Quiz 1

- Do not open this quiz booklet until you are directed to do so.
- This quiz ends at 3:55 P.M.
- When the quiz begins, write your name on the top of every page in this quiz booklet, because the pages will be separated for grading.
- Write your solutions in the space provided. If you need more space, write on the *back* of the sheet containing the problem. Do not put part of the answer to one problem on the back of the sheet for another problem.
- Plan your time wisely. Do not spend too much time on any one problem. Read through all of them first and attack them in the order that allows you to make the most progress.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- When describing an algorithm, describe the main idea in English. Use pseudocode only to the extent that it helps clarify the main ideas.
- Good luck!

Problem	Points	Grade
1	20	
2	32	
3	22	
4	16	
Total	90	

Name: ______________________________

Please circle your TA's name and recitation:

Rachel   Brian   Jon   Josh   Yoav

10am   11am   12pm   1pm   2pm

Problem 1. Recurrences **[20 points]**

Solve the following recurrences (provide only the $\Theta()$ bounds). You can assume $T(n) = 1$ for n smaller than some constant in all cases. You *do not* have to provide justifications, just write the solutions.

- $T(n) = T(n/7) + 1$

- $T(n) = 3T(n/3) + n$

- $T(n) = 5T(n/5) + n \log n$

- $T(n) = 10T(n/3) + n^{1.1}$

Problem 2. True or False, and Justify **[32 points]** (8 parts)

Circle **T** or **F** for each of the following statements to indicate whether the statement is true or false, respectively. If the statement is correct, briefly state why. If the statement is wrong, explain why. Your justification is worth more points than your true-or-false designation.

T F The solution to the recurrence

$$T(n) = T(n/3) + T(n/6) + n^{\sqrt{\log n}}$$

is $T(n) = \Theta(n^{\sqrt{\log n}})$ (assume $T(n) = 1$ for n smaller than some constant c).

T F Radix sort works in linear time **only if** the elements to sort are integers in the range $\{1 \ldots cn\}$, for some $c = O(1)$.

T F There exists a comparison-based sorting algorithm that can sort any 6-element array using at most 9 comparisons.

T F Consider an implementation of Quicksort which always chooses the partition element x to be equal to $A[3]$, where $A[1 \ldots n]$ is the array to partition. This implementation of Quicksort runs in $O(n \log n)$ time in the worst case.

T F Computing the most frequently occuring element in an array $A[1 \dots n]$ can be done in $O(n \log n)$ time.

T F Consider a set S of n two-dimensional points $(x_1, y_1), \dots (x_n, y_n)$, n even. Assume that all coordinates x_i, y_j are different. A vertical line L is a *separator* for S if the number of points of on the left of L is equal to the number of points on the right of L.

A separator for S can be computed in $O(n)$ time.

T F Consider a n-digit decimal number $x_1 x_2 \cdots x_n$ (i.e. each x_i is between 0 and 9). Is the hash function $f(x) = \sum_{i=1}^{n} a_i x_i \pmod 7$ universal where the a_i's are independently and uniformly selected in $\{0, 1, 2, 3, 4, 5, 6\}$?

T F Is your name written on every page of this booklet? (Yes, we will check it.)

Problem 3. Anagram pattern matching **[22 points]**

Assume you are given a *text* array $T[1 \ldots n]$ containing letters from the standard Latin alphabet. In other words, $T[i] \in \{a, b, \ldots z\}$ for $i = 1 \ldots n$. In addition, you are given a *pattern* array $P[1 \ldots m]$, $m < n$, which also contains letters from the Latin alphabet. For any sub-array $T_i^m = T[i \ldots i + m - 1]$ of T, we say that T_i^m is an *anagram* of P if there is a way of permuting symbols in T_i^m so that the resulting array is equal to P.

(a) (8 points) Give an algorithm that, given an index i (and m), determines whether T_i^m is an *anagram* of P. Try to give an algorithm that is as efficient as possible. However, partial credit will also be given for less efficient solutions.

(b) (14 points) Design an algorithm, which given T and P as an input, reports all i's such that T_i^m is an anagram of P. Ideally, your algorithm should run in $O(n+m)$ time. However, partial credit will also be given for less efficient solutions.

Problem 4. Mode finding **[16 points]**

Assume that you are given an array $A[1 \dots n]$ of distinct numbers. You are told that the sequence of numbers in the array is *unimodal*, i.e., there is an index i such that the sequence $A[1 \dots i]$ is increasing (i.e. $A[j] < A[j+1]$ for $1 \leq j < i - 1$) and the sequence $A[i \dots n]$ is decreasing. The index i is called the *mode* of A.

Show that the mode of A can be found in $O(\log n)$ time.