

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 $\Phi(n)$ for an input n is count of numbers in $\{1, 2, 3, \dots, n\}$ 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

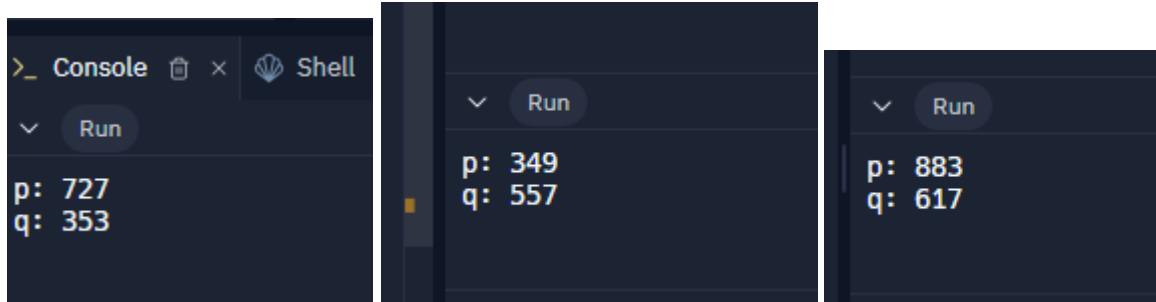
```

```
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 if they have no common factors, and their only common factor is 1.
- Euler's 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 (ϕ) of a number(n).

Returns:

The value of Euler's Totient Function (ϕ) 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

```

1  import random
2  ''' This function checks if a number is prime.
3  It takes num as an argument, and returns true if it is prime and false otherwise. '''
4
5  def prime_num(num):
6
7      if num <= 1:
8          return False
9      if num <= 3:
10         return True
11     if num % 2 == 0 or num % 3 == 0:
12         return False
13
14     i = 5
15     while i * i <= num:
16         if num % i == 0 or num % (i + 2) == 0:
17             return False
18         i += 6
19     return True
20 '''

```




```

6  def key_gen(lower_bound, upper_bound):
7
8      while True:
9          p = random.randint(lower_bound, upper_bound)
10         if prime_num(p):
11             break
12     while True:
13         q = random.randint(lower_bound, upper_bound)
14         if prime_num(q) and q != p:
15             break
16     return p, q
7
8  #Set the lower bound of the range to 100, and the upper bound of the range to 1000.
9  lower_bound = 100
10 upper_bound = 1000
11 #Call the key_gen function and assign the result to p and q
12 p, q = key_gen(lower_bound, upper_bound)
13 print("p:", p)
14 print("q:", q)
15 

```

Output

...

>_ Console    Shell

format

Run

p: 727

q: 353

Run

p: 349

q: 557

Run

p: 883

q: 617

Run

p: 727

q: 823

Run

p: 599

q: 251

Screenshot for lab 2

```
fun.py > ...  
1  import random  
2  
3  ''' This function checks if a number is prime.  
4  It takes num as an argument, and returns true if it is prime and  
5  false otherwise.''  
6  
7  def prime_num(num):  
8      if num <= 1:  
9          return False  
10     if num <= 3:  
11         return True  
12     if num % 2 == 0 or num % 3 == 0:  
13         return False  
14     i = 5  
15     while i * i <= num:  
16         if num % i == 0 or num % (i + 2) == 0:  
17             return False  
18         i += 6  
19     return True  
20
```

```

21 """
22 This function generates two random prime numbers within a given
23 range. It takes lower_bound and upper_bound
24 as arguments,
25 and returns, the two prime numbers (p, q).
26 """
27 def key_gen(lower_bound, upper_bound):
28     while True:
29         p = random.randint(lower_bound, upper_bound)
30         if prime_num(p):
31             break
32     while True:
33         q = random.randint(lower_bound, upper_bound)
34         if prime_num(q) and q != p:
35             break
36     return p, q
37

