LAB 1: Cipher Fundamentals

Dilan Coral[1]
School of Mathematical and Computational Science
Universidad Yachay Tech, Urcuquí - Ecuador

March 08, 2024.

You can find the code for this project on GitHub. Here is the repository link:

- **Repository**: `https://github.com/Lelis10/Computer-Security---YT.git`

1 Exercise 1: Base-64 encoding, hexadecimal representation, and modulus operator.

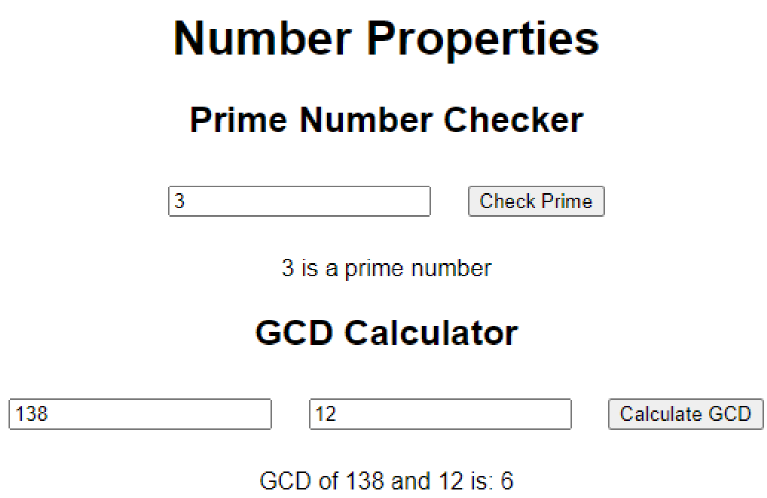
1.1 Develop a web application that allows users to determine certain properties of numbers.

(a) Prime Number Checker:

   (i) Provide an input field where users can enter a number.
   (ii) Implement a button that, when clicked, checks if the entered number is prime or not.
   (iii) Display the result indicating whether the number is prime or not.

(b) GCD Calculator:

   (i) Present two input fields where users can enter two numbers.
   (ii) Implement a button that, when clicked, calculates the Greatest Common Divisor (GCD) of the two entered numbers.
   (iii) Display the GCD result to the user.



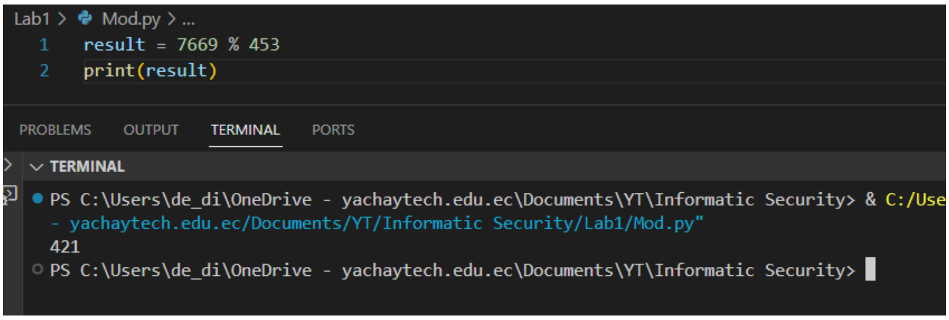
1.2 Determine the Base 64 , Hex and Binary values for the following strings:

1. Hola
   - Base64: `SG9sYQ==`
   - Hexadecimal: `486f6c61`
   - Binary: `01001000 01101111 01101100 01100001`
2. HOLA
   - Base64: `SE9MQQ==`
   - Hexadecimal: `484f4c41`
   - Binary: `01001000 01001111 01001100 01000001`
3. HolA
   - Base64: `SG9sQQ==`
   - Hexadecimal: `486f6c41`
   - Binary: `01001000 01101111 01101100 01000001`

