(j)):
                for k in range(2,n):
                    if(isPrime(k)):
                        if i*i + 1 == j*j + k*k:
                            count = count+1
                            # print(i,j,k)
                            li.append((i,j,k))
                    if(count==6): break
            if(count==6): break
    if(count==6): break
print(li)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-5.py
Enter a number: 100
[(7, 5, 5), (13, 7, 11), (13, 11, 7), (17, 11, 13), (17, 13, 11), (23, 13, 19)]
PS C:\Users\vidit>
```

F. Find all prime solutions of the equation p(p+1)+q(q+1) = r(r+1)

```
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

count = 0
li = []
for i in range(2,n):
    if(isPrime(i)):
        for j in range(2,n):
            if(isPrime(j)):
                for k in range(2,n):
                    if(isPrime(k)):
                        if i*(i+1) + j*(j+1) == k*(k+1):
                            li.append((i,j,k))
                            break
print(li)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-6.py
Enter a number: 300
[(2, 2, 3)]
PS C:\Users\vidit>
```

G. Find all primes numbers p(p+1), q(q+1), r(r+1) form an AP

```
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []

for i in range(2,n):
    if(isPrime(i)):
        for j in range(2,n):
            if(i==j): continue
            if(isPrime(j)):
                for k in range(2,n):
                    if(i==k or j==k): continue
                    if(isPrime(k)):
```

```
                        if (j*(j+1) - i*(i+1)) == (k*(k+1) - j*(j+1)):
                            li.append((i,j,k))
                            break
print(li)
print(len(li))
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-7.py
Enter a number: 600
[(127, 397, 547), (547, 397, 127)]
2
PS C:\Users\vidit>
```

H. Find integers n where n+1, n+3, n+7, n+9, n+13, and n+15 = prime

```
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []
for i in range(2,n):
    if(isPrime(i+1)):
        if(isPrime(i+3)):
            if(isPrime(i+7)):
                if(isPrime(i+9)):
                    if(isPrime(i+13)):
                        if(isPrime(i+15)):
                            li.append(i)
                            print(i)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-8.py
Enter a number: 300
4
PS C:\Users\vidit>
```

I. Find five primes that are sums of two fourth powers of integers

```
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []
for i in range(2,n):
    if(isPrime(i)):
```

```
        li.append(i)
count = 0
for i in range(1,n):
    for j in range(i,n):
        ans = pow(i,4)+pow(j,4)
        for k in range(len(li)):
            if li[k] == ans:
                count= count + 1
                print(ans, "i & j are", i, j)
            if(count == 5): break
        if(count == 5): break
    if(count == 5): break
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-9.py
Enter a number: 100
2 i & j are 1 1
17 i & j are 1 2
97 i & j are 2 3
PS C:\Users\vidit>
```

J. Prove that there are infinite pairs of consecutive primes that are not twin primes

```
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []
for i in range(2,n):
    if(isPrime(i)):
        li.append(i)
ans = []
for i in range(len(li)-1):
    for j in range(i+1,len(li)):
        if(li[j] - li[i] !=2):
            ans.append((li[i],li[j]))
print(ans)
print(len(ans))
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-10.py
Enter a number: 100
[(2, 3), (2, 5), (2, 7), (2, 11), (2, 13), (2, 17), (2, 19), (2, 23), (2, 29), (2, 31), (2, 37), (2, 41), (2
9), (2, 97), (3, 7), (3, 11), (3, 13), (3, 17), (3, 19), (3, 23), (3, 29), (3, 31), (3, 37), (3, 41), (3, 43
(3, 97), (5, 11), (5, 13), (5, 17), (5, 19), (5, 23), (5, 29), (5, 31), (5, 37), (5, 41), (5, 43), (5, 47),
 11), (7, 13), (7, 17), (7, 19), (7, 23), (7, 29), (7, 31), (7, 37), (7, 41), (7, 43), (7, 47), (7, 53), (7,
19), (11, 23), (11, 29), (11, 31), (11, 37), (11, 41), (11, 43), (11, 47), (11, 53), (11, 59), (11, 61), (11
3), (13, 29), (13, 31), (13, 37), (13, 41), (13, 43), (13, 47), (13, 53), (13, 59), (13, 61), (13, 67), (13,
), (17, 41), (17, 43), (17, 47), (17, 53), (17, 59), (17, 61), (17, 67), (17, 71), (17, 73), (17, 79), (17,
, (19, 53), (19, 59), (19, 61), (19, 67), (19, 71), (19, 73), (19, 79), (19, 83), (19, 89), (19, 97), (23, 2
 (23, 71), (23, 73), (23, 79), (23, 83), (23, 89), (23, 97), (29, 37), (29, 41), (29, 43), (29, 47), (29, 53
(31, 37), (31, 41), (31, 43), (31, 47), (31, 53), (31, 59), (31, 61), (31, 67), (31, 71), (31, 73), (31, 79)
37, 67), (37, 71), (37, 73), (37, 79), (37, 83), (37, 89), (37, 97), (41, 47), (41, 53), (41, 59), (41, 61),
3, 59), (43, 61), (43, 67), (43, 71), (43, 73), (43, 79), (43, 83), (43, 89), (43, 97), (47, 53), (47, 59),
, 61), (53, 67), (53, 71), (53, 73), (53, 79), (53, 83), (53, 89), (53, 97), (59, 67), (59, 71), (59, 73), (
 89), (61, 97), (67, 71), (67, 73), (67, 79), (67, 83), (67, 89), (67, 97), (71, 79), (71, 83), (71, 89), (7
97), (89, 97)]
292
```

K. Find 5 integers for n^2-1 is a product of three different primes.

```
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []
for i in range(2,n):
    if(isPrime(i)):
        li.append(i)
print(li)

ans = []
count = 0
for i in range(len(li)):
    for j in range(i+1,len(li)):
        for k in range(j+1,len(li)):
            for z in range(1,n):
                if z*z - 1 == li[i]*li[j]*li[k]:
                    count = count + 1
                    print(z, " - ", li[i],li[j],li[k])
                    ans.append(z)
                if(count == 5): break
            if(count == 5): break
        if(count == 5): break
    if(count == 5): break

print(ans)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-11.py
Enter a number: 100
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
14  -  3 5 13
16  -  3 5 17
20  -  3 7 19
22  -  3 7 23
32  -  3 11 31
[14, 16, 20, 22, 32]
PS C:\Users\vidit>
```

L. Find five least positive integers n for which n^2+1 is a product of three different primes, and find a positive integer n for which n^2+1 is a product of three different odd primes.

