Recursive Functions

`factorial` Algorithm

Create a recursive function called `factorial` that will calculate the product of all positive integers less than or equal to an int, `n`. E.g.,

`factorial(n=5)` would return: 5*4*3*2*1 = **120**

`factorial(n=2)` would return: 2*1 = **2**

`factorial(n=1)` would return: 1 = **1**

`factorial(n=0)` would return: **1**

Conceptually, what will our **base case** be?

What will our **recursive case** be?

What is an **edge case** for this function? How could we account for it?

Visualizing recursive calls to `factorial`

Let’s write the `factorial` function in VS Code!

Memory diagram

```python
def find_index_of(hay: list[int], ndl: int) -> int | None:
    return search(hay, ndl, 0, len(hay) - 1)

def search(hay: list[int], ndl: int, low: int, high: int) -> int | None:
    if low > high:
        return None

    mid: int = (low + high) // 2

    if hay[mid] == ndl:
        return mid
    elif hay[mid] < ndl:
        return search(hay, ndl, mid + 1, high)
    else:
        return search(hay, ndl, low, mid - 1)

haystack: list[int] = [1, 2, 8, 13]
needle: int = 8
result: int | None = find_index_of(haystack, needle)

if result is not None:
    print(f"Needle {needle} found at index: {result}")
else:
    print(f"Needle {needle} not found.")
```

What does this code do?

Which functions (if any) are recursive?
How do we know?

On which line(s) (if any) do the following occur?

- Base case?
- Recursive case?
- Edge case?

Memory diagram

```python
def find_index_of(hay: list[int], ndl: int) -> int | None:
    return search(hay, ndl, 0, len(hay) - 1)

def search(hay: list[int], ndl: int, low: int, high: int) -> int | None:
    if low > high:
        return None

    mid: int = (low + high) // 2

    if hay[mid] == ndl:
        return mid
    elif hay[mid] < ndl:
        return search(hay, ndl, mid + 1, high)
    else:
        return search(hay, ndl, low, mid - 1)

haystack: list[int] = [1, 2, 8, 13]
needle: int = 8
result: int | None = find_index_of(haystack, needle)

if result is not None:
    print(f"Needle {needle} found at index: {result}")
else:
    print(f"Needle {needle} not found.")
```

When developing a recursive function:

Base case:

- ❏ Does the function have a clear base case?
  - ❏ Ensure the base case returns a result directly (without calling the function again).
- ❏ Will the base case *always* be reached?

Recursive case:

- ❏ Ensure the function moves closer to the base case with each recursive call.
- ❏ Combine returned results from recursive calls where necessary.
- ❏ Test the function with edge cases (e.g., empty inputs, smallest and largest valid inputs, etc.). Does the function account for these cases?