# 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 every page of this quiz booklet.
- You have 120 minutes to earn 120 points. Do not spend too much time on any one problem. Read them all through first, and attack them in the order that allows you to make the most progress.
- This quiz booklet contains 10 pages, including this one. Two extra sheets of scratch paper are attached. Please detach them before turning in your quiz at the end of the exam period.
- This quiz is closed book. You may use one $8\frac{1}{2}'' \times 11''$ or A4 crib sheet (both sides). No calculators or programmable devices are permitted. No cell phones or other communications devices are permitted.
- 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, since the pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.
- 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.
- Good luck!

| Problem | Parts | Points | Grade | Grader |
|---|---|---|---|---|
| 1 | 8 | 10 | | |
| 2 | 5 | 15 | | |
| 3 | 3 | 10 | | |
| 4 | 4 | 10 | | |
| 5 | 3 | 25 | | |
| 6 | 3 | 25 | | |
| 7 | 3 | 25 | | |
| Total | | 120 | | |

Name: ____________________

| Recitation: | Christina | Christina | Jayant | Jayant | Jason | Matthew |
|---|---|---|---|---|---|---|
| | **10 AM** | **11 AM** | **12 PM** | **1 PM** | **2 PM** | **3 PM** |

**Problem 1. Asymptotic growth** [10 points]

For each pair of functions $f(n)$ and $g(n)$ given below:

- Write $\Theta$ in the box if $f(n) = \Theta(g(n))$
- Write $O$ in the box if $f(n) = O(g(n))$
- Write $\Omega$ in the box if $f(n) = \Omega(g(n))$
- Write X in the box if none of these relations holds

If more than one such relation holds, write only the strongest one. No explanation needed. No partial credit.

| $O$, $\Theta$, $\Omega$ or X | $f(n)$ | $g(n)$ |
|---|---|---|
| | $n^2$ | $n^3$ |
| | $n \lg n$ | $n$ |
| | $1$ | $2 + \sin n$ |
| | $3^n$ | $2^n$ |
| | $4^{n+4}$ | $2^{2n+2}$ |
| | $n \lg n$ | $n^{101/100}$ |
| | $\lg \sqrt{10n}$ | $\lg n^3$ |
| | $n!$ | $(n+1)!$ |

**Problem 2. Miscellaneous True/False** [15 points] (5 parts)

For each of the following questions, circle either T (True) or F (False). **Explain your choice.** (No credit if no explanation given.)

**(a) T F** A hash table guarantees constant lookup time.
*Explain:*

**(b) T F** A non-uniform hash function is expected to produce worse performance for a hash table than a uniform hash function.
*Explain:*

**(c) T F** If every node in a binary tree has either 0 or 2 children, then the height of the tree is $\Theta(\lg n)$.
*Explain:*

**(d) T F** A heap $A$ has each key randomly increased or decreased by 1. The random choices are independent. We can restore the heap property on $A$ in linear time.
*Explain:*

**(e) T F** An AVL tree is balanced, therefore a median of all elements in the tree is always at the root or one of its two children.
*Explain:*

**Problem 3. Sorting short answer** [10 points] (3 parts)

**(a)** What is the worst-case running time of insertion sort? How would you order the elements in the input array to achieve the worst case?

**(b)** Name a sorting algorithm that operates in-place and in $\Theta(n \log n)$ time.

**(c)** Write down the recurrence relation for the running time of merge sort. (You don't need to solve it.)

**Problem 4. Hashing** [10 points] (4 parts)

Give a hash table that uses chaining to handle collisions, how would using sorted python lists in place of unsorted chains affect the following run times? Explain the circumstances of each of the four cases and **justify your choice**.

**(a)** Inserting an element (best case) using unsorted chains is
**Slower / Neither Slower Nor Faster / Faster** than using sorted python lists.

**(b)** Inserting an element (worst case) using unsorted chains is
**Slower / Neither Slower Nor Faster / Faster** than a sorted python lists.

