Problem 1. Consider the following recurrence (for the time of some algorithm).

$$T(n) = 3T(n/5) + 2n + 1, \qquad T(1) = 4$$

(a) Calculate $T(25)$ by hand. Show your work.

(b) Use the recursion tree method to solve the recurrence exactly, assuming n is a power of 5. For each subpart *briefly justify* and / or *show your work* when appropriate.

(a) Draw the tree. You should show at least three levels at the top and at least two levels at the bottom (as done in class).

(b) What is the height of the tree? (Note that a tree with one node has height 0, a tree with a root and some children has height 1, etc.)

(c) How many leaves are there?

(d) What is the total work done by the leaves?

(e) What is the size of each subproblem at level i? (Note that the root is at level 0, its children are at level 1, etc.)

(f) How much work does each subproblem at level i (above the leaves do)?

(g) What is the total work for level i (above the leaves)?

(h) Write a summation for the total work not including the leaves?

(i) Simplify the summation.

(j) What is the total work for the entire algorithm?

(k) Verify the base case, $T(1)$.

Problem 2. Design an optimal strategy to find the second largest number in an unsorted array of size n that uses at most $n + lg\, n - 2$ comparisons. You may assume distinct elements in the array and n to be a power of 2. You don't need to write pseudo-code. A concise English description would be sufficient. Keep your sentences short. The total number of comparisons should be clearly defined in your design strategy.

Problem 3. We want to find the maximum and the minimum elements in an unsorted array of size, n using two different optimal strategies that yield the same runtime. You may assume n to be a power of 2.

(a) Write pseudo-code for an optimal iterative algorithm. Analyze the runtime exactly.

(b) Write pseudo-code for an optimal divide and conquer approach. Analyze the runtime exactly.