Data Structures and Algorithms

Tutorial Week 2

Coding Problems

1. Implement either the **Merge Sort** or **Quick Sort** algorithm, but modify it to accept a **comparison function** as an argument.
    - For example:
        - If the function is `greater_than`, the sorting will be in descending order.
        - If the function is `less_than`, the sorting will be in ascending order.

    - **Note**: Ensure that everyone implements at least one of these sorting algorithms as it can be reused in **Assignment 1**.
    - Example function signature:
```
void merge_sort(int arr[], int n, bool (*comp)(int, int));
```

Discussion Problems

1. **Lower Bound of Comparison-Based Sorting Algorithms**:
    - Why do comparison-based sorting algorithms have a theoretical lower bound of $O(nlogn)$?

2. **Three-way or k-way Merge Sort**:
    - What would be the time complexity of the **Merge Sort** algorithm if we partition into three (or k) parts instead of two?

3. **Quick Sort Worst case**:
    - For what input does **Quick Sort** have a worst-case time complexity of $O(n^2)$? (How can we avoid this?)