The due date for this assignment isThursday, 10/20th, at 11.59 pm*.* This assignment is worth 12.5% of your final grade.

Any sign of collaboration will result in a 0 and is reported to the Graduate Academic Integrity Board. The programming assignment will be done individually. No collaboration is allowed between students. No code from online resources can be used besides the code I will share with you. The late submission policy described in the syllabus will be applied.

**Part 1: Recurrences (20 points, 5 points each)**

Solve the following recurrences using the substitution method. Subtract off a lower-order term to make the substitution proof work or adjust the guess in case the initial substitution fails.

1. Initial Guess: . Show for some constant
2. Initial Guess Show for some constant
3. Initial Guess: Show for some constant
4. Initial Guess: Show for some constant

**Part 2: Master’s Theorem (20 points, 4 points each)**

For each of the following recurrences, give an expression for the running time if the recurrence can be solved with the Master Theorem. Please provide the case number and constants ( if applicable. Otherwise, indicate that the Master Theorem does not apply.

**Part 3: Heap Sort (35 points)**

A d-*ary* heap is a binary heap, but (with one possible exception) non-leaf nodes have *d* children instead of 2 children.

1. [2 pts] How would you represent a *d-*ary heap in an array?
2. [3 pts] What is the height of a *d*-ary heap of *n* elements in terms of *n* and *d*?
3. [10 pts] Give an efficient implementation of EXTRACT-MAX in a *d*-ary max-heap. Analyze its running time in terms of *d* and *n*.
4. [10 pts] Give an efficient implementation of INSERT in a *d*-ary max-heap. Analyze its running time in terms of *d* and *n*.
5. [10 pts] Give an efficient implementation of INCREASE-KEY(*A,i,k*), which flags an error if , but otherwise sets and then updates the *d*-ary max-heap structure appropriately. Analyze its running time in terms of *d* and *n*.

**Part 4: Quicksort with equal element values (25 points)**

The analysis of the expected running time of randomized quicksort in the lecture assumes that all element values are distinct.

1. [5 pts] Suppose that all element values are equal. What would be the randomized quicksort's running time in this case?
2. [10 pts] The PARTITION procedure returns an index *q* such that each element of is less than or equal to and each element of is greater than . Modify the PARITION procedure to produce a procedure which permutes the elements of and returns two indices *q* and *t,* where , such that

* all elements of are equal,
* each element of is less than and
* each element of is greater than
* procedure should take time.

1. [10 pts] Modify the procedure to call and name the new procedure Then modify the QUICKSORT procedure to produce a procedure that calls and recurses only on partitions of elements not known to be equal to each other.