```
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []
for i in range(2,n):
    if(isPrime(i)):
        li.append(i)
print(li)

ans = []
count = 0
for i in range(len(li)):
    for j in range(i+1,len(li)):
        for k in range(j+1,len(li)):
            for z in range(1,n):
                if z*z + 1 == li[i]*li[j]*li[k]:
                    count = count + 1
                    print(z, " - ", li[i],li[j],li[k])
                    ans.append(z)
                if(count == 5): break
            if(count == 5): break
        if(count == 5): break
    if(count == 5): break

print(ans)
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
```

```
        if n%i == 0: return False
    return True

li = []
for i in range(2,n):
    if(isPrime(i)):
        if(i%2!=0):
            li.append(i)
print(li)

ans = []
count = 0
for i in range(len(li)):
    for j in range(i+1,len(li)):
        for k in range(j+1,len(li)):
            for z in range(1,n):
                if z*z + 1 == li[i]*li[j]*li[k]:
                    count = count + 1
                    print(z, " - ", li[i],li[j],li[k])
                    ans.append(z)
                if(count == 5): break
            if(count == 5): break
        if(count == 5): break
    if(count == 5): break

print(ans)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe "c:/Users/vidit/Downloads/exp4/exp4/exp4-12(a).py"
Enter a number: 100
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
13  -  2 5 17
17  -  2 5 29
23  -  2 5 53
27  -  2 5 73
21  -  2 13 17
[13, 17, 23, 27, 21]
PS C:\Users\vidit>
```

```
PS C:\Users\vidit> & C:/Python312/python.exe "c:/Users/vidit/Downloads/exp4/exp4/exp4-12(b).py"
Enter a number: 100
[3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
72  -  5 17 61
[72]
PS C:\Users\vidit>
```

M. Find five least positive integers n such that each of the numbers n, n+1, n+2 is a product of two different primes

```
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []
for i in range(2,n):
    if(isPrime(i)):
        li.append(i)
ans = []
for i in range(len(li)):
    for j in range(i+1,len(li)):
            for z in range(2,n):
                if z == li[i]*li[j]:
                    ans.append(z)

ans.sort()
print(ans)
count = 0
res = []
for i in range(len(ans)-2):
    if ans[i] + 2 == ans[i+1] + 1 == ans[i+2]:
        res.append(ans[i])
result = []
for i in range(len(ans)-3):
    if ans[i] + 3 == ans[i+1] + 3 == ans[i+2] + 1 == ans[i+3]:
        result.append(ans[i])

print(res[0: 5])
print("PART B ====================================")
print(result)
print("PART C ====================================")
con = 0
checker = False
for i in range(len(ans)):
    for j in range(len(li)):
        for k in range(j+1,len(li)):
            if ans[i] == li[j]*li[k]:
```

```
                con = con + 1
                print("Number ", ans[i], "has only 2 primes i.e ", li[j],
li[k])
            if con==4:
                checker = True
                break
        if(checker): break
    if(checker): break
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-13.py
Enter a number: 300
[6, 10, 14, 15, 21, 22, 26, 33, 34, 35, 38, 39, 46, 51, 55, 57, 58, 62, 65, 69, 74, 77, 82, 85, 86, 87,
58, 159, 161, 166, 177, 178, 183, 185, 187, 194, 201, 202, 203, 205, 206, 209, 213, 214, 215, 217, 218,
9]
[33, 85, 93, 141, 201]
PART B ====================================
[]
PART C ====================================
Number  6 has only 2 primes i.e  2 3
Number  10 has only 2 primes i.e  2 5
Number  14 has only 2 primes i.e  2 7
Number  15 has only 2 primes i.e  3 5
PS C:\Users\vidit>
```

N. Find all numbers of the form 2^n-1 with positive integer n, not exceeding million, which are products of two primes

```
n = int(input("Enter a number: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []
for i in range(2,n):
    if(isPrime(i)):
        li.append(i)
ans = []
for i in range(len(li)):
    for j in range(i+1,len(li)):
            for z in range(1,n):
                if pow(2,z)-1 == li[i]*li[j]:
                    print(z,"(",pow(2,z),")", " - ", li[i],li[j])
                    ans.append(z)

print("PART B ============================")
```

```
for i in range(len(li)):
    for j in range(i+1,len(li)):
        for k in range(j+1,len(li)):
            for z in range(1,n):
                if z%2 == 0 and z>4:
                    if pow(2,z)-1 == li[i]*li[j]*li[k]:
                        print(z, " - ", li[i],li[j],li[k])
                        ans.append(z)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp4/exp4/exp4-14.py
Enter a number: 300
4 ( 16 )  -  3 5
9 ( 512 )  -  7 73
11 ( 2048 )  -  23 89
PART B ==========================
8  -  3 5 17
10  -  3 11 31
14  -  3 43 127
PS C:\Users\vidit>
```

EXPERIMENT 6

A. Finding twin primes for a given n

```
n = int(input("Enter the value of N: "))

def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []
for i in range(2,n):
    if(isPrime(i)):
        li.append(i)

ans = []

for i in range(len(li)):
    for j in range(i+1, len(li)):
        if(li[j] - li[i] == 2):
            ans.append({li[i], li[j]})
        if(li[j] - li[i] > 2):
            break

print(ans)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp5/exp5/exp5-1.py
Enter the value of N: 100
[{3, 5}, {5, 7}, {11, 13}, {17, 19}, {29, 31}, {41, 43}, {59, 61}, {73, 71}]
PS C:\Users\vidit>
```

B. Finding primes for a given n of form i^2 + 1 = prime

```
m = int(input("Enter the value of m: "))
def isPrime(n):
    for i in range(2,n):
        if n%i == 0: return False
    return True

li = []
for i in range(2,2000):
    if(isPrime(i)):
```

```
        li.append(i)

ans = []
count = 0
for i in li:
    for j in range(1, 2000):
        if j*j + 1 == i:
            count = count + 1
            ans.append((j, i))
            if(count == m): break
    if(count == m): break

print(ans)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp5/exp5/exp5-2.py
Enter the value of m: 100
[(1, 2), (2, 5), (4, 17), (6, 37), (10, 101), (14, 197), (16, 257), (20, 401), (24, 577), (26, 677), (36, 1297), (40, 1601)]
PS C:\Users\vidit>
```

C. Smallest prime factor of n! + 1

```
ans = []
for j in range(1, 20):
    def fact(n):
        if n == 0: return 1
        ans = n*fact(n-1)
        return ans

    val = fact(j) + 1
    # print(val)

    def isPrime(n):
        for i in range(2,n):
            if n%i == 0: return False
        return True

    li = []
    for i in range(2,2000):
        if(isPrime(i)):
            li.append(i)

    for i in li:
        if val%i == 0:
            print("Smallest prime factor for the value of n as ",j, "is: ", i)
            ans.append(i)
```

