Algorithms
Computer Science 140 & Mathematics 168
Instructor: B. Thom
Fall 2003
Homework 7a
Due on Due 10/17/03 (Friday at 2pm)

1. [**20 points**] **Amortizing your Heap!**

   Recall that, for any node i in a heap, `heapify`(i) runs in at most the height of that subtree rooted at i. This is because heapify percolates i downwards in the tree, swapping it with one of its children until either i becomes a leaf or it maintains a desired ordering with respect to its children. Recall also that a heap can be built as follows:

   ```
   buildheap(A[1,...,i,...,n]) {
     for i = n to 1
       heapify(i)
   }
   ```

   Since the height of the overall tree is $O(\log n)$, a naive analysis of this function's runtime provides $O(n \log n)$. You're now going to use amortization to argue the tighter bound mentioned in class: $\Theta(n)$.

   Here's what you do. At "birth," give every node 3 rubles in which to pay for future work. The goal is then to argue that all of heapify's work can be paid for using this scheme. How? Prove that, for a node of height h, there is always h rubles in the bank.

   Then, use this observation to link back into the desired claim: that `buildheap` runs in $O(n)$ time.

   For what reason can you say this bound is tight, i.e. its $\Theta(n)$?

2. [**10 points**] **Graph Data Structures** This is a short review problem. Briefly, using high-level pseudo code, describe how the following algorithms would work and describe their run-times:

   (a) Identify (i.e. visit) every vertex and edge of a graph stored as an adjacency matrix.
   (b) Identify (i.e. visit) every vertex and edge of a graph stored as an adjacency list.
   (c) Determine an in-degree vector of a graph stored as an adjacency matrix.
   (d) Determine an in-degree vector of a graph stored as an adjacency list.
   (e) Determine an out-degree vector of a graph stored as an adjacency matrix.
   (f) Determine an out-degree vector of a graph stored as an adjacency list.