INSTRUCTIONS:

1. Use the provided latex template to write your solutions.
2. Submit your solution to Gradescope: make sure only one of your group members submits. After submitting, make sure to add your group members.

Problem 1 (10 points).

Consider recurrence relation $T(n) = \Theta(n) + T(a \cdot n) + T(b \cdot n)$, $T(1) = 1$, $0 < a < 1$, $0 < b < 1$. Prove the following:

1. $T(n) = \Theta(n)$, if $a + b < 1$.
2. $T(n) = \Theta(n \cdot \log n)$, if $a + b = 1$.

Problem 2 (10 points).

Let S be an array with n distinct positive integers. We say two indices (i, j) form an inversion of S if we have $i < j$ and $S[i] > S[j]$. Design a divide-and-conquer algorithm to count the number of inversions in S. Your algorithm should run in $O(n \log n)$ time.

Problem 3 (10 points).

Let S be an array with n distinct positive integers. We say two indices (i, j) form a big-inversion of S if we have $2 \cdot i < j$ and $S[i] > 2 \cdot S[j]$. Design a divide-and-conquer algorithm to count the number of big-inversions in S. Your algorithm should run in $O(n \log n)$ time.

Problem 4 (10 points).

You are given n points $P = \{p_1, p_2, \cdots, p_n\}$ on 2D plane, represented as their coordinates. You are informed that the convex hull of P contains $O(\log \log n)$ points in P. Design an algorithm to compute the convex hull of P in $O(n \cdot \log \log n)$ time. You may assume that no three points in P are on the same line.

Problem 5 (10 points).

If we add one more partition step in find-pivot function in selecting problem, and use the size 3 for each subarray, the algorithm would become:

```
function find-pivot (A, k)
    Partition A into n/3 subarrays;
    Let M be the list of medians of these n/3 subarrays;
    Partition M into n/9 subarrays;
    Let M' be the list of medians of these n/9 subarrays;
    return selection(M', |M'|/2)
end function;
```

Analyze the running time of the new selection function with the find-pivot function described above.

Problem 6 (10 points).

Quicksort is another widely used sorting algorithm. Although its worst-case running time is $\Theta(n^2)$, its average running time is $O(n \log n)$. Quicksort partitions the list A using a random pivot x, like the randomized algorithm for selection problem. Then, the algorithm recursively sorts elements smaller than x, then x, and then elements larger than x. The pseudocode of quicksort is as follows.

```
function Quicksort (A)
    if |A| ≤ 3 then
        Sort A by brute-force and output the sorted list of A
    else
        Choose a pivot x ∈ A uniformly at random
        for each element a_i of A
            Put a_i in A⁻ if a_i < x
            Put a_i in A⁺ if a_i > x
        end for
        A₁⁻ = Quicksort(A⁻)
        A₁⁺ = Quicksort(A⁺)
        Return (A₁⁻, x, A₁⁺)
    end if
end function
```

Show that the EXPECTED running time of the above Quicksort algorithm is $\Theta(n \log n)$

Problem 7 (10 points).

We learned that Graham-Scan algorithm can output the convex hull of a set of points P = $\{p_1, p_2, ..., p_n\}$ on 2D plane in $\Theta(n \cdot \log n)$ time. Prove that $\Theta(n \log n)$ is the best asymptotic running time we can expect, i.e., the OPTIMAL algorithm for the convex hull problem runs in $\Theta(n \cdot \log n)$ time.
Hint: We can prove this by showing that the sorting problem can be *reduced* to the convex hull problem in linear time. Since we know that the optimal algorithm for sorting takes $\Theta(n \log n)$ time, therefore the optimal algorithm for convex hull problem must take $O(n \log n)$ time. To achieve this, you will need to design an algorithm for sorting problem using the convex-hull algorithm. Specifically, given an array $S[1 \cdots n]$ with n distinct positive integers, you need to create an instance of the convex-hull problem (i.e., a set of points P, each of which is represented as coordinates on 2D plane) from S. You then use any algorithm (say, Grahan-Scan algorithm) to compute the convex-hull of P. You finally need to return the sorted list of S.

Complete the two procedures (*Procedure 1* and *Procedure 2*) within the following algorithm. Both *Procedure 1* and *Procedure 2* should run in linear time.

---

function sorting-using-convex-hull ($S[1 \cdots n]$)

  Procedure 1: create a set of 2D points P from S;

  $C \leftarrow$ Graham-Scan (P); /* points in C will be sorted in counter-clockwise order */

  Procedure 2: compute the sorted list of S from C;

end function
