Monsoon 2025 | Graduate Algorithms: Quiz | *Max: 50 points*

Date: **02 September** | *Duration: 60 mins*

10% if you write **"I don't know"** for any (sub)question

Q1 [5 points] Assume that n is a power of 2 for this question.

Let $\langle a_0 \dots a_{n-1} \rangle$ denote the coefficients of a degree-$(n-1)$ polynomial $A(x)$. Fill in both the blanks of this FFT algorithm that runs in $O(n \log n)$ time and returns the discrete Fourier transform of $A(x)$.

```
def DFT(A : ⟨a_0 ... a_{n-1}⟩):
    if n=1: return ____
    else:
        ⟨y_0^even, ... y_{n/2-1}^even⟩ ← DFT(⟨a_0, a_2, ... a_{n-2}⟩)
        ⟨y_0^odd, ... y_{n/2-1}^odd⟩ ← DFT(⟨a_1, a_3, ... a_{n-1}⟩)
        for k=0 to n−1:

            y_k = _____ (write a mathematical expression/formula)
        return ⟨y_0, ... y_{n-1}⟩
```

Q2 [5+5=10 points] Consider the following algorithm.

```
def Cruel(A[1...n]):
    if n > 1:
        Cruel(A[1...n/2])
        Cruel(A[n/2+1...n])
        Unusual(A[1...n])

def Unusual(A[1...k]):
    if k=2:
        if A[1] > A[2]:
            swap A[1] and A[2]
    else:
        // swap 2nd and 3rd quarters
        for i = 1 ... k/4:
            swap A[i+k/4] and A[i+k/2]
        Unsual(A[1...k/2]) // recurse on left half
        Unsual(A[k/2+1...k]) // recurse on right half
        Unsual(A[k/4+1...3k/4]) // recurse on middle half
```

Analyse the complexity of both the algorithms.

Q3 [15 points] An element x in an array A (in which elements may be occurring multiple times) is said to be a *semi-majority* if its frequency in A is **greater than** $|A|/3$. Observe that not every array would have a semi-majority element, and that there may more than one such element.

For example, 6 is a semi-majority of [6,2,4,6,2,4,3,6], both 4 and 6 are semi-majority of [6,2,4,6,2,4,4,6], but [1,1,3,3,5,5,7,7] has no semi-majority element.

Design a divide-and-conquer subroutine to return any one semi-majority of an input array A, or return `null` if there is no such element (or, return -1, or do a small dance, or do something other than returning an element from A); assume that A has only positive integers. Write its pseudocode, add enough comments or a separate explanation to make it clear, and then derive its time-complexity. An $O(n \log n)$ algorithm is required for full-credit; however, a slower but correct approach will fetch more marks than a fast but incorrect approach.

Q4 [20 points] You are given an array A of n distinct integers. Suppose that you also know the sum, denoted S. You have to figure out if A can be partitioned into 3 equal partitions, i.e., every element of A must belong to exactly one partition, such that all partitions have equal sum.

For example, [1,2,3,4,5] can be partitioned into [2,3], [1,4], and [5], all with sum 5. But, [1,2,3,4,8] cannot be partitioned into 3 equal-sum partitions.

Design a recursive backtracking algorithm, write its pseudocode and explain the steps. Correctness should be clear from the construction of the algorithm, else you need to explain why your algorithm is correct.

Write down a recurrence for the time-complexity and try to find some solution of the algorithm.
