1. (10 points) Consider a binary counter with $k$ bits for some positive integer $k$. Suppose that in addition to \textsc{Increment} we also wish to support a \textsc{Decrement} operation, which decreases the value of the counter by 1. Show that a sequence of $n$ operations consisting of the above two operations, done on a counter that is zero initially, could take $\Theta(nk)$ time. (Note that a $\Theta$ bound is called for, not $O$.)

2. (12 points) Suppose that you wish to do two types of operations on an infinite binary counter: \textsc{Increment} and \textsc{Reset}, where the latter resets all bits in the counter to zero. Explain clearly in words (pseudocode is not needed) how such a counter could be implemented so that any sequence of $n$ \textsc{Increment} and \textsc{Reset} operations on an initially-zero counter takes $O(n)$ time.

Use the \textit{accounting} (i.e., \textit{credits}) \textit{method} to do the analysis. State clearly the invariant that you use and the number of credits assigned to each operation. (Assume that it takes constant time to examine or to modify a bit.)

\textit{Hint:} Consider keeping a pointer to the high-order 1-bit.

3. (14 points) Let $Q$ be a queue, with the usual operations \textsc{Enqueue} and \textsc{Dequeue}. We wish to implement $Q$ using two stacks $S_1$ and $S_2$, so that the amortized cost of each queue operation is a constant. (Assume that $Q$ is empty initially.) Describe in words how this can be done and give pseudocode for \textsc{Enqueue} and \textsc{Dequeue}. Analyse your solution using the \textit{accounting} (i.e., \textit{credits}) \textit{method} of amortized analysis. State clearly the invariant that you use and the number of credits assigned to each operation.

You may use the stack operations \textsc{Push} and \textsc{Pop} as black boxes, without writing code for them.

4. (12 points) Let $L$ be a linked list of integers that is subjected to the following operations:

\textsc{Insert}($L, x$): Insert integer $x$ into $L$.
\textsc{Remove}($L$): Remove the largest $|L|/2$ integers from $L$. (For simplicity, assume $|L|/2$ is an integer.)

Assume that $L$ is empty initially and that its integers are always distinct.

Explain clearly in words (pseudocode is not needed) how to do these operations so that the total time for any sequence of $n$ such operations is $O(n)$. Use the \textit{potential method} of amortized analysis. State clearly your potential function.

You may find it useful to use the linear-time selection algorithm as a black box. For convenience, assume that on a set of $m$ items it takes exactly $m$ units of time (i.e., ignore constant factors).
5. (12 points) Ex. 17.4-2, p. 471.

6. (12 points) This problem assumes familiarity with Ch. 6. In that chapter, it is shown that a max-heap, A, on $n$ keys can be built in $O(n)$ time (See Sec. 6.3). Derive this result using the potential method of amortized analysis. State clearly your potential function.

Hint: Try to relate your potential function to some aspect of the maximal max-heaps present just before the next Heapify operation is done. For instance, in Fig. 6.3(d), just before Heapify is done from the node numbered 2 (shown dark gray), the maximal max-heaps are the ones with roots numbered 3, 4, and 5. They are maximal in the sense that none of them is contained in a larger max-heap at that moment.

This problem illustrates two aspects of amortized analysis: i.e., it often leads to simpler proofs (as compared to the proof in Section 6.3) and, furthermore, it “forces” one to really understand how a data structure works.

Note: This is an optional extra credit problem to help you improve your performance in the class, if you so desire. The points you earn on this problem will be added to the score you receive on the other problems that we grade, without changing the overall value of the assignment. So, for instance, suppose that the required part of the homework that we grade is worth (say) 50 points. If you get (say) 45 points on that and 8 points on this optional problem, then your score will be 53/50; thus, it is possible to earn more than 100% on this assignment.