# Time Complexity + Recursion

## Assignment

PW SKILLS

# Find time complexity of below code blocks :

## Problem 1 :

```
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quicksort(left) + middle + quicksort(right)
```

## Problem 2 :

```
def nested_loop_example(matrix):
    rows, cols = len(matrix), len(matrix[0])
    total = 0
    for i in range(rows):
        for j in range(cols):
            total += matrix[i][j]
    return total
```

## Problem 3 :

```
def example_function(arr):
    result = 0
    for element in arr:
        result += element
    return result
```

## Problem 4 :

```
def longest_increasing_subsequence(nums):
    n = len(nums)
    lis = [1] * n
    for i in range(1, n):
        for j in range(0, i):
            if nums[i] > nums[j] and lis[i] < lis[j] + 1:
                lis[i] = lis[j] + 1
    return max(lis)
```

**Problem 5 :**

```
def mysterious_function(arr):
    n = len(arr)
    result = 0
    for i in range(n):
        for j in range(i, n):
            result += arr[i] * arr[j]
    return result
```

# Solve the following problems on recursion

## Problem 6 : Sum of Digits

Write a recursive function to calculate the sum of digits of a given positive integer.
sum_of_digits(123) -> 6

## Problem 7: Fibonacci Series

Write a recursive function to generate the first n numbers of the Fibonacci series.
fibonacci_series(6) -> [0, 1, 1, 2, 3, 5]

## Problem 8 : Subset Sum

Given a set of positive integers and a target sum, write a recursive function to determine if there exists a subset of the integers that adds up to the target sum.

subset_sum([3, 34, 4, 12, 5, 2], 9) -> True

## Problem 9: Word Break

Given a non-empty string and a dictionary of words, write a recursive function to determine if the string can be segmented into a space-separated sequence of dictionary words.

word_break("leetcode", ["leet", "code"]) -> True

## Problem 10 : N-Queens

Implement a recursive function to solve the N-Queens problem, where you have to place N queens on an N×N chessboard in such a way that no two queens threaten each other.

n_queens(4)

```
[
 [".Q..",
  "...Q",
  "Q...",
  "..Q."],
 ["..Q.",
  "Q...",
  "...Q",
  ".Q.."]
]
```