Lab 3

1. 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.

2. 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?

3. In class, we discussed the "good case" for QuickSort. Is it the best case for QuickSort? If not, what is the best case? Use a recursion tree to explain the running time in the best case.

4. https://leetcode.com/problems/kth-largest-element-in-an-array/description/ (We know we can use QuickSelect Algorithm for this problem. There is a partition step in QuickSelect Algorithm, and we have done it 'in place' before. For this problem, implement partition step in place and then see how we can get the kth largest element from the result of InPlacePartition.)