```

```

38 """
39 Calculate Euler's Totient Function (phi) of a number(n).
40
41 Returns:
42     The value of Euler's Totient Function (phi) of n.
43 """
44
45 def phi(n):
46     result = n
47     if n <= 1:
48         return result
49
50     p = 2
51     while p * p <= n:
52         if n % p == 0:
53
54             result = result * (1 - 1/p)
55
56             while n % p == 0:
57                 n //= p
58
59     p += 1

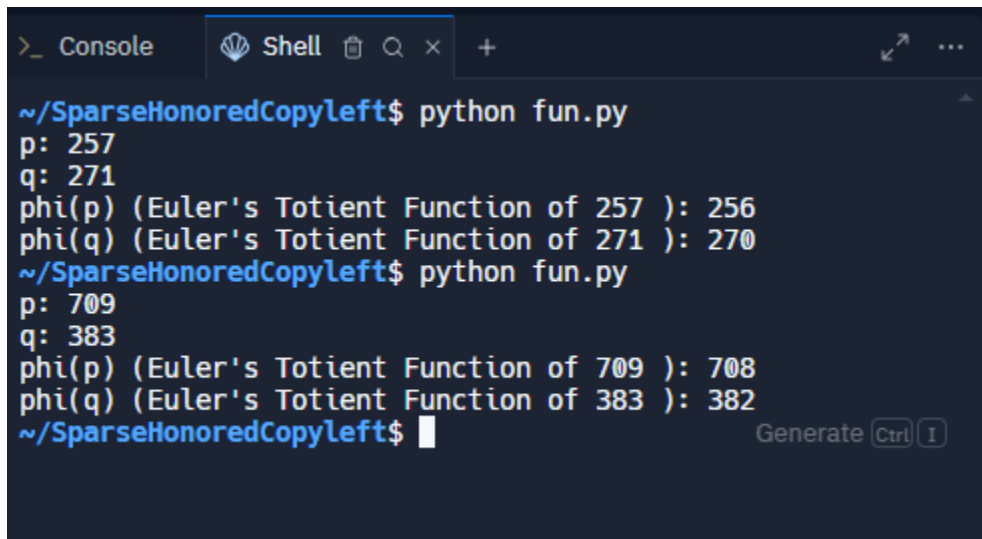
```

```

60
61     if n > 1:
62         result *= (1 - 1/n)
63     return int(result)
64
65
66 lower_bound = 100
67 upper_bound = 1000
68
69 # Generate two random prime numbers
70 p, q = key_gen(lower_bound, upper_bound)
71 print("p:", p)
72 print("q:", q)
73
74 # Calculate Euler's Totient Function for p and q
75 phi_p = phi(p)
76 phi_q = phi(q)
77
78 print("phi(p) (Euler's Totient Function of", p, "):", phi_p)
79 print("phi(q) (Euler's Totient Function of", q, "):", phi_q)
80

```

Output



```

>_ Console  Shell  [Icons] +
~/SparseHonoredCopyLeft$ python fun.py
p: 257
q: 271
phi(p) (Euler's Totient Function of 257 ): 256
phi(q) (Euler's Totient Function of 271 ): 270
~/SparseHonoredCopyLeft$ python fun.py
p: 709
q: 383
phi(p) (Euler's Totient Function of 709 ): 708
phi(q) (Euler's Totient Function of 383 ): 382
~/SparseHonoredCopyLeft$
Generate Ctrl I

```

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 $q \neq p$ 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 `num` 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.

2. 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 $\Phi(n)$ using Euler's product formula. It iteratively factors n and applies the formula to calculate $\Phi(n)$. In this program, the Φ values are calculated separately for p and q using the `phi` function. The calculated $\Phi(p)$ and $\Phi(q)$ 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.

2. Generate Prime Numbers

The `key_gen` function is called to generate two distinct prime numbers p and q within the specified range.

.

3. Calculate Euler's Totient Function for p and q

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