# Question 8: Finding an Inverse Modulo *n*

# Introduction about Extended Euclidean Algorithm

## Extended Euclidean Algorithm

The extended Euclidean algorithm not only computes the GCD but also finds integers x and y such that ax + by = gcd(a, b). This is an extension of the basic Euclidean algorithm.

### Example

Suppose you want to find the GCD of 120 and 23, and also find integers x and y so that

120x + 23y = gcd (120, 23):

Apply the Euclidean algorithm to 120 and 23:

- 120 % 23 = 8 (120 - 5 \* 23 = 8)

- 23 % 8 = 7 (23 - 2 \* 8 = 7)

- 8 % 7 = 1 (8 - 7 = 1)

- 7 % 1 = 0 (Ends here, GCD is 1)

Now, we backtrack through the steps to find x and y:

- 1 = 8 - 7

- 1 = 8 - (23 - 2 \* 8) = 3 \* 8 - 23

- 1 = 3 \* (120 - 5 \* 23) - 23 = 3 \* 120 - 16 \* 23

Thus, x = 3 and y = −16y are the integers found.

# Introduction about Inverse Modul

In modular arithmetic, there is no traditional division operation. Instead, we have the concept of **modular inverses**. The modular inverse of a number A modulo C is a number such that (mod n).

In other words, that mod n = 1. This means that if gcd(a,n) ≠ 1, then A does not have a modular inverse. Only numbers that share no common factors with n (i.e., their greatest common divisor with n is 1) have a modular inverse modulo n.

**Example to illustrate:** Let's take n = 14. We know that 14 is the product of the prime numbers 2 and 7. According to the definition, numbers that share no common factors with 14 are those not divisible by 2 or 7.

**Method for finding a modular inverse using the extended Euclidean algorithm:**

1. Use the Euclidean algorithm to find gcd(a,b)
2. Write down the equation ax + by = gcd (a,b) and solve for x and y.
3. If gcd (a,b) = 1, then x is the inverse of a modulo b. If x is negative, add b to it until you get a positive value.

**Example:** Choose the number 10 to check if it has an inverse modulo 17 and then calculate the modular inverse if it exists.

First, we need to calculate its gcd with 17 using the extended Euclidean algorithm:

* 17 =10 × 1 + 7
* 10 = 7 × 1 + 3
* 7 = 3 × 2 + 1
* 3 = 1 × 3 + 0
* gcd(10,17) = 1

Since gcd (10, 17) = 1, 10 has a modular inverse modulo 17.

Next, solve for x and y in the equation 10x + 17y = 1

* 1 = 7 − 2 × 3
* Replace each 3 with 10 – 7 : 1 = 7 – 2 × (10−7) = 3 × 7 – 2 × 10
* 17 – 10 : 1 = 3 × (17 − 10) – 2 × 10 = 3 × 17 – 5 × 10
* 1 = 3 × 17 – 5 × 10
* x = -5 and y = 3

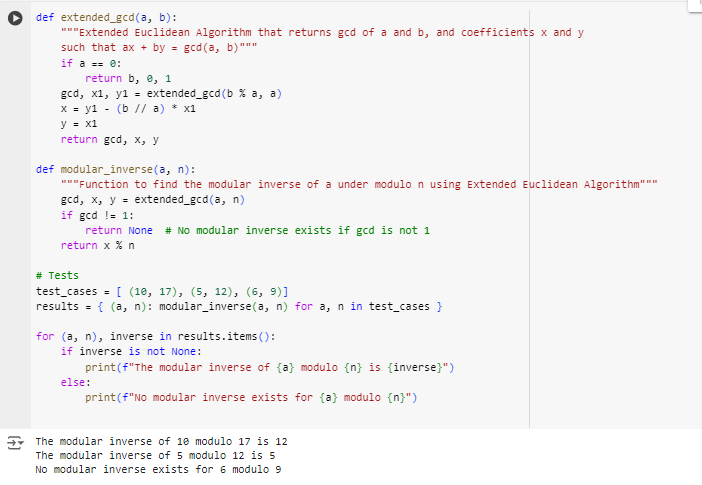
From 1 = 7 – 2 × 3

So, x = −5. Add 17 to get a positive value:

* x = −5 + 17 = 12

Therefore, the inverse 10 modulo 17 is 12 because 10 × 12 ≡ 120 ≡ 1 (mod17).

**Python implementation and result :**



**Explanation of python implementation**

1. Extension function\_gcd(a, b)

This function implements the Extended Euclidean Algorithm, used to find the greatest common divisor (gcd). And also determine the integers x and y such that ax + by = gcd(a, b).

• Base case: If a is 0 then gcd is b, respectively x = 0 and y = 1 are instantaneous solutions.

• Recursive: If a is non-zero, the algorithm calculates gcd(b % a, a) and then uses the result from the recursive call to calculate x and y for the current step.

* x is updated according to the formula: x = y1 - (b // a) \* x1.
* y is taken directly from x1 from the previous recursive call.

2. Function module\_inverse(a, n)

This function finds the modular inverse of the modulus below n using the Extended Euclidean Algorithm. The modular inverse of a is a number x such that (a \* x) % n = 1.

• Check gcd: Initially calculate gcd(a, n). If gcd is not equal to 1, then the modular inverse does not exist because a and n are not coprime.

• Calculate the inverse: If gcd = 1, use the x value from extend\_gcd to determine x %n, this is the modulus inverse to find.

3.Testing the Functions:

The script defines test cases (10, 17), (5, 12), and (6, 9).

It computes the modular inverse for each pair using a dictionary comprehension and then iterates through the results to print the inverse or a message stating no inverse exists.

* The modular\_inverse function returns the modular inverse if it exists, and returns None if it does not exist.

### Output Explanation

* **For (10, 17)**: The gcd is 1, so it calculates the modular inverse, which is 12. The output "The modular inverse of 10 modulo 17 is 12" confirms that 10 × 12 mod  17 = 1
* **For (5, 12)**: Similarly, the gcd of 5 and 12 is 1, allowing the computation of the modular inverse, which is 5. The output "The modular inverse of 5 modulo 12 is 5" confirms that 5× 5 mod 12 = 1.
* **For (6, 9)**: The gcd is 3 (since both 6 and 9 are divisible by 3), hence no modular inverse exists, as reflected by "No modular inverse exists for 6 modulo 9".