1.3 Determine the following ASCII strings for these encoded formats:

- **Ecuador** (a) **45637561646F72**
- **YACHAY TECH** (b) **01011001 01000001 01000011 01001000 01000001 01011001 01010100 01000101 01000011 01001000**
- **Seguridad Informática** (c) **U2VndXJpZGFkIEluZm9ybT90aWNh**
- **Escuelas de ciencias Matemáticas y Computacionales** (d) **RXNjdWVsYSBkZSBjaWVuY2lhcyBNYXRlbT90aWNhcyB5IENvbXB1dGFjaW9uYWxlcw==**

1.4 Using Python, what is the result of 7669 (mod 453)? Prove that this result is correct.



To find $7669 \mod 453$, you need to divide 7669 by 453 and find the remainder:

$$7669 = 16 \times 453 + 421$$

So, the quotient is 16, and the remainder is 421.
Therefore, $7669 \mod 453 = 421$.
So, the correct result is indeed 421.

2 Greatest common divisor GCD.

2.1 Write a Python program to determine the GCD for the following:

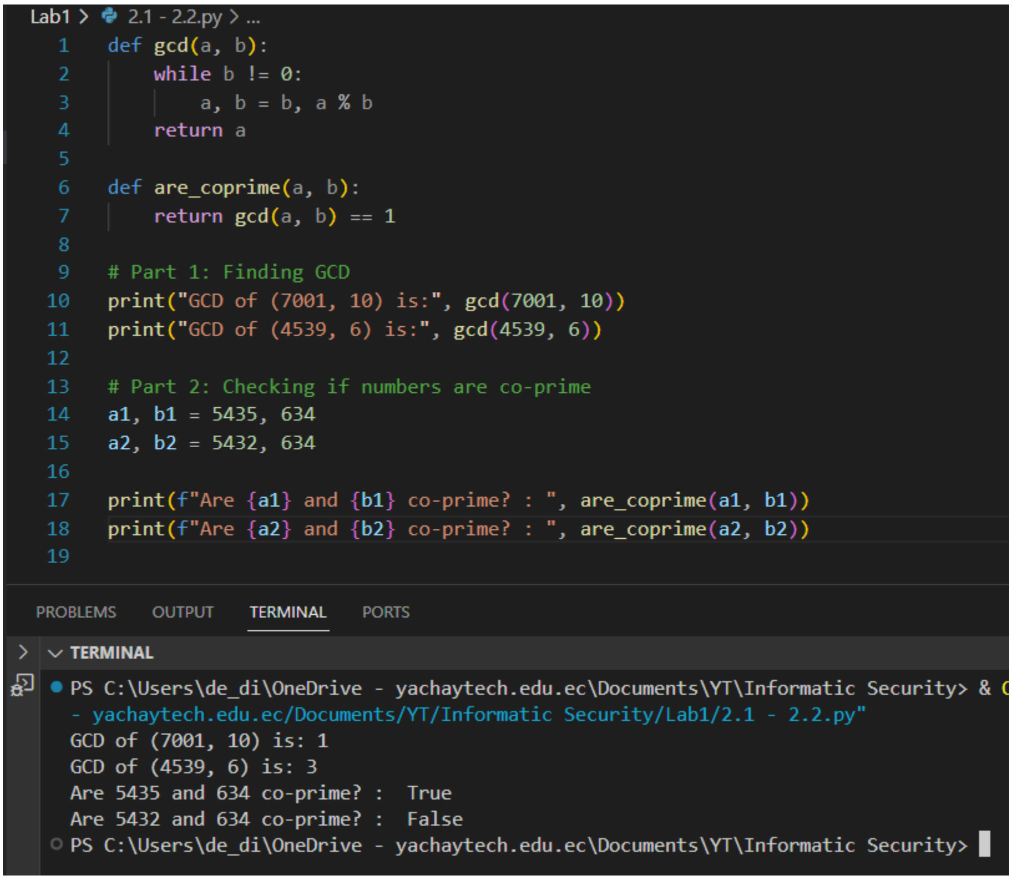
(a) 7001 and 10

(b) 4539 and 6

2.2 Write a Python program to determine the GCD for the following:

(a) 5435 and 634

(b) 5432 and 634



3 Modulus and Exponentiation

3.1 What is the result of the following:

(a) $\mathbf{8}^{13} \bmod (\mathbf{271})$ Steps

Convert the exponent to binary

$$(13)_2 = 1101$$

Create the table

$(13)_2$	$c_0 = 1$
1	$c_1 \equiv 1^2 \cdot 8^1 = 1 \cdot 8 = 8 \bmod 271$
1	$c_2 \equiv 8^2 \cdot 8^1 = 64 \cdot 8 = 512 \equiv 241 \bmod 271$
0	$c_3 \equiv 241^2 \cdot 8^0 = 58081 \cdot 1 = 58081 \equiv 87 \bmod 271$
1	$c_4 \equiv 87^2 \cdot 8^1 = 7569 \cdot 8 = 60552 \equiv 119 \bmod 271$

Solution

$$8^{13} \equiv 119 \bmod 271$$

(b) $12^{23} \bmod (973)$ Steps

Convert the exponent to binary

$$(23)_2 = 10111$$

Create the table

$(23)_2$	$c_0 = 1$
1	$c_1 \equiv 1^2 \cdot 12^1 = 1 \cdot 12 = 12 \bmod 973$
0	$c_2 \equiv 12^2 \cdot 12^0 = 144 \cdot 1 = 144 \bmod 973$
1	$c_3 \equiv 144^2 \cdot 12^1 = 20736 \cdot 12 = 248832 \equiv 717 \bmod 973$
1	$c_4 \equiv 717^2 \cdot 12^1 = 514089 \cdot 12 = 6169068 \equiv 248 \bmod 973$
1	$c_5 \equiv 248^2 \cdot 12^1 = 61504 \cdot 12 = 738048 \equiv 514 \bmod 973$

Solution $12^{23} \equiv 514 \bmod 973$