```
            break
print(ans)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp5/exp5/exp5-3.py
Smallest prime factor for the value of n as  1 is:  2
Smallest prime factor for the value of n as  2 is:  3
Smallest prime factor for the value of n as  3 is:  7
Smallest prime factor for the value of n as  4 is:  5
Smallest prime factor for the value of n as  5 is:  11
Smallest prime factor for the value of n as  6 is:  7
Smallest prime factor for the value of n as  7 is:  71
Smallest prime factor for the value of n as  8 is:  61
Smallest prime factor for the value of n as  9 is:  19
Smallest prime factor for the value of n as  10 is:  11
Smallest prime factor for the value of n as  12 is:  13
Smallest prime factor for the value of n as  13 is:  83
Smallest prime factor for the value of n as  14 is:  23
Smallest prime factor for the value of n as  15 is:  59
Smallest prime factor for the value of n as  16 is:  17
Smallest prime factor for the value of n as  17 is:  661
Smallest prime factor for the value of n as  18 is:  19
Smallest prime factor for the value of n as  19 is:  71
[2, 3, 7, 5, 11, 7, 71, 61, 19, 11, 13, 83, 23, 59, 17, 661, 19, 71]
PS C:\Users\vidit>
```

D. Determine whether a number is prime or not

```
n = int(input("Enter the value of n: "))
no = int(pow(n, 0.5))
# print(no)
def isPrime(n, no):
    for i in range(2,no):
        if n%i == 0: return False
    return True
check = isPrime(n, no)
print(check)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp5/exp5/exp5-4.py
Enter the value of n: 200
False
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp5/exp5/exp5-4.py
Enter the value of n: 101
True
PS C:\Users\vidit>
```

E. Euclidean algorithm to find gcd

```
def gcd(n1, n2):
    if n2 == 0:
        return n1
    else:
        return gcd(n2, n1%n2)
```

```
a = int(input("Enter the 1st no: "))
b = int(input("Enter the 1st no: "))

d = gcd(a, b)
print("GCD OF 2 NOS = ", d)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp5/exp5/exp5-5.py
Enter the 1st no: 43
Enter the 1st no: 12
GCD OF 2 NOS =  1
PS C:\Users\vidit>
```

F. Set of 2 or more numbers, find their gcd

```
a = []
n = int(input("Enter number of values to enter: "))
print("Enter the values: ")

for i in range(n):
    b = int(input())
    a.append(b)

def gcd(n1, n2):
    if n2 == 0:
        return n1
    else:
        return gcd(n2, n1%n2)

d = gcd(a[0], a[1])
for i in range(2, n):
    d = gcd(d, a[i])
print("GCD IS ==========================")
print(d)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/exp5/exp5/exp5-6.py
Enter number of values to enter: 5
Enter the values:
10
23
12
11
16
GCD IS ==========================
1
PS C:\Users\vidit>
```

EXPERIMENT 7

A. Ceaser cipher

```
text = input("Enter the plaintext or message: ")

arr = {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8,
'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r':
17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25}

k = 3
cipher = ""
for i in range(len(text)):
    value = arr[text[i]]
    a = (value + k)%26
    for key, value in arr.items():
        if value == a:
            b = key
            cipher = cipher + b

print("Encryption--------------------------")
print(cipher)

plaintext = ""
for i in range(len(cipher)):
    value = arr[cipher[i]]
    a = (value - k)%26
    for key, value in arr.items():
        if value == a:
            b = key
            plaintext = plaintext + b
print("Decryption--------------------------")
print(plaintext)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/Exp6-1.py
Enter the plaintext or message: thisissecret
Encryption--------------------------
wklvlvvhfuhw
Decryption--------------------------
thisissecret
PS C:\Users\vidit>
```

B. Monoalphabetic cipher - affine cipher

```
arr = {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8,
'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r':
17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25}

k1 = int(input("Enter the key 1 that is prime to 26: "))
k2 = int(input("Enter the key 2: "))

text = input("Enter the plaintext or message: ")

cipher = ""
for i in range(len(text)):
    value = arr[text[i]]
    a = (value*k1 + k2)%26
    for key, value in arr.items():
        if value == a:
            b = key
            cipher = cipher + b

print("Encryption--------------------------")
print(cipher)
c = pow(k1, -1, 26)
plaintext = ""
for i in range(len(cipher)):
    value = arr[cipher[i]]
    a = ((value-k2)*c)%26
    for key, value in arr.items():
        if value == a:
            b = key
            plaintext = plaintext + b

print("Decryption--------------------------")
print(plaintext)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/Exp6-2.py
Enter the key 1 that is prime to 26: 5
Enter the key 2: 19
Enter the plaintext or message: tommeetingattwo
Encryption--------------------------
klbbnnkhgxtkkzl
Decryption--------------------------
tommeetingattwo
PS C:\Users\vidit>
```

C. Playfair cipher

```
text = input("Enter the plaintext or message: ")
key = input("Enter the key: ")

arr = ['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 generate_key_matrix(key):
    key = key.upper().replace("J", "I")
    matrix = []
    used = set()

    for ch in key:
        if ch not in used and ch.isalpha():
            used.add(ch)
            matrix.append(ch)

    for ch in "ABCDEFGHIKLMNOPQRSTUVWXYZ":
        if ch not in used:
            used.add(ch)
            matrix.append(ch)

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


def find_position(matrix, ch):
    for i, row in enumerate(matrix):
        for j, val in enumerate(row):
            if val == ch:
                return i, j
    return None


def prepare_text(text, for_encryption=True):
    text = text.upper().replace("J", "I")
    result = ""
    i = 0

    while i < len(text):
        ch1 = text[i]
        if not ch1.isalpha():
```

```
            i += 1
            continue
        if i + 1 < len(text):
            ch2 = text[i+1]
            if not ch2.isalpha():
                i += 1
                continue
            if ch1 == ch2:
                result += ch1 + 'X'
                i += 1
            else:
                result += ch1 + ch2
                i += 2
        else:
            result += ch1 + 'X'
            i += 1
    return result


def playfair_encrypt(text, matrix):
    text = prepare_text(text)
    cipher = ""

    for i in range(0, len(text), 2):
        a, b = text[i], text[i+1]
        r1, c1 = find_position(matrix, a)
        r2, c2 = find_position(matrix, b)

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

    return cipher

def playfair_decrypt(cipher, matrix):
    plain = ""
```

```
    for i in range(0, len(cipher), 2):
        a, b = cipher[i], cipher[i+1]
        r1, c1 = find_position(matrix, a)
        r2, c2 = find_position(matrix, b)

        if r1 == r2:
            plain += matrix[r1][(c1 - 1) % 5]
            plain += matrix[r2][(c2 - 1) % 5]
        elif c1 == c2:
            plain += matrix[(r1 - 1) % 5][c1]
            plain += matrix[(r2 - 1) % 5][c2]
        else:
            plain += matrix[r1][c2]
            plain += matrix[r2][c1]

    return plain

matrix = generate_key_matrix(key)
print("Key Matrix:")
for row in matrix:
    print(row)

print("Encryption--------------------------")
ciphertext = playfair_encrypt(text, matrix)
print("\nEncrypted:", ciphertext)

print("Decryption--------------------------")
decrypted = playfair_decrypt(ciphertext, matrix)
print("Decrypted:", decrypted)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/Exp6-3.py
Enter the plaintext or message: message
Enter the key: monarchy
Key Matrix:
['M', 'O', 'N', 'A', 'R']
['C', 'H', 'Y', 'B', 'D']
['E', 'F', 'G', 'I', 'K']
['L', 'P', 'Q', 'S', 'T']
['U', 'V', 'W', 'X', 'Z']
Encryption--------------------------

