Data Structures and Algorithms (CS 130b)	**Prof. Suri**

Exercises 1

Handed Out: Jan. 8 — *Due: Jan. 15*

1. Consider the following Room Assignment problem. The input is a set of intervals over the real line. Think of an interval (x_i, y_i) as being a request for a room for a class that meets from time x_i to time y_i.

   Your goal is find an assignment of classes to rooms that **uses the fewest number of rooms**. (Note that every room request must be honored and that no two classes can use a room at the same time.)

   (a) Suppose we try the following iterative greedy scheme:

   > *Assign as many classes as possible to the first room (we can do this using the greedy algorithm discussed in class), then assign as many classes as possible to the second room, and so on.*

   Does this algorithm always find an optimal solution? Justify your answer.

   (b) Another possible greedy algorithm is the following.

   > *We process the classes in increasing order of start times. Assume that you are processing class C. If there is a room R such that R has been assigned to an earlier class, and C can be assigned to R without overlapping previously assigned classes, then assign C to R. Otherwise, put C in a new room.*

   Does this algorithm always find an optimal solution? Justify your answer.

2. Suppose a cell phone company wants to do stress testing on its product to determine the maximum height, measured in feet, from which it can be dropped and still not break. That is, the company wants to know the value h such that cell phones dropped from height h always survive, but break when dropped from $h + 1$ feet. (We assume that each cell phone made by this company is identical.) The company knows that value h lies between 0 and some (large) integer $n \geq 1$.

   One possible way to determine the safe height h is to drop the phone repeatedly from height 1, 2, and so on, until it breaks at some height k. Then, we know that the safe height is $k - 1$. This strategy destroys at most one phone, but *it requires as many as n drop experiments in the worst case.*

   On the other hand, we can try a binary search: try height $n/2$. If the phone survives, search the range $[n/2+1, n]$; otherwise, search the range $[0, n/2-1]$. This strategy reduces the number of drop experiments to $\lceil \log n \rceil$, but unfortunately, *it also may destroy* $\lceil \log n \rceil$ *phones.*

Explore the tradeoff between the number of phones destroyed and the number of drop experiments. In particular, how many drop experiments always suffice if we are willing to destroy **2 phones**?

Specifically, your strategy should involve dropping the phones at most $f(n)$ times for some function $f(n)$ that grows strictly sub-linearly, meaning $\lim_{n\to\infty} f(n)/n = 0$.

What strategy would you propose if we had $k > 2$ phones to destroy? How does the function $f(n)$ depend on k?