3.2 Implement a Python program which will determine the result of $C = M^e \bmod (p)$

(a) Prove the following:

- $\text{message} = 101, \text{e} = 7, \text{p} = 293$
- $\text{message} = 4, \text{e} = 11, \text{p} = 79$
- $\text{message} = 5, \text{e} = 5, \text{p} = 53$

Lab1 > 3.2.py > ...

```python
def modular_exponentiation(M, e, p):
    result = 1
    M = M % p
    while e > 0:
        if e % 2 == 1:
            result = (result * M) % p
        e = e // 2
        M = (M * M) % p
    return result

# Test cases
test_cases = [
    (101, 7, 293),  # i. message = 101, e=7, p = 293
    (4, 11, 79),    # ii. message = 4, e=11, p = 79
    (5, 5, 53)      # iii. message = 5, e=5, p = 53
]

for i, (message, e, p) in enumerate(test_cases, start=1):
    result = modular_exponentiation(message, e, p)
    print(f"({i}) Result for message={message}, e={e}, p={p}: {result}")
```

PROBLEMS OUTPUT TERMINAL PORTS

TERMINAL

```
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security> & C
- yachaytech.edu.ec/Documents/YT/Informatic Security/Lab1/3.2.py"
(1) Result for message=101, e=7, p=293: 176
(2) Result for message=4, e=11, p=79: 36
(3) Result for message=5, e=5, p=53: 51
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security>
```

3.3 Implement the Python code given above and determine the highest prime number possible in the following ranges: (a) Up to 100 (b) Up to 1000 (c) Up to 5000 (d) Up to 10000

Lab1 > 3.3.py > ...

```python
def sieve_of_eratosthenes(limit):
    primes = [True] * (limit + 1)
    primes[0] = primes[1] = False
    for i in range(2, int(limit ** 0.5) + 1):
        if primes[i]:
            for j in range(i * i, limit + 1, i):
                primes[j] = False
    return [i for i in range(limit + 1) if primes[i]]

# Define the ranges
ranges = [(1000, 2000), (2000, 3000), (3000, 4000)]

# Find the highest prime number in each range
for i, (start, end) in enumerate(ranges, start=1):
    primes_in_range = sieve_of_eratosthenes(end)
    primes_in_range = [p for p in primes_in_range if p >= start]
    highest_prime = max(primes_in_range)
    print(f"Highest prime number in range {i}: {highest_prime}")
```