Encrypted: CLXAXBIF
Decryption--------------------------
Decrypted: MESXSAGE
PS C:\Users\vidit>
```

D. Hill cipher

```
import numpy as np

plaintext = (input("Enter the text : "))

def mod_inverse(a, m):
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    return None

def encrypt_hill_cipher(plaintext, key_matrix):
    n = key_matrix.shape[0]
    if len(plaintext) % n != 0:
        plaintext += 'X' * (n - (len(plaintext) % n))

    ciphertext = ""
    for i in range(0, len(plaintext), n):
        block = plaintext[i : i + n]
        message_vector = np.array([ord(char) - ord('A') for char in
block]).reshape(n, 1)

        cipher_vector = np.dot(key_matrix, message_vector) % 26

        ciphertext += "".join(chr(int(val) + ord('A')) for val in
cipher_vector)
    return ciphertext


def decrypt_hill_cipher(ciphertext, key_matrix):
    n = key_matrix.shape[0]
    det = int(round(np.linalg.det(key_matrix)))
    det_mod_26 = det % 26
    det_inv = mod_inverse(det_mod_26, 26)

    if det_inv is None:
        print("Key matrix is not invertible modulo 26. Cannot decrypt.")

    adjugate_matrix = np.round(det * np.linalg.inv(key_matrix)).astype(int) %
26
    inverse_key_matrix = (det_inv * adjugate_matrix) % 26
```

```
    decrypted_text = ""

    for i in range(0, len(ciphertext), n):
        block = ciphertext[i : i + n]
        cipher_vector = np.array([ord(char) - ord('A') for char in
block]).reshape(n, 1)

        message_vector = np.dot(inverse_key_matrix, cipher_vector) % 26

        decrypted_text += "".join(chr(int(val) + ord('A')) for val in
message_vector)
    return decrypted_text

key_matrix_2x2 = np.array([[3, 5], [4, 7]])

encrypted_text = encrypt_hill_cipher(plaintext.upper(), key_matrix_2x2)
print(f"Plaintext: {plaintext}")
print("Encryption--------------------------")
print(f"Encrypted Text: {encrypted_text}")

print("Decryption--------------------------")
decrypted_text = decrypt_hill_cipher(encrypted_text, key_matrix_2x2)
print(f"Decrypted Text: {decrypted_text}")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/exp6-4.py
Enter the text : secret
c:\Users\vidit\Downloads\Cryptoexp6\exp6\exp6-4.py:25: DeprecationWarning: Conversion of an array
from your array before performing this operation. (Deprecated NumPy 1.25.)
  ciphertext += "".join(chr(int(val) + ord('A')) for val in cipher_vector)
Plaintext: secret
Encryption--------------------------
Encrypted Text: WWNXDT
Decryption--------------------------
c:\Users\vidit\Downloads\Cryptoexp6\exp6\exp6-4.py:49: DeprecationWarning: Conversion of an array
from your array before performing this operation. (Deprecated NumPy 1.25.)
  decrypted_text += "".join(chr(int(val) + ord('A')) for val in message_vector)
Decrypted Text: SECRET
PS C:\Users\vidit>
```

E. Vignere cipher

```
text = (input("Enter the text : "))
k = (input("Enter the key : "))

arr = {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8,
'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r':
17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25}
```

```
while(len(k)<len(text)):
    if(len(k)!=len(text)):
        k = k + k

cipher = ""

for i in range(len(text)):
    value1 = arr[text[i]]
    value2 = arr[k[i]]
    a = (value1 + value2)%26
    for key, value in arr.items():
        if value == a:
            b = key
            cipher = cipher + b
print("Encoding--------------------")
print(cipher)

plaintext = ""
for i in range(len(cipher)):
    value1 = arr[cipher[i]]
    value2 = arr[k[i]]
    a = (value1 - value2)%26
    for key, value in arr.items():
        if value == a:
            b = key
            plaintext = plaintext + b
print("Decoding--------------------")
print(plaintext)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/exp6-5.py
Enter the text : veryimportantmessage
Enter the key : guyswelcome
Encoding--------------------
bypqeqaqffetnkwowlis
Decoding--------------------
veryimportantmessage
PS C:\Users\vidit>
```

F. Autokey cipher

```
text = (input("Enter the text : "))
k = (input("Enter the key of size 1: "))
```

```
arr = {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8,
'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r':
17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25}

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

cipher = ""

for i in range(len(text)):
    value1 = arr[text[i]]
    value2 = arr[k[i]]
    a = (value1 + value2)%26
    for key, value in arr.items():
        if value == a:
            b = key
            cipher = cipher + b
print("Encoding------------------")
print(cipher)

plaintext = ""
for i in range(len(cipher)):
    value1 = arr[cipher[i]]
    value2 = arr[k[i]]
    a = (value1 - value2)%26
    for key, value in arr.items():
        if value == a:
            b = key
            plaintext = plaintext + b
print("Decoding------------------")
print(plaintext)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/Exp6-6.py
Enter the text : helloguys
Enter the key of size 1: j
Encoding------------------
qlpwzuasq
Decoding------------------
helloguys
```

G. Vernam cipher

```
text = (input("Enter the text : "))
k = (input("Enter the key of same size: "))

arr = {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8,
'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r':
17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25}


cipher = ""
for i in range(len(text)):
    value1 = arr[text[i]]
    value2 = arr[k[i]]
    a = (value1^value2)%26
    for key, value in arr.items():
        if value == a:
            b = key
            cipher = cipher + b

print("Encoding------------------")
print(cipher)
plaintext = ""
for i in range(len(cipher)):
    value1 = arr[cipher[i]]
    value2 = arr[k[i]]
    a = (value1^value2)%26
    for key, value in arr.items():
        if value == a:
            b = key
            plaintext = plaintext + b

print("Decoding------------------")
print(plaintext)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/exp6-7.py
Enter the text : mine
Enter the key of same size: four
Encoding------------------
jgzv
Decoding------------------
mine
PS C:\Users\vidit>
```

H. One time pad cipher

```
text = (input("Enter the text : "))
k = (input("Enter the key of the same size : "))

arr = {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8,
'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r':
17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25}

cipher = ""

for i in range(len(text)):
    value1 = arr[text[i]]
    value2 = arr[k[i]]
    a = (value1 + value2)%26
    for key, value in arr.items():
        if value == a:
            b = key
            cipher = cipher + b
print("Encoding--------------------")
print(cipher)

plaintext = ""
for i in range(len(cipher)):
    value1 = arr[cipher[i]]
    value2 = arr[k[i]]
    a = (value1 - value2)%26
    for key, value in arr.items():
        if value == a:
            b = key
            plaintext = plaintext + b
print("Decoding--------------------")
print(plaintext)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/exp6-8.py
Enter the text : helloworld
Enter the key of the same size : moneymoney
Encoding--------------------
tsypmicepb
Decoding--------------------
helloworld
PS C:\Users\vidit>
```

I. Kasiksi method

