# Lab 7 – Recursive Functions

## Part 1 – Lab Checkoff (7 pts)

1. Write a function f which takes a function g and up to two additional arguments, returning the return value of a call to g with the other arguments being passed as its arguments. For example,

```
>>> f(math.sqrt, 25)
5.0
>>> f(math.pow, 5, 2)
25.0
```

2. (2 pts) Traditionally, we use the multiplication operator (*) to multiply two numbers. But multiplication is essentially repeated addition. For instance, 3 multiplied by 4 (3 * 4) is essentially 3 + 3 + 3 + 3, which equals 12.

Write a Python function named *recursive_multiply* that takes two **non-negative** integers, a and b, and returns their product by using the above concept of repeated addition. However, you're only allowed to use addition and subtraction operators (no direct multiplication). The function should be implemented **recursively**.

For example:
*Input: 3, 4*
*Output: 12*

3. (3 pts) Deepcopy creates a new copy of an object without any shared reference with the original one. It constructs a new compound object and then, recursively inserts copies into it of the objects found in the original.

Write a python function *deep_copy_list* to implement deepcopy of a list without copy module using recursion. Design a few test cases to verify the deecopy.
Note:

- You can use isinstance(object, classinfo) to check if the object is an instance of the class
- You only need to consider the case of list in list.
- Think of a few test cases for the *deep_copy_list* function, that can demonstrate deep copy. One way is to modify the copied object and see if the original object is affected.

## Part 2 – Assignment

**Deadline**: Monday, Oct. 27th 2025 11:59 pm

Please submit your proposal presentation slides on Canvas by end of next Monday!