CS 240: Data Structures and Data Management **Fall 2025**

Midterm Practice Problem

Note: This is a sample of problems designed to help prepare for the midterm exam. These problems do *not* encompass the entire coverage of the exam, and should not be used as a reference for its content.

1. **True or False**
   (a) The midterm for this course is on October 9th at 4:30pm to 6:20pm.
   (b) If $T_1(n) \in \Omega(f(n))$ and $T_2(n) \in O(g(n))$, then $\frac{T_1(n)}{T_2(n)} \in \Omega(\frac{f(n)}{g(n)})$
   (c) If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = e^{42}$, then $f(n) \in \Theta(g(n))$
   (d) If $f(n) \in \Theta(g(n))$, then $\lim_{n\to\infty} \frac{f(n)}{g(n)} = L$ where $0 < L < \infty$
   (e) All heaps satisfy AVL tree's height-balance requirement
   (f) A binary search tree with n leaves must have height in O(n)
   (g) The runtime for insert and delete in heaps is always $\Theta(\log n)$
   (h) The average-case and expected-case run-time of an algorithm must always be the same
   (i) It is possible for a sorting algorithm to have a runtime in $o(n \log n)$ on an input of size n
2. **Order Notation and Recurrence Relation**
   (a) Show that $3n^2 - 8n + 2 \in \Theta(n^2)$ from first principles.
   (b) Prove from first principle that $14n + 22$ is $o(nlogn)$
   (c) Prove from first principle that $n \in \omega(2^{\sqrt{\log n}})$
   (d) Given $T(1) = 1$, resolve $T(n) = T(\frac{3n}{4}) + n$ by providing a Θ bound.
   (e) Disprove the following statement - if $f(n) \in o(n \log n)$, then $f(n) \in O(n)$
3. **Pseudo-code Analysis**
   Analyze following pieces of pseudo-code and give a tight bound on the running time as a function of n.
   (a) Analyze the following piece of pseudo-code and give a tight (Θ) bound on the running time as a function of n.

```
i = 2
x = 0
while (i < n):
    for j = 1 to n:
        for k = 1 to j:
            x = x + 1
    i = i * i
```

(b) Give a tight big-O bound for the expected runtime of the following algorithm.

```
ArrayAlg(A, n, k)
    // n = A.size()
    // A is a permutation of [0, ..., n-1]
    // k is in the set {0, ..., n-1}
    i = random(n)
    if A[i] == k then
        return i
    for j = 0 to n-1
        print("a")
    return ArrayAlg(A, n, k)
```

(c) Let A and B be two bit-strings of length n (modelled here as arrays where each entry is 0 or 1). A `string-compare` tests whether A is smaller, larger, or the same as B and works as follows:

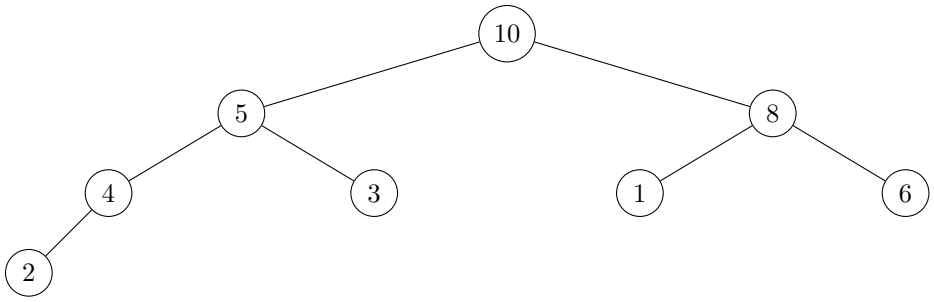
```
str-cmp(A, B, n)
    for i = 0; i < n: i++ do
        if (A[i] < B[i]) then return "A is smaller"
        if (A[i] > B[i]) then return "A is bigger"
    return "They are equal"
```

Show that the average-case run-time of `str-cmp` is in $O(1)$. You may use without proof that $\sum_{i \geq 0} \frac{i}{2^i} \in O(1)$.

(d) Give the best-case and expected running time for the following function. You can assume that the Shuffle operation requires $O(n)$ time and the array A contains no duplicates
Note: the *Shuffle*() function produces each permutation equally likely.

```
MonkeySort(A):
    // Input: Array A of size n
    // Output: None (A is sorted in-place)
    shuffle(A)
    if A is sorted then
        return A
    else do
        MonkeySort(A)
```

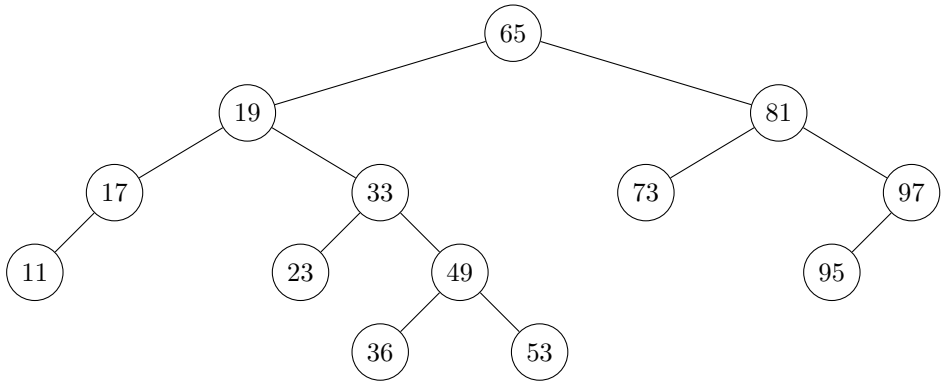
4. **Heap Operations**
Insert 9 into the following max-heap and then delete-max



5. **Heap Merging**
   Describe an algorithm for merging 2 binary heaps. That is, given 2 heaps A and B, return a new heap C containing all of the elements of A and B.

6. **Priority-Queue**
   Given a family k sorted arrays $A_1, \ldots, A_k$, where the combination of the k arrays has n elements, give an $O(n \log k)$ time algorithm that produces a single sorted array containing all n elements. Hint: use a priority queue.

7. **Basics of AVL Tree**
   Consider following AVL tree.



   (a) Fill out height factor of each node. For example, node 33 will have height factor of 2.

   (b) Fill out balance factor of each node. For example, node 33 will have height factor of 1.

8. **Lower bound finding**
   Michael thinks he has discovered a new realization of priority-queues, which is comparison based and performs insert and delete-max in $O(\log \log n)$ time. Explain why this realization cannot be correct. Hint: do not tackle by talking about his implementation of priority queue.

9. **Epsilon**
   Let $0 < \epsilon < 1$. Suppose that we have an array A of n items such that the first $n - n^{\epsilon}$ items are sorted. Describe an $O(n)$ time algorithm to sort A.

10. **Radix Sort**
    Perform MSD and LSD on the following array of decimal integers using $R = 2$:

$$A = [100, 64, 127, 1, 17, 18, 67]$$

11. **Numbers in Range**
    We have an array A of n non-negative integers such that each integer is less than k. Give an $O(n + k)$ time preprocessing algorithm such that queries of the form "how many integers are there in A that are in the range $[a, b]$?" can be answered in $O(1)$ time. Note that a and b are not fixed; they are parameters given to the query algorithm.