```
def vignere(plaintext, key):
    ciphertext = ""
    key_index = 0
    for char in plaintext:
        if 'A' <= char <= 'Z':
            shift = ord(key[key_index % len(key)].upper()) - ord('A')
            encrypted_char = chr(((ord(char) - ord('A') + shift) % 26) +
ord('A'))
            ciphertext += encrypted_char
            key_index += 1
        elif 'a' <= char <= 'z':
            shift = ord(key[key_index % len(key)].lower()) - ord('a')
            encrypted_char = chr(((ord(char) - ord('a') + shift) % 26) +
ord('a'))
            ciphertext += encrypted_char
            key_index += 1
        else:
            ciphertext += char
    return ciphertext



def find_repeated_sequences(ciphertext, min_length=3):
    repetitions = {}
    for i in range(len(ciphertext) - min_length + 1):
        sequence = ciphertext[i:i+min_length]
        for j in range(i + min_length, len(ciphertext) - min_length + 1):
            if ciphertext[j:j+min_length] == sequence:
                if sequence not in repetitions:
                    repetitions[sequence] = []
                repetitions[sequence].append(j - i)
    return repetitions


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

```
def estimate_key_length_kasiski(ciphertext, min_sequence_length=3):
    repetitions = find_repeated_sequences(ciphertext, min_sequence_length)

    distances = []
    for seq, dists in repetitions.items():
        distances.extend(dists)

    if not distances:
        return None

    current_gcd = distances[0]
    for i in range(1, len(distances)):
        current_gcd = calculate_gcd(current_gcd, distances[i])

    return current_gcd


long_plaintext = "THISISAMUCHLONGERTEXTTOILLUSTRATETHEKASISKIMETHODBETTER"
long_plaintext.lower()
long_key = "SECRET"
long_key.lower()
long_ciphertext = vignere(long_plaintext, long_key)
print(f"\nLong Ciphertext for Kasiski: {long_ciphertext}")

estimated_key_length = estimate_key_length_kasiski(long_ciphertext)
print(f"Estimated Key Length (Kasiski Method): {estimated_key_length}")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/Exp6-11.py

Long Ciphertext for Kasiski: LLKJMLSQWTLEGRIVVMWBVKSBDPWJXKSXGKLXCEUZWDAQGKLHVFGKXXJ
Estimated Key Length (Kasiski Method): 12
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/Exp6-9.py
```

J. Rail fence cipher

```
text = (input("Enter the text : "))

t1 = ""
t2 = ""
for i in range(len(text)):
    if i%2==0:
        t1 = t1 + text[i]
    else:
        t2 = t2 + text[i]
```

```
cipher = t1 + t2
print("Encoding-------------------")
print(cipher)
plaintext = ""
n = len(cipher)
n1 = ""
n2 = ""
if(n%2 !=0):
    for i in range(n):
        if(i >= int(n/2 + 1)):
            n2 = n2 + cipher[i]
        else:
            n1 = n1 + cipher[i]

    j = 0
    k = 0
    for i in range(n):
        if(i%2 == 0):
            plaintext = plaintext + n1[j]
            j = j+1
        else:
            plaintext = plaintext + n2[k]
            k = k + 1
    print("Decoding-------------------")
    print(plaintext)
else:
    for i in range(n):
        if(i >= int(n/2)):
            n2 = n2 + cipher[i]
        else:
            n1 = n1 + cipher[i]

    j = 0
    k = 0
    for i in range(n):
        if(i%2 == 0):
            plaintext = plaintext + n1[j]
            j = j+1
        else:
            plaintext = plaintext + n2[k]
            k = k + 1
```

```
    print("Decoding-------------------")
    print(plaintext)
```

```
Enter the text : messagemetom
Encoding-------------------
msaeeoesgmtm
Decoding-------------------
messagemetom
PS C:\Users\vidit>
```

K. Row column transposition cipher

```
text = (input("Enter the text : "))
k = (input("Enter the key as per no.of columns required: "))

arr = {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8,
'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r':
17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25}

a = []
for i in range(len(k)):
    temp = []
    temp.append(arr[k[i]])
    temp.append(i)
    a.append(temp)
print(a)
count = 0
cip = []
while(True):
    if count >= len(text): break
    temp = []
    for i in range(len(k)):
        if count >= len(text):
            count = count + 1
            temp.append('x')
        else:
            temp.append(text[count])
            count = count + 1
    cip.append(temp)

print(cip)
```

```
a.sort()
print(a)

cipher = ""

count = 0
k = 0
while(True):
    if(count == len(cip)*len(cip[0])): break
    for i in range(len(cip)):
        cipher = cipher + cip[i][a[k][1]]
        count = count + 1
    k = k+1
print("Encryption----------------------")
print(cipher)


plain = []
for i in range(int(len(cipher)/len(a))):
    temp = []
    for j in range(len(a)):
        temp.append(0)
    plain.append(temp)

plaintext = ""
count = 0
k = 0
for i in range(len(a)):
    m = a[i][1]
    for j in range(int(len(cipher)/len(a))):
        plain[j][m] = cipher[count]
        count = count + 1


print("Decryption----------------------")
print(plain)
for i in range(len(plain)):
    for j in range(len(plain[0])):
        plaintext = plaintext + plain[i][j]
print(plaintext)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/Exp6-10.py
Enter the text : messagemetodayasap
Enter the key as per no.of columns required: world
[[22, 0], [14, 1], [17, 2], [11, 3], [3, 4]]
[['m', 'e', 's', 's', 'a'], ['g', 'e', 'm', 'e', 't'], ['o', 'd', 'a', 'y', 'a'], ['s', 'a', 'p', 'x', 'x']]
[[3, 4], [11, 3], [14, 1], [17, 2], [22, 0]]
Encryption----------------------
ataxseyxeedasmapmgos
Decryption----------------------
[['m', 'e', 's', 's', 'a'], ['g', 'e', 'm', 'e', 't'], ['o', 'd', 'a', 'y', 'a'], ['s', 'a', 'p', 'x', 'x']]
messagemetodayasapxx
PS C:\Users\vidit>
```

