**Lab 1: Key Generation Algorithm Prime numbers**

Read about prime numbers, GCD and generation of random numbers

Write a program using any Object oriented programming language to show generation of keys using prime numbers. The program should randomly pick two prime numbers from a given range, the first output random number is p and the second one is q

**Evaluation Criteria**

1. **Random Generation of p, q**
2. **Correctness of the code**

**Lab 2: Euler’s Totient function**

Read about relatively prime numbers, Eulers and totient

**Euler's theorem** states that, “if p and q are relatively prime, then”, where φ is **Euler's** totient function for integers. That is, is the number of non-negative numbers that are less than q and relatively prime to q.Euler’s Totient function Φ(n) for an input n is count of numbers in {1, 2, 3, …, n} that are relatively prime to n, i.e., the numbers whose GCD (Greatest Common Divisor) with n is 1.

# Modify Lab 1 program and write a program for finding Euler Totient Function Values

**Evaluation Criteria**

1. **Random Generation of p, generation of Euler Totient Function Values**
2. **Prepare one lab report showing the code and the results (screenshots) and explanation of your two algorithms.**

**SOLUTION**

**LAB 1**

Prime numbers are numbers that have exactly 2 divisors which are 1 and the number itself. For example, if N is prime, then the divisors are 1 and N. Examples include; 2, 3, 5, 7, 11, 13, etc.

2 being the smallest prime number.

The Greatest Common Divisor (GCD) is also known as the Highest Common Factor (HCF). The GCD of two integers a and b is the largest integer that divides both a and b without leaving a remainder. For example, the GCD of 12 and 33 is 3.

Random numbers are essential in cryptography for creating keys, nonces, salts, and initialization vectors (IVs). They ensure the security and unpredictability required for cryptographic operations. Types of random number generators include;

* True Random Number Generators (TRNGs)
* Pseudo-Random Number Generators (PRNGs)

Below is a program that randomly picks two prime numbers (p and q) from a given range.

import random

''' This function checks if a number is prime.

It takes num as an argument,and returns true if it is prime and false otherwise.'''

def prime\_num(num):

if num <= 1:

return False

if num <= 3:

return True

if num % 2 == 0 or num % 3 == 0:

return False

i = 5

while i \* i <= num:

if num % i == 0 or num % (i + 2) == 0:

return False

i += 6

return True

"""

This function generates two random prime numbers within a given range. It takes lower\_bound and upper\_bound

as arguments,

and returns, the two prime numbers (p, q).

"""

def key\_gen(lower\_bound, upper\_bound):

while True:

p = random.randint(lower\_bound, upper\_bound)

if prime\_num(p):

break

while True:

q = random.randint(lower\_bound, upper\_bound)

if prime\_num(q) and q != p:

break

return p, q

#Set the lower bound of the range to 100, and the upper bound of the range to 1000.

lower\_bound = 100

upper\_bound = 1000

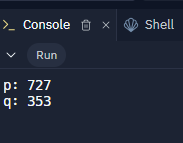
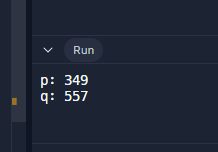
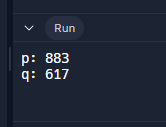
#Call the key\_gen function and assign the result to p and q

p, q = key\_gen(lower\_bound, upper\_bound)

print("p:", p)

print("q:", q)

Screenshot of the output below shows the prime numbers p and q are picked at random.

**SOLUTION**

**LAB 2**

* Relatively prime numbers are also known as co-prime numbers. Two numbers are said to be relatively prime of they have no common factors,and their only common factor is 1.
* Eulers totient function of n means the number of positive integers less than n that are relatively prime to n

import random

''' This function checks if a number is prime.

It takes num as an argument, and returns true if it is prime and false otherwise.'''

def prime\_num(num):

if num <= 1:

return False

if num <= 3:

return True

if num % 2 == 0 or num % 3 == 0:

return False

i = 5

while i \* i <= num:

if num % i == 0 or num % (i + 2) == 0:

return False

i += 6

return True

"""

This function generates two random prime numbers within a given range. It takes lower\_bound and upper\_bound

as arguments,

and returns, the two prime numbers (p, q).

"""

def key\_gen(lower\_bound, upper\_bound):

while True:

p = random.randint(lower\_bound, upper\_bound)

if prime\_num(p):

break

while True:

q = random.randint(lower\_bound, upper\_bound)

if prime\_num(q) and q != p:

break

return p, q

"""

Calculate Euler's Totient Function (phi) of a number(n).

Returns:

The value of Euler's Totient Function (phi) of n.

