Problem 1
Recall that in Fibonacci heaps, $\text{DECREASE-KEY}(x, k)$, where $x$ is a node and $k$ is the new key of $x$, is implemented as follows:

1. decrease the key of $x$ to $k$
2. if $x$ is a root, stop; else, $\text{CUT}(x)$: remove $x$ from the child list of its parent $y$ and add $x$ to the root list; if $x$ is marked, unmark $x$
3. $\text{CASCADE-CUT}(y)$: if $y$ is a root, stop; else, if $y$ is unmarked, mark $y$ and stop; otherwise, $\text{CUT}(y)$ and $\text{CASCADE-CUT}(z)$, where $z$ is the parent of $y$.

Consider a new implementation of $\text{CASCADE-CUT}(y)$ in which we do not mark or unmark nodes. Instead, we flip a fair independent coin (with probability half, the coin shows up heads, and with probability half, it shows up tails; further, the outcome of the coin flip is independent of the outcomes of all other coin flips). If it shows up heads, then we do $\text{CUT}(y)$ and $\text{CASCADE-CUT}(z)$; otherwise, we stop. Show that the amortized expected running time of each operation ($\text{DECREASE-KEY}$, $\text{DELETE-MIN}$, and $\text{ENQUEUE}$) for this new data structure is asymptotically identical to the amortized running time in the original implementation taught in class.

Problem 2
(a) Recall that the rank of a splay tree node was defined as the logarithm of its number of descendants, including itself. Instead, suppose we give an arbitrary positive integer weight $w(x)$ to each node $x$ and define a new rank function as the logarithm of the sum of weights of all descendants of a node, including itself. Show that for each of the three types of splay operations ($l$, $ll$, and $lr$), the bounds on the amortized running time derived in class still hold.

(b) Given a sequence $Q$ of $\text{FIND}$ queries on a set of distinct integers $I$, where each integer in $I$ is queried at least once, the optimal static binary search tree (BST) $T_Q$ is defined as a BST on $I$ that minimizes the total running time of the queries in $Q$. Let $C_Q$ denote this minimum total running time. Now suppose we build a splay tree $T_S$ on $I$ and let $C_S$ denote the total running time of the queries $Q$ on $T_S$. Use the property you derived in part (a) to show that there exists some constant $k$ such that $C_S \leq k \cdot C_Q$. 

Due on September 12, 2013