Name: □

ID: □

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 7a (14 points) **Fall 2019, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solution:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Gradescope if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.

Name:

ID:

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 7a (14 points) **Fall 2019, CU-Boulder**

1. (1 pt) Provide a one-sentence description of each of the components of a divide and conquer algorithm.

   Solution.

2. (3 pts) Use the array $A = [2, 5, 1, 6, 7, 9, 3]$ for the following questions

   (a) (1 pt) What is the value of the pivot in the call $partition(A, 0, 6)$?
   Solution.

   (b) (1 pt) What is the index of that pivot value at the end of that call to $partition()$?
   Solution.

   (c) (1 pt) On the next recursive call to Quicksort, what sub-array does $partition()$ evaluate?

   Solution.

3. (4 pts) Draw the tree of recursive calls that Quicksort makes to sort the list
   E, X, A, M, P, L, E in alphabetical order. Use the last element in the sub-list in each recursive call as the pivot.

   Solution.

4. (6 pts) You are given a collection of n bottles of different widths and n lids of different widths and you need to find which lid goes with which bottle. You can compare a lid to a bottle, from which you can determine if the lid is larger than the bottle, smaller than the bottle, or the correct size. However, there is no way to compare the bottles or the lids directly to each other, i.e. you can't compare lids to lids or bottles to bottles. Design an algorithm for this problem with an average-case efficiency of $\Theta(nlgn)$

   Solution.