"""

def phi(n):

result = n

if n <= 1:

return result

p = 2

while p \* p <= n:

if n % p == 0:

result = result \* (1 - 1/p)

while n % p == 0:

n //= p

p += 1

if n > 1:

result \*= (1 - 1/n)

return int(result)

lower\_bound = 100

upper\_bound = 1000

# Generate two random prime numbers

p, q = key\_gen(lower\_bound, upper\_bound)

print("p:", p)

print("q:", q)

# Calculate Euler's Totient Function for p and q

phi\_p = phi(p)

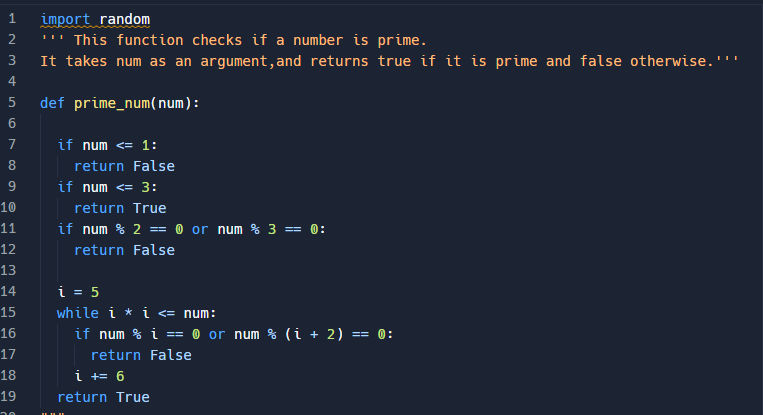
phi\_q = phi(q)

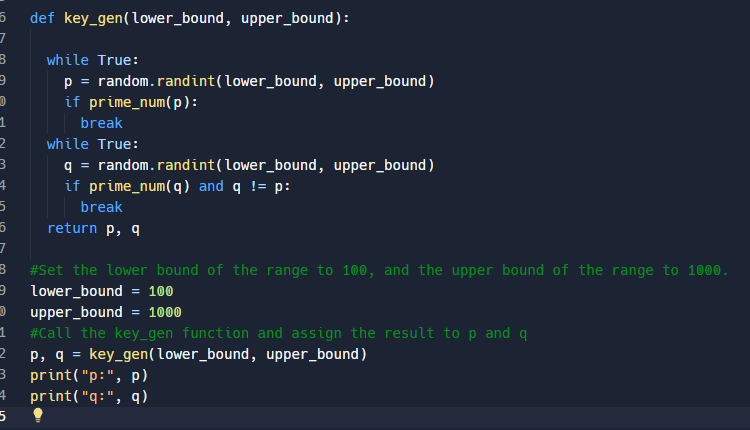
print("phi(p) (Euler's Totient Function of", p, "):", phi\_p)

print("phi(q) (Euler's Totient Function of", q, "):", phi\_q)

