22C:44 Midterm 1
October 5, 2000, 2:30-3:45

Notes: (a) All problems carry equal weight. (b) Show all your working. (c) Feel free to consult your notes and books.

1. Solve the recurrence

$$\begin{aligned} T(2) &= 1 \\ T(n) &\leq T(n/10) + T(9n/10) + n, \qquad \text{for all } n \geq 2 \end{aligned}$$

   by using the substitution method with the guess $T(n) \leq cn \lg n$.

2. Analyze the following function to determine its running time as a function of n. Express your answer in Θ-notation.

```
printHello(n) {
     s ← 1;
     for i ← 1 to n do {
          u ← ⌊ n/s ⌋;
          for j ← 1 to u do
               print ``Hello'';
          s ← 2 s;
     }
}
```

3. The input is an array of n elements that contains 10 distinct elements, each appearing $n/10$ times. Modify `Partition` so that `QuickSort` runs in *worst case time* of $O(n \lg n)$. Set up a recurrence for the worst case running time of `QuickSort` and using this prove that the worst case running time of `QuickSort` is indeed $O(n \lg n)$.
   Note: It is acceptable (in fact, preferable) if you describe your modification to `Partition` in words, rather than in code. In any case, you should try to make your main idea clear.

4. In class we studied *binary heaps*, that is, heaps in which each node has 2 children. Binary heaps can be extended to d-ary heaps, that is, heaps in which each node has d children. How would you represent a d-ary heap in an array? If a node is in slot i, which slots would its d children occupy?

5. The problem is to find the median in an array of n elements, given that the median occurs $n/10$ times. For this problem, your friend suggests the following simple, randomized algorithm:

   Step 1 Pick an element x at random, uniformly from among the n elements in the array.

   Step 2 Check whether x is a median. If it is, stop. Otherwise, repeat the process starting with Step 1.

   Analyze this algorithm to determine its expected running time.