Lab 5

1. Show all steps of QuickSort in sorting the array [1, 6, 2, 4, 3, 5]. Use leftmost values as pivots at each step.

2. Show all steps of In-Place QuickSort in sorting the array [1, 6, 2, 4, 3, 5] when doing first partition. Use leftmost values as pivots.

3. In our average case analysis of QuickSort, we defined a *good self-call* to be one in which the pivot *x* is chosen so that number of elements < x is less than 3n/4, and also the number of elements > x is less than 3n/4. We call an x with these properties a *good pivot.* When n is a power of 2, it is not hard to see that at least half of the elements in an n-element array could be used as a good pivot (exactly half if there are no duplicates). For this exercise, you will verify this property for the array A = [5, 1, 4, 3, 6, 2, 7, 1, 3] (here, n = 9). Note: For this analysis, use the version of QuickSort in which partitioning produces 3 subsequences *L, E, R* of the input sequence *S.*
   a. Which x in A are good pivots? In other words, which values x in A satisfy:
      i. the number of elements < x is less than 3n/4, and also
      ii. the number of elements > x is less than 3n/4
   b. Is it true that at least half the elements of A are good pivots?

4. *Interview Question.* Give an o(n) ("little-oh") algorithm for determining whether a sorted array A of distinct integers contains an element m for which A[m] = m. You must also provide a proof that your algorithm runs in o(n) time.

5. Devise a pivot-selection strategy for QuickSort that will guarantee that your new QuickSort has a worst-case running time of O(nlog n). Explain why your new QuickSort has this running time. (Hint: Use QuickSelect to pick the pivot.)