L. 2 substitutions cipher

```
text = (input("Enter the text : "))
k2 = (input("Enter the key of same size: "))
arr = {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8,
'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r':
17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25}

k1 = 3
cipher = ""
for i in range(len(text)):
    value = arr[text[i]]
    a = (value + k1)%26
    for key, value in arr.items():
        if value == a:
            b = key
            cipher = cipher + b

print("Encryption---------------------------")
print(cipher)

cipher2 = ""
for i in range(len(cipher)):
    value1 = arr[cipher[i]]
    value2 = arr[k2[i]]
    a = (value1^value2)%26
    for key, value in arr.items():
        if value == a:
            b = key
            cipher2 = cipher2 + b

print("Encoding-----------------")
print(cipher2)
```

```
plaintext = ""
for i in range(len(cipher2)):
    value1 = arr[cipher2[i]]
    value2 = arr[k2[i]]
    a = (value1^value2)%26
    for key, value in arr.items():
        if value == a:
            b = key
            plaintext = plaintext + b

print("Decoding------------------")
print(plaintext)



plaintext1 = ""
for i in range(len(plaintext)):
    value = arr[plaintext[i]]
    a = (value - k1)%26
    for key, value in arr.items():
        if value == a:
            b = key
            plaintext1 = plaintext1 + b
print("Decryption--------------------------")
print(plaintext1)
```

```
  rerlo
  PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/Exp6-12.py
  Enter the text : mine
  Enter the key of same size: four
  Encryption--------------------------
  plqh
  Encoding------------------
  kfew
  Decoding------------------
  plqh
  Decryption--------------------------
  mine
  PS C:\Users\vidit>
```

M. 2 transpositions cipher

```
arr = {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8,
'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r':
17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25}

text = (input("Enter the text : "))
```

```
t1 = ""
t2 = ""
for i in range(len(text)):
    if i%2==0:
        t1 = t1 + text[i]
    else:
        t2 = t2 + text[i]


cipher = t1 + t2
print("Encoding-------------------")
print(cipher)


k = (input("Enter the key as per no.of columns required: "))

a = []
for i in range(len(k)):
    temp = []
    temp.append(arr[k[i]])
    temp.append(i)
    a.append(temp)
print(a)
count = 0
cip = []
while(True):
    if count >= len(cipher): break
    temp = []
    for i in range(len(k)):
        if count >= len(cipher):
            count = count + 1
            temp.append('x')
        else:
            temp.append(cipher[count])
            count = count + 1
    cip.append(temp)

print(cip)
a.sort()
print(a)
```

```
cipher2 = ""

count = 0
k = 0
while(True):
    if(count == len(cip)*len(cip[0])): break
    for i in range(len(cip)):
        cipher2 = cipher2 + cip[i][a[k][1]]
        count = count + 1
    k = k+1
print("Encryption-----------------------")
print(cipher2)
plain = []
for i in range(int(len(cipher2)/len(a))):
    temp = []
    for j in range(len(a)):
        temp.append(0)
    plain.append(temp)

plaintext = ""
count = 0
k = 0
for i in range(len(a)):
    m = a[i][1]
    for j in range(int(len(cipher2)/len(a))):
        plain[j][m] = cipher2[count]
        count = count + 1

print("Decryption-----------------------")
print(plain)
for i in range(len(plain)):
    for j in range(len(plain[0])):
        plaintext = plaintext + plain[i][j]
print(plaintext)

plaintext1 = ""
n = len(plaintext)
n1 = ""
n2 = ""
if(n%2 !=0):
    for i in range(n):
        if(i >= int(n/2 + 1)):
```

```
            n2 = n2 + plaintext[i]
        else:
            n1 = n1 + plaintext[i]

    j = 0
    k = 0
    for i in range(n):
        if(i%2 == 0):
            plaintext1 = plaintext1 + n1[j]
            j = j+1
        else:
            plaintext1 = plaintext1 + n2[k]
            k = k + 1
    print("Decoding-------------------")
    print(plaintext)
else:
    for i in range(n):
        if(i >= int(n/2)):
            n2 = n2 + plaintext[i]
        else:
            n1 = n1 + plaintext[i]

    j = 0
    k = 0
    for i in range(n):
        if(i%2 == 0):
            plaintext1 = plaintext1 + n1[j]
            j = j+1
        else:
            plaintext1 = plaintext1 + n2[k]
            k = k + 1
    print("Decoding-------------------")
    print(plaintext1)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/Exp6-13.py
Enter the text : meet
Encoding------------------
meet
Enter the key as per no.of columns required: fine
[[5, 0], [8, 1], [13, 2], [4, 3]]
[['m', 'e', 'e', 't']]
[[4, 3], [5, 0], [8, 1], [13, 2]]
Encryption----------------------
tmee
Decryption----------------------
[['m', 'e', 'e', 't']]
meet
Decoding------------------
meet
PS C:\Users\vidit>
```

N. substitution + transposition cipher

```
text = (input("Enter the text : "))
arr = {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8,
'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r':
17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25}

k = 3
cipher = ""
for i in range(len(text)):
    value = arr[text[i]]
    a = (value + k)%26
    for key, value in arr.items():
        if value == a:
            b = key
            cipher = cipher + b

print("Encryption--------------------------")
print(cipher)


t1 = ""
t2 = ""
for i in range(len(cipher)):
    if i%2==0:
        t1 = t1 + cipher[i]
    else:
        t2 = t2 + cipher[i]
```

```
cipher2 = t1 + t2
print("Encoding-------------------")
print(cipher2)

plaintext = ""
n = len(cipher2)
n1 = ""
n2 = ""
if(n%2 !=0):
    for i in range(n):
        if(i >= int(n/2 + 1)):
            n2 = n2 + cipher2[i]
        else:
            n1 = n1 + cipher2[i]

    j = 0
    k = 0
    for i in range(n):
        if(i%2 == 0):
            plaintext = plaintext + n1[j]
            j = j+1
        else:
            plaintext = plaintext + n2[k]
            k = k + 1
    print("Decoding-------------------")
    print(plaintext)
else:
    for i in range(n):
        if(i >= int(n/2)):
            n2 = n2 + cipher2[i]
        else:
            n1 = n1 + cipher2[i]

    j = 0
    k = 0
    for i in range(n):
        if(i%2 == 0):
            plaintext = plaintext + n1[j]
            j = j+1
        else:
            plaintext = plaintext + n2[k]
            k = k + 1
```

```
    print("Decoding-------------------")
    print(plaintext)



k = 3
plaintext1 = ""
for i in range(len(plaintext)):
    value = arr[plaintext[i]]
    a = (value - k)%26
    for key, value in arr.items():
        if value == a:
            b = key
            plaintext1 = plaintext1 + b
print("Decryption---------------------------")
print(plaintext1)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp6/exp6/Exp6-14.py
Enter the text : message
Encryption--------------------------
phvvdjh
Encoding-------------------
pvdhhvj
Decoding-------------------
phvvdjh
Decryption--------------------------
message
PS C:\Users\vidit>
```

EXPERIMENT 8

A. Diffie hellman key exchange

```
p = int(input("Enter prime p: "))
privA = int(input("Alice enter your private key: "))
g = int(input("Enter generator g: "))
privB = int(input("Bob enter your private key: "))

temp1 = pow(g, privA)%p
temp2 = pow(g, privB)%p
print(temp1, temp2)

keyA = temp2
keyB = temp1

Ka = pow(keyA, privA)%p
Kb = pow(keyB, privB)%p

if(Ka == Kb): print("Encryption has been successfully implemented")
print(Ka, "is the shared secret key")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp7/exp7/exp7-1.py
Enter prime p: 23
Alice enter your private key: 7
Enter generator g: 9
Bob enter your private key: 13
4 12
Encryption has been successfully implemented
16 is the shared secret key
PS C:\Users\vidit>
```

