1. One way to find the median of a list is to sort the list and then take the middle element.
   (a) Assume you use Bubble Sort to sort a list with 11 elements (i.e. $n = 11$). Exactly how many comparisons do you use (in the worst case)?
   (b) i. Assume you use Mergesort to sort a list with 11 elements (i.e. $n = 11$). Exactly how many comparisons do you use (in the worst case)?
   ii. We know that if n were a power of 2 then the number of comparisons would be $n \lg n - n + 1$. What is this value for $n = 11$ rounded to the nearest integer?
   iii. How do the two values above compare?

2. You can actually find the median by running a sorting algorithm and stopping early, as soon as you know the median.
   (a) Assume you use Bubble Sort to find the median of 11 elements (i.e. $n = 11$), but stop as soon as you know the median. Exactly how many comparisons do you use (in the worst case)?
   (b) Assume you use Mergesort to find the median of 11 elements (i.e. $n = 11$), but stop as soon as you know the median. Exactly how many comparisons do you use (in the worst case)?

3. The selection algorithm (to find the *kth* smallest value in a list), described in the class (and in the book), uses columns of size 5. Assume you implement the same selection algorithm using columns of size 11, rather than 5.
   (a) Exactly how far from either end of the array is the median of medians guaranteed to be. Just give the high order term. (Recall that with columns of size 5 we got $\frac{3n}{10}$.)
   (b) It turns out that there is an algorithm that finds the median of 11 elements with 18 comparisons. Using this algorithm, briefly list each step of Selection with columns of size 11 and how many comparisons the step takes. Note that partition can now be done with only $(5/11)n$ comparisons.
   (c) Write a recurrence for the number of comparisons the algorithm uses.
   (d) Solve the recurrence using constructive induction. Just get the high order term exactly.
   (e) With more careful analysis, in class we could have obtained $16n$ comparisons using columns of size 5. How does this new value, using columns of size 11, compare?

4. Assume that we are trying to find the kth smallest number in a list of size n. Assume that we use the selection algorithm where we pivot on a random element. Make the simplifying assumption that at every recursive call, (the local) k is a random number between p and r (inclusive), so in the original recursive call k is a random number between 1 and n (inclusive). We will analyze the number of comparisons.
   (a) To warm up, assume that the pivot turns out to be always at the 1/4th mark, so in the original recursive call the pivot is at index $n/4$.
      i. There are actually three things that could happen in the original recursive call: we get lucky and $k = n/4$ so we are done, $k < n/4$, or $k > n/4$. What are these three probabilities?
      ii. Write a recurrence for the number of comparisons the algorithm uses.
      iii. Solve the recurrence using constructive induction. Just get the high order term exactly.
   (b) Now assume that the pivot is random as above.
      i. There are actually three things that could happen in the original recursive call: we get lucky and $k = q$ so we are done, $k < q$, or $k > q$. For fixed q, what are these three probabilities?
      ii. Write a recurrence for the number of comparisons the algorithm uses. It should involve a summation.
      iii. Solve the recurrence using constructive induction. Just get the high order term exactly.