CS2223
HW#3

DUE: Thursday, November 20

1. (10 points) There are *m* nuts of distinct sizes $(n_1,...,n_m)$ and *m* bolts of distinct sizes $(b_1,...,b_m)$. Each bolt fits **exactly** one nut. You may not compare two nuts directly to each other, nor may you compare two bolts directly to each other. The only (benchmark) operation you may do is to compare a nut, say n_i, to a bolt, say b_j. The three possible outcomes of a comparison are that

- $n_i < b_j$, the nut is too small for the bolt,
- $n_i = b_j$, the nut fits the bolt,
- $n_i > b_j$, the nut is too large for the bolt.

Your goal is to find an algorithm for finding the correct nut for each bolt, and your algorithm must execute in expected time in $\Theta(m \lg m)$. You may describe your algorithm in precise pseudocode. (Hint: Think of a variation of QUICKSORT.)

2. (10 points) This problem is a variation on **Problem *2-1*** of our text. QUICKSORT is faster (on average) than INSERTIONSORT for large lists, but INSERTIONSORT is faster (on average) than QUICKSORT for small lists. Write and test a program to call QUICKSORT when the list being sorted is large (it contains more than Ψ numbers), and INSERTIONSORT when the list being sorted is small (it less than or equal to Ψ numbers), where Ψ is an integer to be determined. Note that Ψ =0 corresponds to traditional QUICKSORT.

Do some tests to determine Ψ_{opt}, the optimal value of Ψ for your system. Show results to show how much of a speedup your modified algorithm yields over traditional QUICKSORT. Describe your system.