Write your answers in the separate answer booklet.

1. **Multiple Choice:** Each question below has one of the following answers.

   (a) $\Theta(1)$ (b) $\Theta(\log n)$ (c) $\Theta(n)$ (d) $\Theta(n \log n)$ (e) $\Theta(n^2)$

   For each question, write the letter that corresponds to your answer. You do not need to justify your answers. Each correct answer earns you 1 point, but each incorrect answer *costs* you $\frac{1}{2}$ point. You cannot score below zero.

   (a) What is $\sum_{i=1}^{n} H_i$?

   (b) What is $\sum_{i=1}^{\lg n} 2^i$?

   (c) How many digits do you need to write $n!$ in decimal?

   (d) What is the solution of the recurrence $T(n) = 16T(n/4) + n$?

   (e) What is the solution of the recurrence $T(n) = T(n-2) + \lg n$?

   (f) What is the solution of the recurrence $T(n) = 4T\left(\left\lceil \frac{n+51}{4} \right\rceil - \sqrt{n}\right) + 17n - 2^{8\log^*(n^2)} + 6$?

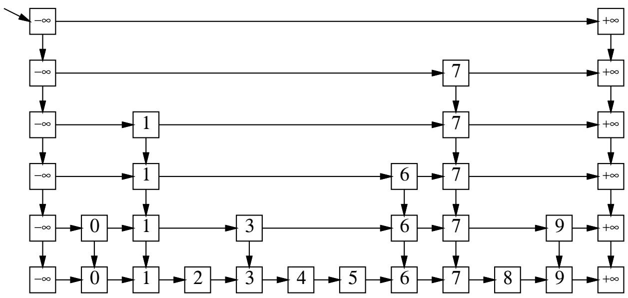
   (g) What is the worst-case running time of randomized quicksort?

   (h) The expected time for inserting one item into a treap is $O(\log n)$. What is the worst-case time for a sequence of n insertions into an initially empty treap?

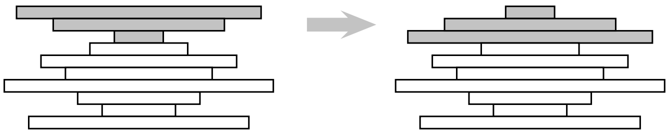
   (i) The amortized time for inserting one item into an n-node splay tree is $O(\log n)$. What is the worst-case time for a sequence of n insertions into an initially empty splay tree?

   (j) In the worst case, how long does it take to solve the traveling salesman problem for $10,000,000,000,000,000$ cities?

2. What is the *exact* expected number of nodes in a skip list storing n keys, *not* counting the sentinel nodes at the beginning and end of each level? Justify your answer. A correct $\Theta()$ bound (with justification) is worth 5 points.



3. Suppose we have a stack of n pancakes of all different sizes. We want to sort the pancakes so that smaller pancakes are on top of larger pancakes. The only operation we can perform is a *flip* — insert a spatula under the top k pancakes, for some k between 1 and n, turn them all over, and put them back on top of the stack.



Flipping the top three pancakes

   (a) (3 pts) Describe an algorithm to sort an arbitrary stack of n pancakes using flips.

   (b) (3 pts) Prove that your algorithm is correct.

   (c) (2 pts) Exactly how many flips does your sorting algorithm perform in the worst case? A correct $\Theta()$ bound is worth one point.

   (d) (2 pts) Suppose one side of each pancake is burned. Exactly how many flips do you need to sort the pancakes, so that the burned side of every pancake is on the bottom? A correct $\Theta()$ bound is worth one point.

4. Suppose we want to maintain a set of values in a data structure subject to the following operations:

   - $\textsc{Insert}(x)$: Add x to the set (if it isn't already there).
   - $\textsc{DeleteRange}(a, b)$: Delete every element x in the range $a \le x \le b$. For example, if the set was $\{1, 5, 3, 4, 8\}$, then $\textsc{DeleteRange}(4, 6)$ would change the set to $\{1, 3, 8\}$.

   Describe and analyze a data structure that supports these operations, such that the amortized cost of either operation is $O(\log n)$. *[Hint: Use a data structure you saw in class. If you use the same* Insert *algorithm, just say so—you don't need to describe it again in your answer.]*

5. *[1-unit grad students must answer this question.]*

   A *shuffle* of two strings X and Y is formed by interspersing the characters into a new string, keeping the characters of X and Y in the same order. For example, 'bananaananas' is a shuffle of 'banana' and 'ananas' in several different ways.

   $$\text{banana}_{\text{ananas}} \qquad \text{ban}_{\text{ana}}{}^{\text{ana}}{}_{\text{nas}} \qquad \text{b}_{\text{an}}{}^{\text{an}}{}_{\text{a}}{}^{\text{a}}{}_{\text{na}}{}^{\text{na}}{}_{\text{s}}$$

   The strings 'prodgyrnamammiincg' and 'dyprongarmammicing' are both shuffles of 'dynamic' and 'programming':

   $$\text{pro}^{\text{d}}\text{g}^{\text{y}}\text{r}^{\text{nam}}\text{ammi}^{\text{i}}\text{n}^{\text{c}}\text{g} \qquad {}^{\text{dy}}\text{pro}^{\text{n}}\text{g}^{\text{a}}\text{r}^{\text{m}}\text{amm}^{\text{ic}}\text{ing}$$

   Given three strings $A[1..m]$, $B[1..n]$, and $C[1..m+n]$, describe and analyze an algorithm to determine whether C is a shuffle of A and B. For full credit, your algorithm should run in $\Theta(mn)$ time.