1) (10 pts) ANL (Algorithm Analysis)

Consider the following algorithm to find the smallest item in a list of *n* distinct integers:

1. Pick an element, *x*, from the list at random.
2. Go through every other element in the list. If an element is less than *x* put it in list 1, and if it's more than *x*, put it in list 2. (Note: Since all elements in the list are distinct, none will equal *x*.)
3. If list 1 is empty, then return *x*, since it's the smallest element. If list 1 is NOT empty, go back to the first step, only using list 1.

In terms of *n*, what is the best case run-time of this algorithm? In terms of *n*, what is the worst case run-time of this algorithm? Please give justifications (both words and equations) for both answers. (Note: 8 points out of the 10 come from the justifications and the actual Big-Oh answers are only worth 1 point each.)

The best case run time is O(n). In the best case, the very first time we pick a random element, it turns out to be the smallest. In this case, we do n-1 comparisons in step 2, and then when list 1 is empty, we produce our result.

The worst case run time is $O(n^2)$. In the worst case what will happen is that we always pick the largest element in step 1. So, the first time, we do n-1 comparisons, yielding list 1 with n-1 values. This is followed by n-2 comparisons, yielding list 1 with n-2 values, and so forth. The total number of comparisons would be (n-1) + (n-2) + (n-3) + … + 1, or more formally, $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ comparisons. There are a few other steps that weren't counted (picking the random element and checking to see if list 1 is empty), but these are relatively minor steps. The function above can be expressed as $O(n^2)$, since it's roughly $\frac{1}{2}n^2$.

Grading:

1 pt for best case answer, 3 pts for justification - should include possibility of x being minimum off the bat and the corresponding run-time analysis of that case, give partial as necessary.

1 pt for worst case answer, 5 pts for justification - 2 pts for describing worst case scenario, 2 pts for obtaining the appropriate summation, 1 pt for solving the summation and concluding with the run time.