B. Elgamal

```
import random

def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

p = int(input("Enter a prime number: "))
g = random.randint(1, p - 2)

private = random.randint(2, p-2)
h = pow(g, private)%p
```

```
m = int(input("Enter a message: "))
k = random.randint(2, p-2)
c1 = pow(g, k)%p
c2 = (m*pow(h, k))%p
print("Encryption-------------")
print(c1, c2)

print("Decryption-------------")
s = pow(c1, private)%p
m = (c2 * pow(s, -1, p))%p
print(m)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp7/exp7/exp7-2.py
Enter a prime number: 37
Enter a message: 10
Encryption-------------
1 10
Decryption-------------
10
PS C:\Users\vidit>
```

C. RSA

```
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

p = int(input("Enter a prime number: "))
q = int(input("Enter a prime number: "))
n = p*q
phi = (p-1)*(q-1)
print(n, phi)
e = 0

arr = {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8,
'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r':
17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25}

text = (input("Enter the text : "))

t1 = ""
t2 = ""
for i in range(len(text)):
    if i%2==0:
```

```
        t1 = t1 + text[i]
    else:
        t2 = t2 + text[i]


cipher = t1 + t2
print("Encoding-------------------")
print(cipher)
for e in range(2, phi):
    if gcd(e, phi) == 1:
        break


def modInv(e, phi):
    for d in range(2, phi):
        if((d*e)%phi == 1):
            return d
    return -1


d = modInv(e, phi)

m = int(input("Enter a message: "))
print(e, d)
print("Encryption------------------")
c = pow(m, e)%n
print(c)
print("Decryption------------------")
m = pow(c, d)%n
print(m)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp7/exp7/exp7-3.py
Enter a prime number: 29
Enter a prime number: 17
493 448
Enter the text : hello
Encoding-------------------
hloel
Enter a message: 10
3 299
Encryption------------------
14
Decryption------------------
10
PS C:\Users\vidit>
```

D. Pseudo random key generation

```
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

p = int(input("Enter a prime number: "))
q = int(input("Enter a prime number: "))
n = p*q
phi = (p-1)*(q-1)
print(n, phi)
e = 0


for e in range(2, phi):
    if gcd(e, phi) == 1:
        break


def modInv(e, phi):
    for d in range(2, phi):
        if((d*e)%phi == 1):
            return d
    return -1

d = modInv(e, phi)

for x in range(2, n):
    if(gcd(x, n) == 1): break

print("pseudorandom generation based on rsa are: ")
for i in range(5):
    x = pow(x, e)%n
    print(x)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp7/exp7/exp7-4.py
Enter a prime number: 23
Enter a prime number: 17
391 352
pseudorandom generation based on rsa are:
8
121
331
223
25
PS C:\Users\vidit>
```

E. SHA1

```
import hashlib

str = input("Enter a message: ")
result = hashlib.sha1(str.encode())

print("The hexadecimal equivalent of SHA1 is : ")
print(result.hexdigest())
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp7/exp7/exp7-5.py
Enter a message: messagemetom
The hexadecimal equivalent of SHA1 is :
04729ebad186bc7d8ae75f8dd52a3879927ead02
PS C:\Users\vidit>
```

F. MD5

```
import hashlib

str = input("Enter a message: ")
res = hashlib.md5(str.encode())
print(res.hexdigest())
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp7/exp7/exp7-6.py
Enter a message: secretmessage
27ddff0f0dafc65149db660d29edb31d
PS C:\Users\vidit>
```

G. HMAC

```
import hmac
import hashlib

str1 = input("Enter a secret key: ")
str2 = input("Enter a message: ")
secret_key = b'str1'
message = b'str2'

mac = hmac.new(secret_key, message, hashlib.sha256)
mac_digest = mac.digest()
mac_hexdigest = mac.hexdigest()
print(f"MAC digest: {mac_digest}")
print(f"MAC hexdigest: {mac_hexdigest}")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp7/exp7/exp7-7.py
Enter a secret key: hello
Enter a message: worldison
MAC digest: b"\x14\x9c?\xff\tk\x94B\xce\xecm\xaf\xdb\xcb\xf5\x1b\xe8'\xfb\x92\xe8\x8b\xf3s!\xc4\xbc<@\x10\xb8\xac"
MAC hexdigest: 149c3fff096b9442ceec6dafdbcbf51be827fb92e88bf37321c4bc3c4010b8ac
PS C:\Users\vidit>
```

H. RSA digital signature

```
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

p = int(input("Enter a prime number: "))
q = int(input("Enter a prime number: "))
n = p*q
phi = (p-1)*(q-1)
print(n, phi)
e = 0

for e in range(2, phi):
    if gcd(e, phi) == 1:
        break

def modInv(e, phi):
    for d in range(2, phi):
        if((d*e)%phi == 1):
            return d
    return -1

d = modInv(e, phi)

m = int(input("Enter a message: "))
print(e, d)
print("Digital signature is: ")
s = pow(m, d)%n
print(s)
m1 = pow(s, e)%n
if(m1==m): print("Digital signature has been successfully created!")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/Cryptoexp7/exp7/exp7-8.py
Enter a prime number: 37
Enter a prime number: 13
481 432
Enter a message: 112
5 173
Digital signature is:
112
Digital signature has been successfully created!
PS C:\Users\vidit>
```

I. Elgamal digital signature

```
import random

def gcd(a, b):
    if b==0:
        return a
    else: return gcd(b, a%b)

def generate_keys(q):
    g = random.randint(2, q - 1)
    x_a = random.randint(2, q - 2)
    y_a = pow(g, x_a, q)
    return (q, g, y_a), x_a

def mod_inverse(a, m):
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    return None

def sign_message(message_hash, public_key, private_key):
    q, g, y_a = public_key
    x_a = private_key
    while True:
        k = random.randint(2, q - 2)
        if gcd(k, q - 1) == 1:
            break
    r = pow(g, k, q)
    k_inv = mod_inverse(k, q - 1)
    s = (message_hash - x_a * r) * k_inv % (q - 1)
    return (r, s)

def verify_signature(message_hash, signature, public_key):
    q, g, y_a = public_key
    r, s = signature
    if not (0 < r < q and 0 <= s < q - 1):
        return False
    v1 = (pow(y_a, r, q) * pow(r, s, q)) % q
    v2 = pow(g, message_hash, q)
    return v1 == v2
```

```
q = int(input("Enter a prime no: "))
public_key, private_key = generate_keys(q)
print(f"Public Key (q, g, y_a): {public_key}")
print(f"Private Key (x_a): {private_key}")
original_message_hash = int(input("Enter a number b/w 1 and prime-2: "))

signature = sign_message(original_message_hash, public_key, private_key)
print(f"Signature (r, s): {signature}")
is_valid = verify_signature(original_message_hash, signature, public_key)
print(f"Signature is valid: {is_valid}")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe "c:/Users/vidit/Downloads/Cryptoexp7/exp7/exp7-8(b).py"
Enter a prime no: 23
Public Key (q, g, y_a): (23, 10, 13)
Private Key (x_a): 12
Enter a number b/w 1 and prime-2: 11
Signature (r, s): (17, 21)
Signature is valid: True
PS C:\Users\vidit>
```

