# Question-1

Write a Python function is_multiple(n, m) that checks if an integer n is a multiple of another integer m. The function should return True if n is a multiple of m (i.e., n = m × i for some integer i), and False otherwise.

Sample Input: 5, 9

Expected Output:False

In [7]:
```python
def is_multiple(n, m):
    """Check if n is a multiple of m.

    Args:
        n: Integer to check if it's a multiple of m.
        m: Integer to check if it's a divisor of n.

    Returns:
        bool: True if n is a multiple of m, False otherwise.
    """
    return n % m == 0


# Alternative approach using lambda
check_multiple = lambda n, m: n % m == 0

if __name__ == "__main__":
    n = int(input("Enter first number: "))
    m = int(input("Enter second number: "))

    print(f"{n} is multiple of {m}: {is_multiple(n, m)}")
    print(f"{n} is multiple of {m}: {check_multiple(n, m)}")
```

```
Enter first number: 18
Enter second number: 12
18 is multiple of 12: False
18 is multiple of 12: False
```

# Question-2

Write a short Python function, is even(k), that takes an integer value and returns True if k is even, and False otherwise. However, your function cannot use the multiplication, modulo, or division operators.

sample input: 8

sample ouput: True

In [9]:
```python
## Hint-1:- In Binary Form of a number LSB is 0 if the number is even

def is_even(k: int) -> bool:
    """Check if an integer is even without using *, /, or % operators.

    Args:
        k (int): The integer to check.

    Returns:
        bool: True if k is even, False otherwise.
    """
    return (k & 1) == 0


if __name__ == "__main__":
    k = int(input("Enter an integer: "))
    print(f"Is {k} even? {is_even(k)}")
```

```
Enter an integer: 110
Is 110 even? True
```

## Question-3

Write a short Python function, minmax(data), that takes a sequence of one or more numbers, and returns the smallest and largest numbers, in the form of a tuple of length two. Do not use the built-in functions min or max in implementing your solution.

sample input:- 889,1,2,4,6,800,1566

sample output:- 1,1566

In [13]:

```python
def minmax(data):
    """Find the smallest and largest numbers in a sequence without using min/m

    Args:
        data: A non-empty sequence of one or more numbers.

    Returns:
        tuple: (smallest, largest) in the sequence.

    Raises:
        ValueError: If the input sequence is empty.

    """
    if not data:
        raise ValueError("Input sequence must not be empty.")

    smallest = largest = data[0]  # Initialize with the first element

    for num in data[1:]:  # Iterate from the second element
        if num < smallest:
            smallest = num
        elif num > largest:
            largest = num

    return smallest, largest


if __name__ == "__main__":

    input_data = list(map(int, input().split()))
    result = minmax(input_data)
    print(f"Smallest and largest numbers: {result}")
```

```
1  2 4 5
Smallest and largest numbers: (1, 5)
```

# Question-4

Write a short Python function that takes a positive integer n and returns the sum of the squares of all the positive integers smaller than n.

sample input:- 6

sample ouptput:- 55

In [21]:
```python
def sum_of_squares(n: int) -> int:
    """Calculate the sum of squares of all positive integers smaller than n.

    Args:
        n: A positive integer.

    Returns:
        The sum of squares from 1² to (n-1)².

    Raises:
        ValueError: If n is not a positive integer.
    """

    if n <= 0:
        raise ValueError("Input must be a positive integer.")

    # Actual formula: sum = n*(n+1)*(2n+1)//6.
    # Here till n-1 only needed. So substitute n-1 in place of n.
    # Using the mathematical formula: sum = (n-1) * n * (2n - 1) // 6

    return (n - 1) * n * (2 * n - 1) // 6


if __name__ == "__main__":

    num = int(input("Enter Number: "))

    print(f"sum of squares from 1 to {num-1} is {sum_of_squares(num)}")

    ## One liner...
    print(sum(ele**2 for ele in range(num)))
```

```
Enter Number: 7
sum of squares from 1 to 6 is 91
91
```

## Question-5

Write a short Python function that takes a positive integer n and returns the sum of the squares of all the odd positive integers smaller than n.

Sample input:- 5

Sample output:- 10

In [45]:
```python
def sum_of_odd_squares(n):
    """Return the sum of squares of all odd positive integers smaller than n.

    Args:
        n: A positive integer.

    Returns:
        Sum of squares of odd integers from 1 to n-1.

    Raises:
        ValueError: If n is not a positive integer.
    """
    if not isinstance(n, int) or n <= 0:
        raise ValueError("n must be a positive integer")

    k = (n) // 2  # Count of odd numbers < n
    return k*(2*k+1)*(2*k-1) // 3


if __name__ == "__main__":
    num = int(input("Enter a positive integer: "))
    print(f"Sum of odd squares from 1 to {num-1} is {sum_of_odd_squares(num)}"

    ## one liner:-
    sum_of_odd_squares = lambda n: sum(k ** 2 for k in range(1, n, 2))

    print(sum_of_odd_squares(num))
```

```
Enter a positive integer: 25
Sum of odd squares from 1 to 24 is 2300
2300
```

In [ ]:

In [ ]: