COL351: Analysis and Design of Algorithms — Announced on: July 24, 2026 (Friday)

Tutorial Sheet 1

1. Prove or disprove:
   a) $\log_2(n!) = \Theta(n \log_2 n)$.
   b) $n! = \Theta(n^n)$.

2. Prove or disprove:
   a) $\log n = \mathcal{O}(n^\varepsilon)$ for any constant $\varepsilon > 0$.
   b) $4^n = \mathcal{O}(2^n)$.
   c) $n^2 + \mathcal{O}(n) = \mathcal{O}(n^2)$.
   d) $\lceil n \rceil = \Theta(n)$.

3. Identify and discuss the error(s) in the following false statement and its false proof. Mention the erroneous step(s) clearly.

   Claim: Let $f(n) = \sum_{i=1}^{n} i$. Then, $f(n) = \mathcal{O}(n)$.

   Proof: By induction on n.

   Let the induction hypothesis be $P(n) : f(n) = \mathcal{O}(n)$.

   Base case: $f(1)$ equals 1, which is $\mathcal{O}(1)$. Thus, $P(1)$ is TRUE.

   Induction step: Observe that $f(n+1) = f(n) + (n+1)$. Since $f(n) = \mathcal{O}(n)$ by induction assumption, and since $n + 1 = \mathcal{O}(n)$, we have that $f(n+1) = \mathcal{O}(n)$, which, in turn, is $\mathcal{O}(n+1)$. This proves the claim. □

4. Consider the following algorithm for checking whether a given number n is prime.

ALGORITHM 1: Checking primality

Input: An integer $n > 1$.
Output: Yes/No

```
1 if n equals 2 or 3 then
2     return Yes
3 for i = 2 to ⌊√n⌋ do
4     if i divides n then
5         return No
6 return Yes
```

Prove or disprove:

a) The algorithm is correct.

b) The algorithm runs in polynomial time.

5. Let A and B be two sorted arrays of length n each. Assume that all elements within and across the two arrays are distinct. Design an $\mathcal{O}(\log n)$ algorithm to compute the n^{th} smallest element of the union of A and B.

6. You are given a sorted (from smallest to largest) array A of n distinct integers which can be positive, negative, or zero. You want to decide whether or not there is an index i such that $A[i] = i$. Design the fastest algorithm you can for solving this problem.

7. Imagine you are a trader in a market in which a particular commodity can be bought only once and sold only once during a season. Naturally, the purchase much precede the sale. The price of the commodity fluctuates every day. The data from the last season is now available, and you want to know the maximum profit you could have made. Design an $\mathcal{O}(n)$ algorithm for this problem, where the input is the list of prices $\{p_1, p_2, \ldots, p_n\}$ for the n days in the last season and the output is the maximum possible profit. Assume that addition, subtraction, pairwise comparison, etc. takes unit time.

   Sample input prices: $70, 100, 140, 40, 60, 90, 120, 30, 60$.
   Output: Max profit: 80.