**(c)** Finding an element (best case) using unsorted chains is
**Slower / Neither Slower Nor Faster / Faster** than using sorted python lists.

**(d)** Finding an element (worst case) using unsorted chains is
**Slower / Neither Slower Nor Faster / Faster** than using sorted python lists.

**Problem 5. Sporadically-Rebalanced Trees** [25 points] (3 parts)

Ben Bitdiddle has invented a new kind of data structure, which he calls a sporadically-rebalanced tree (SRT). Ben's tree is a binary search tree, with a twist: every time the size of the SRT doubles, it calls the REBALANCE procedure. That is, REBALANCE is called every time the SRT contains $n = 2^k$ nodes, where $k$ is a natural number. REBALANCE rebalances the tree such that the height of an $n$ element tree is $\Theta(\log n)$.

**(a)** Does Ben's scheme preserve the $\Theta(\log n)$ height of an $n$ element tree? If so, explain why. If not, what is the worst-case height of an $n$ element tree, in $\Theta$ notation?

**(b)** Argue briefly that rebalancing an $n$-node SRT can be done in $\Theta(n)$ time.

**(c)** What is the worst-case running time (in $\Theta$ notation) for a sequence of $n$ INSERT operations in Ben's scheme? Assume the SRT is initially empty.

**Problem 6. Ben's List Matcher** [25 points] (3 parts)

Ben got tired of dealing with SRTs and decides to build a matcher for lists of numbers. Ben's matcher takes two lists of numbers and decides the lists are equal if and only if they contain exactly the same set of numbers. For example, `[1, 11, 13, 27]` is equal to `[11, 1, 27, 13]`, but not equal to `[1, 11, 13, 27, 2]` or `[1, 11, 27]`. Assume the lists do not contain multiple instances of the same number.

Ben implements his function in Python as follows:

```
def listcmp(list1, list2):
    for num in list1:
        if num not in list2:
            return False
        else:
            # Remove the first occurrence of num from list2
            list2.remove(num)
    # Return True iff list2 is now empty
    return list2 == []
```

(The python function `L.remove(x)` for a list `L` is a mutator that removes the first occurrence of `x` from `L`. Its implementation scans the list `L` from the beginning until it finds `x`.)

**(a)** Let $n$ be the length of `list1` and $m$ be the length of `list2`. What is the worst-case running time of Ben's implementation? Justify your answer.

**(b)** Louis Reasoner suggests Ben implement his function with an AVL tree. In Louis' implementation, the elements of `list1` are inserted into an AVL tree, then the elements of `list2` are searched for and deleted if found from the AVL tree. The procedure returns `True` if and only if every element of `list2` was found in the tree, and the tree is empty after the elements of `list2` are removed.

What is the worst-case running time of Louis' implementation? Justify your answer.

**(c)** Assume the elements of `list1` and `list2` are all in $\{1, 2, ..., k\}$. Describe (in English) a solution with a $\Theta(k + n)$ worst-case running time.

**Problem 7. Dynamic Medians** [25 points] (3 parts)

Marianne Midling needs a data structure "DM" for maintaining a set $S$ of numbers, supporting the following operations:

- Create an empty set $S$
- Add a new given number $x$ to $S$
- Return a median of $S$. (Note that if $S$ has even size, there are two medians; either may be returned. A median of a set of $n$ distinct elements is larger than or equal to exactly $\lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$ elements.)

(Assume no duplicates are added to $S$.)

Marianne proposes to implement this "dynamic median" data structure DM using a max-heap $A$ and a min-heap $B$, such that every element in $A$ is less than every element in $B$, and the size of $A$ equals the size of $B$, or is one less.

To return a median of $S$, she proposes to return the minimum element of $B$.

**(a)** Argue that this is correct (i.e., that a median is returned).

**(b)** Explain how to add a new number $y$ to this data structure, while maintaining the relevant properties.

**(c)** How much time does your solution to (b) take, in the worst case?

SCRATCH PAPER

SCRATCH PAPER