1. Use the integral method to get upper and lower bounds for $\sum_{j=0}^{n} j^2$. The two values should have exactly the same high order term. Show your work.

2. Consider an array of size nine with the numbers $50, 30, 70, 10, 90, 20, 80, 40, 60$. Assume you execute quicksort using the version of partition from CLRS. Note that in this algorithm an element might exchange with itself (which counts as one exchange).
   (a) Show the array after the first partition. How many comparisons and exchanges are used?
   (a) How many comparisons and exchanges are used for the entire quicksort?

3. Consider the following quicksort-like sorting algorithm. Pick two elements of the list. Partition based on both of the elements. So the elements smaller than both are to the left, the elements in between are in the middle, and the elements larger than both are to the right.
   (a) Given a brief English description of how you would partition. Write high level pseudo-code for this algorithm. Try to minimize the number of comparisons. Your algorithm should use a linear number of comparisons.
   (b) How many comparisons does the partition algorithm use in the worst case? Justify.
   (c) How many comparisons does the partition algorithm use on average. Just get the high order term. Justify informally.
   (d) Assume that the two partition elements always partition exactly into thirds. Write a recurrence for the number of comparisons. Solve this recurrence using constructive induction. Just get the high order term exactly.
   (e) Assume that the two partition elements always partition so exactly one quarter are to the left, one half in the middle, and one quarter to the right. Write a recurrence for the number of comparisons. Solve this recurrence using constructive induction. Just get the high order term exactly.
   (f) **Challenge problem, will not be graded.** Find the exact high order term for the average number of comparisons.

4. Selection Sort finds the largest element and puts it at the end of the array. Consider a version of Selecton Sort that finds the two largest elements and puts both of them at the end of the array (in order). You can think of this algorithm as: find the largest two elements, put them at the end of the list, and then recursively solve the problem on the remainder of the list. Write the pseudo-code for this recursive version of Selecton Sort. Make sure that it works when the size of the array is odd.