Quiz 1

- Do not open this quiz booklet until directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on the top of every page of this quiz booklet. Circle your recitation at the bottom of this page.
- You have 120 minutes to earn a maximum of 120 points. Do not spend too much time on any one problem. Read them all first, and attack them in the order that allows you to make the most progress.
- **You are allowed one double-sided letter-sized sheet with your own notes**. No calculators, cell phones, or other programmable or communication devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the scratch pages at the end of the exam, and refer to the scratch pages in the solution space provided. Pages will be scanned and separated for grading.
- Do not waste time and paper rederiving facts that we have studied. Simply cite them.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required. But be sure to prove the required bound on running time.
- **Pay close attention to the instructions for each problem**. Depending on the problem, partial credit may be awarded for incomplete answers.

Problem	Parts	Points	Grade	Grader
0: Name	0	2		
1: Recurring Recurrences	2	10		
2: Theoretical Herpetology	2	10		
3: Theoretical Arborist	2	20		
4: Expect the Unexpected	2	6		
5: Firehose Sorting	1	10		
6: Peak Speed	1	11		
7: Resorting to Re-sorting	1	10		
8: Se7en Keys	3	26		
9: Smooth Operator	1	15		
Total		120		

Name: _____

Circle your recitation:	R01 Skanda Koppula	R02 Ludwig Schmidt	R03 Parker Zhao	R04 Gregory Hui	R05 Yonadav Shavit	R06 Atalay Ileri	R07 Anton Anastasov
	R08 Akshay Ravikumar	R09 Aradhana Sinha	R10 Ray Hua Wu	R11 Daniel Zuo	R12 Themistoklis Gouleakis	R13 Adam Hesterberg	R14 Ali Vakilian

Problem 0. [2 points] **What is Your Name?** (2 parts)

(a) [1 point] Flip back to the cover page. Write your name and circle your recitation section.

(b) [1 point] Write your name on top of each page.

Problem 1. [10 points] **Recurring Recurrences** (2 parts)

Solve the following recurrences, expressing your solution using asymptotic Θ notation:

(a) [5 points] $T(n) = 9T(\frac{n}{3}) + \Theta(n \log n)$

(b) [5 points] $T(n) = T(\frac{n}{2}) + \Theta(\log n)$

Problem 2. [10 points] **Theoretical Herpetology** (2 parts)

Analyze the **worst-case time complexity** of each of the following Python functions, expressing your answer using asymptotic Θ notation.

(a) [5 points] Assume that A is a list of n integers.

```
def double(A):
  C = []
  for x in A:
    C.append(x)
  for x in A:
    C.append(None)
  return C
```

(b) [5 points] Assume that A and B are both lists, each with n integers.

```
def all_a_in_b(A, B):
  for x in A:
    if x in B:
      continue
    else:
      return False
  return True
```

Problem 3. [20 points] **Theoretical Arborist** (2 parts)

You are given a rooted binary tree T (where one node r is given as the root, and each node stores a key, a left pointer, a right pointer, and a parent). You know that the nodes of T store n distinct keys, but you are not sure whether T has useful structure.

(a) [10 points] Give an $O(n)$-time algorithm that checks whether T represents a max-heap. (Your running time should be $O(n)$ for any height of the tree.)

(b) [10 points] Give an $O(n)$-time algorithm that checks whether T is a binary search tree. (Your running time should be $O(n)$ for any height of the tree.)

Problem 4. [6 points] **Expect the Unexpected** (2 parts)

Suppose you have a hash table of size m, storing n keys, where $m = \Theta(n)$, and collisions are resolved by chaining. Assume the simple uniform hashing assumption.

(a) [3 points] What is the **expected** running time of SEARCH(x)?

(b) [3 points] What is the **worst-case** running time of SEARCH(x)?

Problem 5. [10 points] **Firehose Sorting** (1 part)

Assume for simplicity that all MIT class numbers are of the form $x.y$ where $1 \leq x \leq 24$ and y comprises between 2 and 4 decimal digits. Give an $O(n)$-time algorithm to sort n class numbers (possibly including duplicates).

The desired order is the usual real-number ordering with a decimal point; for example: 6.004, 6.006, 6.008, 6.01, 6.02, 6.034, 6.042, 6.854, 18.03, 18.06, 18.100.

Problem 6. [11 points] **Peak Speed** (1 part)

Recall that a ***peak*** (local maximum) in a 1D array $A[0\,..\,n-1]$ is an index i, $0 \leq i < n$, such that $A[i-1] \leq A[i] \geq A[i+1]$, where we imagine $A[-1] = A[n] = -\infty$.

Prove an $\Omega(\lg n)$ lower bound for finding a peak in an (unsorted) array $A[0\,..\,n-1]$, in the comparison model.

Hint: Consider arrays with only one peak.

Problem 7. [10 points] **Resorting to Re-sorting** (1 part)

After carefully sorting your array $A[0\,..\,n-1]$, teenage hackers have modified 10 values in your array. Unfortunately, you do not know which 10 values have been modified. Lacking any backups, you can't restore the original data, but you'd still like the data to be sorted.

Give an algorithm to sort A in $O(n)$ time. Your algorithm can only compare values, i.e., it must work within the comparison model.

Problem 8. [26 points] **Se7en Keys** (3 parts)

(a) [10 points] If the numbers 1, 2, 3, 4, 5, 6, and 7 are inserted, in that order, into an initially empty AVL tree, how many total rotations are performed? To ensure partial credit, show your work, circling the number of rotations performed by each insertion. Draw two circles around the total number of rotations.

(b) [6 points] In what order can you insert the numbers 1, 2, 3, 4, 5, 6, and 7 into an initially empty AVL tree to minimize the total number of rotations performed?

(c) [10 points] Give an AVL tree and an element in it such that deleting that element causes at least 3 rotations.

Problem 9. [15 points] **Smooth Operator** (1 part)

You have an array $A[0\,..\,n-1]$ of integers with the guarantee that each value $A[i]$ is within ± 1 of the previous value $A[i-1]$, that is, $|A[i] - A[i-1]| \leq 1$ for all $0 < i < n$.

Give an $O(\log n)$-time algorithm to search in A for a given integer x with the property that $A[0] \leq x \leq A[n-1]$. Your algorithm should **always** return an index j such that $A[j] = x$.

SCRATCH PAPER 1. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to refer to the scratch paper next to the question itself.

SCRATCH PAPER 2. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to refer to the scratch paper next to the question itself.