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: $\Theta(n^2)$ X: I don't know.

   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. Each X earns you $\frac{1}{4}$ point. **Each incorrect answer *costs* you $\frac{1}{2}$ point.** Your total score will be rounded **down** to an integer. Negative scores will be rounded up to zero.

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

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

   (c) How many bits do you need to write 10^n in binary?

   (d) What is the solution of the recurrence $T(n) = 9T(n/3) + n$?

   (e) What is the solution of the recurrence $T(n) = T(n-2) + \frac{3}{n}$?

   (f) What is the solution of the recurrence $T(n) = 5T\left(\left\lceil \frac{n-17}{25} \right\rceil - \lg\lg n\right) + \pi n + 2^{\sqrt{\log^* n}} - 6$?

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

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

   (i) Suppose StupidAlgorithm produces the correct answer to some problem with probability $1/n$. How many times do we have to run StupidAlgorithm to get the correct answer with high probability?

   (j) Suppose you correctly identify three of the possible answers to this question as obviously wrong. If you choose one of the three remaining answers at random, each with equal probability, what is your expected score for this question?

2. Consider the following algorithm for finding the smallest element in an unsorted array:

```
RandomMin(A[1 .. n]):
    min ← ∞
    for i ← 1 to n in random order
        if A[i] < min
            min ← A[i]   (⋆)
    return min
```

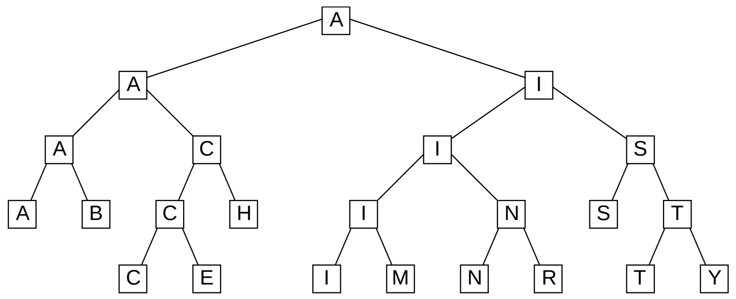
   (a) **[1 point]** In the worst case, how many times does RandomMin execute line $(\star)$?

   (b) **[3 points]** What is the probability that line $(\star)$ is executed during the nth iteration of the for loop?

   (c) **[6 points]** What is the exact expected number of executions of line $(\star)$? (A correct $\Theta()$ bound is worth 4 points.)

3. Algorithms and data structures were developed millions of years ago by the Martians, but not quite in the same way as the recent development here on Earth. Intelligent life evolved independently on Mars' two moons, Phobos and Deimos.[1] When the two races finally met on the surface of Mars, after thousands of Phobos-orbits[2] of separate philosophical, cultural, religious, and scientific development, their disagreements over the proper structure of binary search trees led to a bloody (or more accurately, ichorous) war, ultimately leading to the destruction of all Martian life.

   A *Phobian* binary search tree is a full binary tree that stores a set X of search keys. The root of the tree stores the *smallest* element in X. If X has more than one element, then the left subtree stores all the elements less than some pivot value p, and the right subtree stores everything else. Both subtrees are *nonempty* Phobian binary search trees. The actual pivot value p is *never* stored in the tree.



A Phobian binary search tree for the set $\{M, A, R, T, I, N, B, Y, S, C, H, E\}$.

   (a) **[2 points]** Describe and analyze an algorithm Find(x, T) that returns True if x is stored in the Phobian binary search tree T, and False otherwise.

   (b) **[2 points]** Show how to perform a rotation in a Phobian binary search tree in $O(1)$ time.

   (c) **[6 points]** A *Deimoid* binary search tree is almost exactly the same as its Phobian counterpart, except that the *largest* element is stored at the root, and both subtrees are Deimoid binary search trees. Describe and analyze an algorithm to transform an n-node Phobian binary search tree into a Deimoid binary search tree in $O(n)$ time, using as little additional space as possible.

4. Suppose we are given an array $A[1..n]$ with the special property that $A[1] \geq A[2]$ and $A[n-1] \leq A[n]$. We say that an element $A[x]$ is a *local minimum* if it is less than or equal to both its neighbors, or more formally, if $A[x-1] \geq A[x]$ and $A[x] \leq A[x+1]$. For example, there are five local minima in the following array:

9	7	7	2	**1**	3	7	5	**4**	7	**3**	**3**	4	8	**6**	9

   We can obviously find a local minimum in $O(n)$ time by scanning through the array. Describe and analyze an algorithm that finds a local minimum in $O(\log n)$ time. *[Hint: With the given boundary conditions, the array* ***must*** *have at least one local minimum. Why?]*

[1]Greek for "fear" and "panic", respectively. Doesn't that make you feel better?

[2]1000 Phobos orbits $\approx$ 1 Earth year

5. *[Graduate students must answer this question.]*

   A *common supersequence* of two strings A and B is a string of minimum total length that includes both the characters of A in order and the characters of B in order. Design and analyze and algorithm to compute the length of the *shortest* common supersequence of two strings $A[1\,..\,m]$ and $B[1\,..\,n]$. For example, if the input strings are `ANTHROHOPOBIOLOGICAL` and `PRETERDIPLOMATICALLY`, your algorithm should output 31, since a shortest common supersequence of those two strings is $\mathtt{\underline{PRE}\overline{AN}\overline{\underline{T}}\overline{H}\underline{E}\overline{\underline{R}}\overline{OHO}\underline{D}\overline{POB}\overline{\underline{I}}\overline{O}\underline{P}\overline{\underline{LO}}\underline{MAT}\overline{G}\overline{\underline{ICAL}}\underline{LY}}$. You do not need to compute an actual supersequence, just its length. For full credit, your algorithm must run in $\Theta(nm)$ time.