Problem 1

Construct an array of 7 integers such that, when quicksort runs on the array, every call to partition splits its input array exactly in half.

Now suppose you have two arrays, A and B, each of length n. Suppose also that, when quicksort runs on A, each call to partition splits its input array exactly in half, and likewise for B. Construct a new array C of size 2n + 1 such that, when quicksort runs on C, each call to partition splits its input array in half.

Problem 2

Solve the following recurrence relations (in all cases, T(1) = 0). You only need to find \( \Theta() \) solutions, not exact solutions.

- \( T(n) = T(\lfloor n/2 \rfloor) + 1 \)
- \( T(n) = 2T(\lfloor n/2 \rfloor) + 1 \)
- \( T(n) = 4T(\lfloor n/2 \rfloor) + 1 \)
- \( T(n) = T(n-1) + 1 \)
- \( T(n) = 2T(n-1) + 1 \)
- \( T(n) = T(\lfloor n/2 \rfloor) + n \)
- \( T(n) = 2T(\lfloor n/2 \rfloor) + n \) (yeah, I know we did one like this in class)
- \( T(n) = 4T(\lfloor n/2 \rfloor) + n \)

Problem 3

Pick three of the recurrences from Problem 2 and carefully prove, via induction, that your answer is correct. For convenience, you may assume that n is a power of two.

Problem 4

Use the partition algorithm to write a function \texttt{find-kth-smallest}(A, n, k) that, given an array, A, of n integers, returns the kth smallest integer in the array. In other words, your algorithm should return the integer that would be in position k after the array is sorted. A trivial solution to this problem is

\begin{verbatim}
procedure find-kth-smallest(A, n, k)
  qsort(A, n)
  return A[k]
\end{verbatim}

This algorithm would have running time \( O(n \log n) \). Your algorithm should be faster.
Example tabbing environment

In case you decide to write your homework in latex, here’s an example of using the tabbing environment to typeset pseudo-code using the commands defined in the source of this latex file.

```
procedure find-kth-smallest(A, n, k)
    qsort(A, n)
    if foo
        hello
        for blah blah blah
            hello
    else
        goodbye
    for blah blah blah
        hello
    return A[k]
```