**Algorithms**
**Computer Science 140 & Mathematics 168**
**Spring 2009**
Homework 5a
Due Thursday, February 19

- Due to the exam given out on Thursday, **no extensions may be used on this assignment.**
- Please remember that during the exam period (Thursday, February 19 at 11 AM through Tuesday, February 24 at 9:35 AM) no discussions on algorithms material is permitted (including grutors, Ran, friends, relatives, pets, etc.). So, if you would like to discuss any material, please do so before the exam begins.

1. **[20 Points] Analyzing Build-Heap!**

   Recall from class that the Heapsort algorithm sorts an array (indexed from 1 to $n$) as follows: Consider the array as representing a binary tree where the children of the node at index $i$ are found at indices $2i$ and $2i + 1$ (unless these indices are out of range).

   We begin by turning this array into a heap: a binary tree in which each node is less than or equal to its descendants. To do so, we invoke a process called Build-Heap that starts at node $i = \lfloor \frac{n}{2} \rfloor$ and works down to 1. For each $i$ in this range, Build-Heap ensures that the tree rooted at $i$ has the property that it is a heap itself! To this end, node $i$ is compared with both of its children. If it is less than or equal to both of its children, we are done with node $i$ and we continue on to node $i - 1$. Otherwise, we swap the value at node $i$ with the smaller of its two children and then repeatedly "percolate" this value down until it reaches its correct location in the heap rooted at node $i$.

   In class, we argued that since the heap has height $O(\log n)$ and there are approximately $n/2$ nodes that are examined in the Build-Heap process, the total running time of Build-Heap is $O(n \log n)$.

   In this problem you will prove that the actual total running time of Build-Heap is $O(n)$ by using an amortization argument. For simplicity, assume that $n = 2^k - 1$ for some $k > 1$. That is, the heap is a complete binary tree in which all of the leaves are at the same level.

In this amortization scheme, each of the $n$ nodes receives 3 rubles at the outset. We claim that this is enough to pay for all of the work incurred by BUILD-HEAP. To help see why, notice that we start the BUILD-HEAP process at index $\lfloor \frac{n}{2} \rfloor$, a node with two leaf children. *Let's call this node "Joe". This is not a proof yet, so giving names to nodes is fun and easy!* Joe does two comparisons and perhaps a swap, for a total of 3 operations. (We are assuming that a comparison counts as one operation and a swap counts as one operation.) However, the tree rooted at Joe has 9 rubles at its disposal: 3 for each of the two leaves and 3 for Joe. We spend at most 3 rubles, leaving 6 rubles in Joe's tree for future work. Next, Joe's sibling, Josette, does the same thing, leaving her tree with 6 rubles for later work. Eventually, we get to Joe and Josette's parent, Pat. Pat has 3 rubles and each of its children, Joe and Josette, have 6 rubles remaining in their own trees. This is a total of 15 rubles. How much work does Pat incur? There are two comparisons (compare Pat to Joe and Josette) and perhaps a swap. Then, in the worst case that Pat is swapped, there are two more comparisons and perhaps a swap for a total of up to 6 operations. This uses 6 of the 15 rubles in this tree, leaving 9 rubles in Pat's tree for later work.

Based on the observations above, give a careful proof by induction that shows that the 3 rubles allocated at each node pays for the entire BUILD-HEAP process, resulting in $O(n)$ running time for BUILD-HEAP. *Hint: You will need to state a precise invariant that is preserved by this accounting scheme. Proving this invariant by induction will imply that the scheme pays for all of the work.*

2. [**10 Points**] **Real-Queue and Helper-Queue Revisited!** On the last assignment, you used an accounting argument to show that any sequence of $n$ operations (chosen from ENQUEUE, DEQUEUE, and FIND-MIN) takes $O(n)$ time. Now, prove this *again* using the potential method.