CS 2501 Exam 2

Name _____

For this exam, you should answer each question and compile all of your responses into a **pdf** document. This pdf will be uploaded to Gradescope before the deadline. You have 24 hours to complete this exam. The deadline is Wednesday (4/22) at 5pm Eastern Time.

There are 4 pages to this exam.

This exam is open textbook, notes, calculator, etc. However, it is **CLOSED** friends, TAs, instructor, etc. Please post on Piazza or email course staff if you have clarification questions on the exam. Good luck!

In theory, there is no difference between theory and practice.
But, in practice, there is.

Page 2: Recurrence Relations

1. [4 points] Use the master theorem to solve the following recurrence relation $T(n) = 4T(\frac{n}{2}) + n^2$. Make sure to state the case of the Master Theorem you are using and explain why that case applies. If the master theorem cannot be applied, then explain why.

2. [4 points] Use the master theorem to solve the following recurrence relation $T(n) = 2T(\frac{n}{4}) + n^{0.51}$. Make sure to state the case of the Master Theorem you are using and explain why that case applies. If the master theorem cannot be applied, then explain why.

3. [4 points] Use the *substitution method* to show that $T(n) = T(n-1) + 1 \in O(n)$. We are using Big-Oh here because I only want you to show the upper bound.

Page 3: Divide and Conquer

Suppose you are part of a social group that enjoys watching extremely *mediocre movies*. A *mediocre movie* is one that is very close to 50th percentile if you were to sort the quality of all movies. Your group wants to watch the most *mediocre movie* possible. To do this, you are able to find and sort movie ratings across two popular rating services (IMDB and Rotten Tomatoes). When you download these they each come sorted by increasing ranking (one list for each service). Unfortunately, every movie is in one of the two lists, but no movie is in both. Given the two separate, sorted lists of n movies, can you find the most *mediocre movie* among them to propose your group watch next? In other words, can you find the movie that would be ranked at exactly rank n among the $2n$ movies? (*by rank n, we mean taking the $2n$ total movies, you would produce the movie at index n of a hypothetical collectively sorted list, indexing from 1. For example, if there are 4 movies in each list, you would return the fourth worst movie where the worst movie is the movie at index 1*).

4. [4 points] First, provide an algorithm that solves this in $\Theta(n)$ time.

5. [8 points] Now, provide an algorithm that solves this problem in $\Theta(log(n))$ time.

Page 4: Dynamic Programming

For the next few questions, you will use *dynamic programming* to solve the *productivity hiring problem*. Suppose you own a company that hires workers at varying numbers of work hours and you have learned how productive workers can be at various work hours. Given a total number of hours you can hire work for, how many people should you hire and at how many hours each such that productivity is maximized?

More formally, you are given as input an integer h, the total number of work hours available, and an array P where $P[i]$ defines the productivity of a worker working i hours. An example for $h = 8$ is shown below:

i	0	1	2	3	4	5	6	7	8
$P[i]$	0	5	9	1	10	15	20	19	6

The solution to this instance is to hire four workers for 2 hours each, leading to a total productivity of $4 * 9 = 36$. Notice though that solutions may be made up of many workers working different numbers of hours each.

6. [2 points] Suppose we decide to define our subproblems as $S(i)$ being the optimal productivity possible by only allocating i total hours. Given this information, what is the base case(s)?

7. [8 points] Now define a general solution to $S(i)$ in terms of smaller subproblems.

8. [2 points] State the running time of this recurrence implemented with *dynamic programming*?