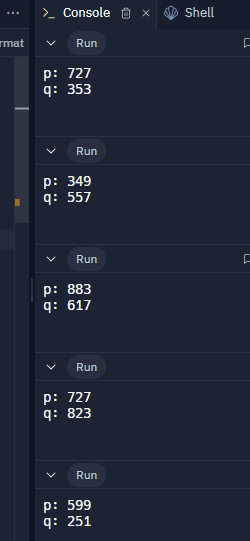
**Lab Report**

**Screenshot for lab1**

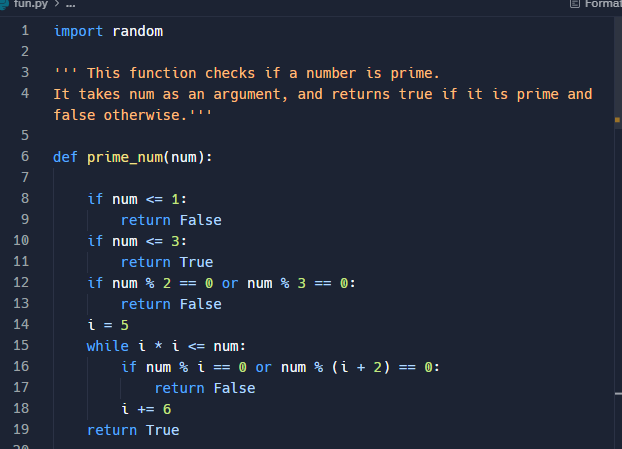


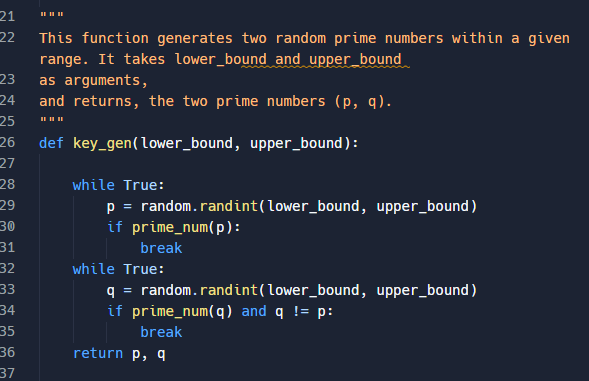


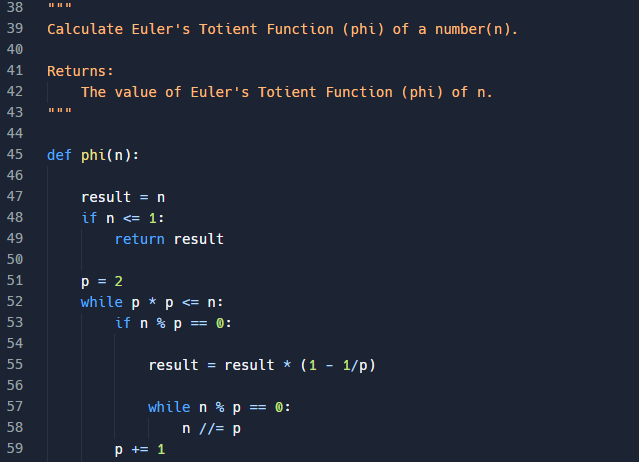
**Output**

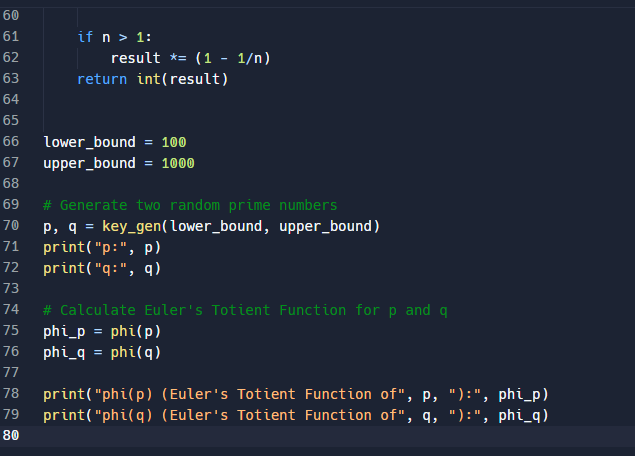


Screenshot for lab 2

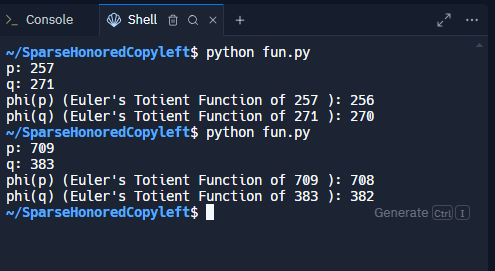








**Output**



**Algorithm 1: Key Generation Algorithm (Prime numbers)**

This algorithm generates two random prime numbers p and q within a specified range. The **prime\_num** function is used to check if a number is prime. It iteratively generates random numbers until both p and q are found to be prime. The condition **q != p** ensures that p and q are distinct prime numbers. The generated prime numbers are crucial for cryptographic operations such as RSA encryption.

**Explanation**

1. **Prime Number Check (prime\_num function):**

**Input:** A number num.

**Process:** The function checks if the number numnumnum is prime.

* + - If num is less than or equal to 1, it returns False (not prime).
    - If num is 2 or 3, it returns True (prime).
    - If num is divisible by 2 or 3, it returns False (not prime).
    - For other numbers, it checks divisibility from 5 up to the square root of num. If any divisor is found, it returns False; otherwise, it returns True.

**Output:** True if num is prime, False otherwise.

1. **Generate Random Prime Numbers (key\_gen function):**

**Input:** A range defined by lower\_bound and upper\_bound.

**Process:**

* + - Randomly generates a number p within the specified range and checks if it is prime using prime\_num. It repeats this process until a prime number is found.
    - Randomly generates another number q within the same range, ensuring q is different from p, and checks if it is prime. This process is repeated until a distinct prime number is found.

**Algorithm 2: Euler's Totient Function Calculation**

This algorithm computes Euler's Totient Function (Φ) for a given integer n. The phi function calculates Φ(n) using Euler's product formula. It iteratively factors n and applies the formula to calculate Φ(n). In this program, the Φ values are calculated separately for p and q using the phi function. The calculated Φ(p) and Φ(q) values represent the number of positive integers less than p and q that are relatively prime to them, respectively.

**Explanation**

1. **Set Range**

The range for generating prime numbers is set between 100 and 1000.

1. **Generate Prime Numbers**

The key\_gen function is called to generate two distinct prime numbers p and q within the specified range.

.

1. **Calculate Euler's Totient Function for p and q**

The phi function is called with p and q to calculate their totient values.