**Final Year B.Tech. (CSE) – VII [ 2024-25]**

**6CS451: Cryptography and Network Security Lab (C&NS Lab)**

**Date: 12/08/2024**

# Assignment 3

**PRN:** 21510042 **Name:** Omkar Rajesh Auti

---

## 1. Implementation of Euclidean and Extended Euclidean Algorithm

**Ans:**

The Euclidean and Extended Euclidean algorithms are essential for finding the greatest common divisor (GCD) of two integers. The Extended Euclidean algorithm also finds the coefficients of Bézout's identity, which are useful in solving linear Diophantine equations and in modular arithmetic.

### Euclidean Algorithm

The Euclidean algorithm finds the GCD of two numbers by repeatedly applying the following rule: gcd(a, b) = gcd(b, a % b) until b becomes zero. The GCD is then the non-zero remainder.

### Extended Euclidean Algorithm

The Extended Euclidean algorithm not only computes the GCD of two integers a and b, but also finds integers x and y such that ax + by = gcd(a, b).

**Python Code:**

```
def euclidean_algorithm(a, b):
    """
    Compute the GCD of a and b using the Euclidean algorithm.

    Parameters:
    a (int): First integer.
    b (int): Second integer.

    Returns:
    int: The GCD of a and b.
    """
```

```
    while b != 0:
        a, b = b, a % b
    return a


def extended_euclidean_algorithm(a, b):
    """
    Compute the GCD of a and b, as well as the coefficients x and y
    such that ax + by = gcd(a, b) using the Extended Euclidean algorithm.

    Parameters:
    a (int): First integer.
    b (int): Second integer.

    Returns:
    tuple: (gcd, x, y) where gcd is the GCD of a and b, and x, y are
    the coefficients of Bézout's identity.
    """
    if b == 0:
        return a, 1, 0
    else:
        gcd, x1, y1 = extended_euclidean_algorithm(b, a % b)
        x = y1
        y = x1 - (a // b) * y1
        return gcd, x, y


def main():
    """
    The main function to run the program.
    """
    while True:
        print("\nEuclidean and Extended Euclidean Algorithm")
        print("1. Compute GCD using Euclidean Algorithm")
        print("2. Compute GCD and coefficients using Extended Euclidean
Algorithm")
        print("3. Exit")
        choice = input("Enter your choice: ")
```

```
        if choice == '1':
            a = int(input("\nEnter the first integer (a): "))
            b = int(input("Enter the second integer (b): "))
            gcd = euclidean_algorithm(a, b)
            print(f"\nGCD of {a} and {b} is: {gcd}")
        elif choice == '2':
            a = int(input("\nEnter the first integer (a): "))
            b = int(input("Enter the second integer (b): "))
            gcd, x, y = extended_euclidean_algorithm(a, b)
            print(f"\nGCD of {a} and {b} is: {gcd}")
            print(f"Coefficients x and y are: x = {x}, y = {y}")
            print(f"\nBézout's identity: {a}*({x}) + {b}*({y}) = {gcd}")
        elif choice == '3':
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please try again.")


if __name__ == "__main__":
    main()
```

**Output:**

```
PS C:\Users\omkar\OneDrive\Desktop\SEM7\CNS LAB> python -u "c:\Users\omkar\OneDrive\Desktop\SEM7\CNS LAB\Assignment 3\euclidean.py"

Euclidean and Extended Euclidean Algorithm
1. Compute GCD using Euclidean Algorithm
2. Compute GCD and coefficients using Extended Euclidean Algorithm
3. Exit
Enter your choice: 1

Enter the first integer (a): 15
Enter the second integer (b): 5

GCD of 15 and 5 is: 5

Euclidean and Extended Euclidean Algorithm
1. Compute GCD using Euclidean Algorithm
2. Compute GCD and coefficients using Extended Euclidean Algorithm
3. Exit
Enter your choice: 2

Enter the first integer (a): 48
Enter the second integer (b): 18

GCD of 48 and 18 is: 6
Coefficients x and y are: x = -1, y = 3

Bézout's identity: 48*(-1) + 18*(3) = 6

Euclidean and Extended Euclidean Algorithm
1. Compute GCD using Euclidean Algorithm
2. Compute GCD and coefficients using Extended Euclidean Algorithm
3. Exit
```

This implementation of the Euclidean and Extended Euclidean algorithms is fundamental in cryptography, number theory, and algorithms related to modular arithmetic.