J. Schnorr digital signature

```
import random

PRIMENO = int(input("Enter a prime no: "))
generator = int(input("Enter a generator number "))

secretVal = random.randint(1, 97)

X = pow(generator, secretVal) % PRIMENO
y = random.randint(1, 97)
Y = pow(generator, y) % PRIMENO

print("secretVal(secret)= ", secretVal)
print("X= ", X)
print("y=", y)
print("Y=", Y)

c = random.randint(1, 97)
print("c=", c)

z = (y + c * secretVal)
print("z=", z)

val1 = pow(generator, z) % PRIMENO
```

```
val2 = (Y * (X**c)) % PRIMENO
print("val1= ", val1, end=' ')
print(" val2= ", val2)

if (val1 == val2):
    print("Proven")
else:
    print("Failure to prove")
```

```
PS C:\Users\vidit> & C:/Python312/python.exe "c:/Users/vidit/Downloads/Cryptoexp7/exp7/exp7-8(c).py"
Enter a prime no: 19
Enter a generator number 11
secretVal(secret)=  7
X=  11
y= 47
Y= 7
c= 27
z= 236
val1=  7  val2=  7
Proven
PS C:\Users\vidit>
```

EXPERIMENT 9

Elliptic Curve cryptography

```
from tinyec import registry
import secrets
import random


def compress(publicKey):
    return hex(publicKey.x) + hex(publicKey.y % 2)[2:]

curve = registry.get_curve('brainpoolP256r1')
Ka = secrets.randbelow(curve.field.n)
X = curve.g
Pubx = Ka * curve.g
print(Ka)
print("----------------------------------------------------------------------
------------------")
# print(X)
#
print("----------------------------------------------------------------------
------------------")
print("\n")
print("Public key of X is: ", Pubx)
print("----------------------------------------------------------------------
------------------")
print("\n")
# print("X:", compress(Pubx))
#
print("----------------------------------------------------------------------
------------------")
# print("\n")
Kb = secrets.randbelow(curve.field.n)
Y = curve.g
Puby = Kb * curve.g
print(Kb)
print("----------------------------------------------------------------------
------------------")
# print(Y)
```

```
#
print("------------------------------------------------------------------------
------------------")
print("\n")
print("Public key of Y is: ", Puby)
print("------------------------------------------------------------------------
------------------")
print("\n")
# print("Y:", compress(Puby))
#
print("------------------------------------------------------------------------
------------------")
# print("\n")

temp1 = Puby
temp2 = Pubx

sharedKa = temp1 * Ka

sharedKb = temp2 * Kb

# print("Shared key of X: ", sharedKa)
# print("\n")
#
print("------------------------------------------------------------------------
------------------")
# print("Shared key of Y: ",sharedKb)
# print("\n")
#
print("------------------------------------------------------------------------
------------------")
print("\n")

print(compress(sharedKa))
print(compress(sharedKb))

print("\n")
new = secrets.randbelow(curve.field.n)
m = new*curve.g
print("------------------------------------------------------------------------
------------------")
print("\n")
```

```
print("Message is =========== ", m)
print("------------------------------------------------------------------------
-----------------")
print("\n")

k = random.randint(1, 27)
C1 = k*curve.g
C2 = k*Puby + m

print("Cipher 1 is ======= ", C1)
print("------------------------------------------------------------------------
-----------------")
print("Cipher 2 is ======= ", C2)
print("------------------------------------------------------------------------
-----------------")
print("------------------------------------------------------------------------
-----------------")
print(C2 - C1*Kb)
print("------------------------------------------------------------------------
-----------------")
print(m)
```

```
PS C:\Users\vidit> & C:/Python312/python.exe c:/Users/vidit/Downloads/ECC.py
4449397303884530176301986784840181832080239693232415139463396289293779185463 1
--------------------------------------------------------------------------------


Public key of X is:  (61057647963146468418216730001568917968358421834692142291317019109804946626322, 47241896
=> y^2 = x^3 + 56698187605326110043627228396178346077120614539475214109386828188763884139993x + 1757723249732
97466290016490930379502009430552037356014450315161977 51)
--------------------------------------------------------------------------------


122898501137568655119362198747727490902034454570404851639930325112727399951 64
--------------------------------------------------------------------------------


Public key of Y is:  (33127365204724999176446262369902728561065831436753254148352495285632407224268, 54238873
> y^2 = x^3 + 56698187605326110043627228396178346077120614539475214109386828188763884139993x + 175772324973218
74662900164909303795020094305520373560144503151619775 1)
--------------------------------------------------------------------------------



0x6c2c2edf0feb5c432abaf44d31ee67ba4708999105e310edcb05f57f66f4e5721
0x6c2c2edf0feb5c432abaf44d31ee67ba4708999105e310edcb05f57f66f4e5721


--------------------------------------------------------------------------------
```

```
0x6c2c2edf0feb5c432abaf44d31ee67ba4708999105e310edcb05f57f66f4e5721
0x6c2c2edf0feb5c432abaf44d31ee67ba4708999105e310edcb05f57f66f4e5721


--------------------------------------------------------------------------------


Message is ===========  (635779310457791355538417330325127204859114606071261424864067593929277630754411" => y^2 = x^3 + 566981876053261100436272283961783460771206145394752141093868281887638841399993x + 1750809746629001649093037950200943055203735601445031516197751)
--------------------------------------------------------------------------------


Cipher 1 is =======  (19720181235508421888676120517820471470614416846488302898122683637708356641964, 54> y^2 = x^3 + 566981876053261100436272283961783460771206145394752141093868281887638841399993x + 1757723746629001649093037950200943055203735601445031516197751)
--------------------------------------------------------------------------------
Cipher 2 is =======  (23263373729115543659574844554848062085076940392901081824314174136801288783607, 7=> y^2 = x^3 + 566981876053261100436272283961783460771206145394752141093868281887638841399993x + 1757729746629001649093037950200943055203735601445031516197751)
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
(635779310457791355538417330325127204859114606071261424864067593929277630754411, 46253222252578636811808760532611004362722839617834607712061453947521410938682818876388413999931 + 175772324973218388410756977020094305520373560144503151619775)
--------------------------------------------------------------------------------
(635779310457791355538417330325127204859114606071261424864067593929277630754411, 46253222252578636811808760532611004362722839617834607712061453947521410938682818876388413999931 + 175772324973218388410756977020094305520373560144503151619775)
PS C:\Users\vidit>
```