Homework 5, Basic Algorithms, Spring 2014.
Due date: Tuesday, March 4.

Problems 3.16, 5.38, 5.41, 11.16 from the text.
For problem 5.38 assume we are using the version of Quicksort with no randomization: the pivot item is simply the item in the leftmost location. For Problem 5.41, code for a Merge Sort procedure is given on Page 2. You may use this code as the basis for your algorithm.

5. Consider the following recurrence equations.

\[
T(n) = \begin{cases} 
  f(n) & \text{for } n \leq c \\
  f(n) + \sum_{i=1}^{k(n)} T(n_i) & \text{for } n > c, \text{ where } 0 \leq n_i \leq n-1, \text{ for } 1 \leq i \leq k(n) 
\end{cases}
\]

\[
L(n) = \begin{cases} 
  \frac{g(n)}{C} & \text{for } n \leq c \\
  \frac{g(n)}{C} + \sum_{i=1}^{k(n)} L(n_i) & \text{for } n > c, \text{ where } 0 \leq n_i \leq n-1, \text{ for } 1 \leq i \leq k(n) 
\end{cases}
\]

\[
U(n) = \begin{cases} 
  C \cdot g(n) & \text{for } n \leq c \\
  C \cdot g(n) + \sum_{i=1}^{k(n)} U(n_i) & \text{for } n > c, \text{ where } 0 \leq n_i < n, \text{ for } 1 \leq i \leq k(n) 
\end{cases}
\]

Suppose that \(0 \leq \frac{g(n)}{C} \leq f(n) \leq C \cdot g(n)\) for all \(n \geq 0\). Prove by induction that, for \(n \geq 0\), \(C^2 \cdot L(n) = U(n)\) and that \(L(n) \leq T(n) \leq U(n)\) for \(n \geq 0\). You may assume that \(n\) is an integer (although this is not necessary).

This result formally demonstrates that the solutions to the recurrence equations

\[
T(n) = \begin{cases} 
  f(n) & \text{for } n \leq c \\
  f(n) + \sum_{i=1}^{k(n)} T(n_i) & \text{for } n > c, \text{ where } n_i < n \text{ for } 1 \leq i \leq k(n) 
\end{cases}
\]

and

\[
\overline{T}(n) = \begin{cases} 
  \Theta(f(n)) & \text{for } n \leq c \\
  \Theta(f(n)) + \sum_{i=1}^{k(n)} \overline{T}(n_i) & \text{for } n > c, \text{ where } n_i < n \text{ for } 1 \leq i \leq k(n) 
\end{cases}
\]

are within a constant factor of each other.

6. Consider inserting the items 1, 2, \ldots, \(n\) into an initially empty binary search tree in random order (i.e. all permutations of the \(n\) items are equally likely as the insertion order). Let \(D(n)\) denote the expected sum of the depths of all the items in the binary search tree, averaging over all the permutations. Here, we define the depth of an item to be its distance from the root measured in vertices (so if it is at the root its depth is 1).

Write a recurrence equation for \(D(n)\).

Hint. Consider each of the possible choices for the item at the root. What is the contribution to the expected sum if \(i\) is at the root?

**Honors Section.** Problem 5.35 and 5.40 in addition.
Driver(A, n);
    MergeSort(A, 1, n)
end (* Driver *)

MergeSort(A, i, k) (* MergeSorts A[i : k] *)
    if i < k then do
        mid = \lceil (i + k)/2 \rceil;
        MergeSort(A, i, mid);
        MergeSort(A, mid + 1, k);
        Merge(A, i, mid, k)
    end (* then do *)
end (* MergeSort *)

Merge(A, i, mid, k, B) (* Merges sorted subarrays A[i : mid], A[mid + 1 : k] *)
    local variable B[i : k], lt, rt, nt;
    lt ← i; rt ← mid + 1; nt ← i;
    while lt ≤ mid and rt ≤ k do
        if A[lt].key ≤ A[rt].key
            then do
                B[nt] ← A[lt]; lt ← lt + 1; nt ← nt + 1
            end (* then do *)
        else do
            B[nt] ← A[rt]; rt ← rt + 1; nt ← nt + 1
        end (* else do *)
    end; (* while do *)
    while lt ≤ mid do
        B[nt] ← A[lt]; lt ← lt + 1; nt ← nt + 1;
    end; (* while do *)
    while rt ≤ k do
        B[nt] ← A[rt]; rt ← rt + 1; nt ← nt + 1
    end; (* while do *)
    copy B[i : k] to A[i : k]
end (* Merge *)