PROBLEMS OUTPUT TERMINAL PORTS

TERMINAL

```
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic S
- yachaytech.edu.ec/Documents/YT/Informatic Security/Lab1/3.3.py"
Highest prime number in range 1: 1999
Highest prime number in range 2: 2999
Highest prime number in range 3: 3989
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic S
```

3.4 Which of the following numbers are prime numbers: (a) Is 858599509 prime? (b) Is 982451653 prime? (c) Is 982451652 prime?

(a) Is 858599509 prime? Yes, it is a prime number.

(b) Is 982451653 prime? Yes, it is a prime number.

(c) Is 982451652 prime? No, it is not a prime number.

4 Random numbers

4.1 Implement the Python code given above. Using: a=21, seed=35, c=31, and m=100, prove that the sequence gives 66 17 88 79 90.

```
Lab1 > 4.py > ...
def linear_congruential_generator(a, seed, c, m, n):
    result = []
    X = seed
    for i in range(n):
        X = (a * X + c) % m
        result.append(X)
    return result

# Given parameters
a = 21
seed = 35
c = 31
m = 100

# Generate sequence
sequence = linear_congruential_generator(a, seed, c, m, 5)
print(sequence)
```

```
PROBLEMS   OUTPUT   TERMINAL   PORTS

TERMINAL
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security> &
- yachaytech.edu.ec/Documents/YT/Informatic Security/Lab1/4.py"
[66, 17, 88, 79, 90]
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security>
```

4.2 Determine the sequence for: a=22, seed=35, c=31, and m=100. First four numbers of sequence?

```
Lab1 > 4.py > ...
def linear_congruential_generator(a, seed, c, m, n):
    result = []
    X = seed
    for i in range(n):
        X = (a * X + c) % m
        result.append(X)
    return result

# Given parameters
a = 22
seed = 35
c = 31
m = 100

# Generate sequence
sequence = linear_congruential_generator(a, seed, c, m, 4)
print(sequence)
```

```
PROBLEMS   OUTPUT   TERMINAL   PORTS

TERMINAL
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security> &
- yachaytech.edu.ec/Documents/YT/Informatic Security/Lab1/4.py"
[1, 53, 97, 65]
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security>
```

4.3 Determine the sequence for: a=954365343, seed=436241, c=55119927, and m=1000000. First four numbers of sequence?

```
Lab1 > 4.py > ...
def linear_congruential_generator(a, seed, c, m, n):
    result = []
    X = seed
    for i in range(n):
        X = (a * X + c) % m
        result.append(X)
    return result

# Given parameters
a = 954365343
seed = 436241
c = 55119927
m = 1000000

# Generate sequence
sequence = linear_congruential_generator(a, seed, c, m, 4)
print(sequence)
```

PROBLEMS OUTPUT TERMINAL PORTS

TERMINAL

```
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informati
- yachaytech.edu.ec/Documents/YT/Informatic Security/Lab1/4.py"
[715590, 917297, 157798, 514641]
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informati
```

4.4 Determine the sequence for: a=2175143, seed=3553, c=10653, and m=1000000. First four numbers of sequence?

```
Lab1 > 4.py > ...
def linear_congruential_generator(a, seed, c, m, n):
    result = []
    X = seed
    for i in range(n):
        X = (a * X + c) % m
        result.append(X)
    return result

# Given parameters
a = 2175143
seed = 3553
c = 10653
m = 1000000

# Generate sequence
sequence = linear_congruential_generator(a, seed, c, m, 4)
print(sequence)
```

PROBLEMS OUTPUT TERMINAL PORTS

TERMINAL

```
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informati
- yachaytech.edu.ec/Documents/YT/Informatic Security/Lab1/4.py"
[293732, 114329, 934700, 172753]
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informati
```