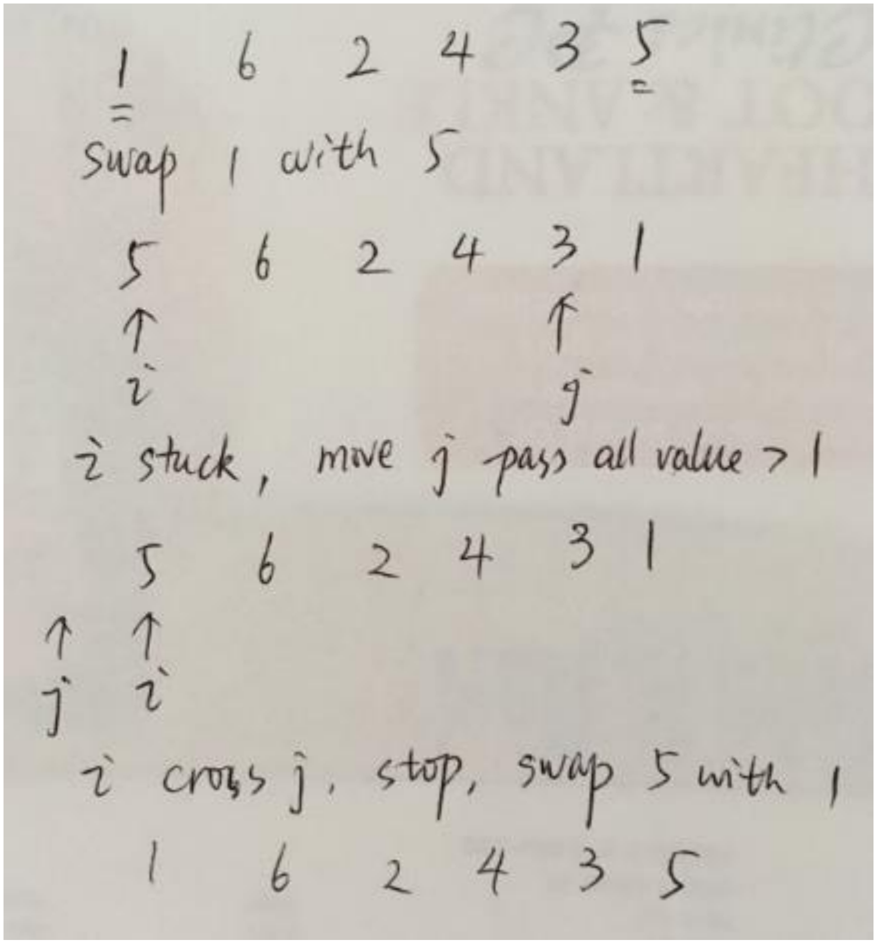
Lab 5 Solutions

Problem 2.



Problem 3.

a. Good pivots: 2,3,3,4,5

b. Yes: 5/9 of the elements are good pivots.

Problem 4.

[22/6] Give an o(n) (that is, better than $\Theta(n)$) algorithm for determining whether a sorted array A of distinct integers contains an element m for which $A[m] = m$, and then implement as a Java function

```
int findFixedPoint(int[] A)
```

which returns such an m if found, or -1 if no such m is found. You must also provide a proof that your algorithm runs in o(n) time.

Step 1: If A[0] = 0, return 0.

Step 2: If A[0] > 0, return -1.

Step 3: Do binary search. The base case will examine A[mid] to see if A[mid] = mid, and if so, return A[mid] (it's the a fixed point). If A[mid] > mid, search the left side. If A[mid] < mid, search the right side. If the usual failure signal occurs (lower > upper) return -1. This solves the problem in O(log n).

Problem 5.

Review of SubsetSum Problem: Given a set S = {s0, s1,s2, …, sn-1} of positive integers and a non-negative integer k, find a subset T of S so that the sum of the integers in T equals k or indicate no such subset can be found.

We have already seen a brute force solution to this problem in an earlier lab. In this exercise, you are going to come up with a recursive solution for SubsetSum. Write down pseudo code for your algorithm or Java code.

Hint:

We are seeking a $T \subseteq S = \{s_0, s_1, \ldots, s_{n-2}, s_{n-1}\}$ whose sum is k. Such a T can be found if and only if one of the following is true:

(1) A subset T_1 of $\{s_0, s_1, \ldots, s_{n-2}\}$ can be found whose sum is k, OR

(2) A subset T_2 of $\{s_0, s_1, \ldots, s_{n-2}\}$ can be found whose sum is $k - s_{n-1}$

If (1) holds, then the desired set T is T_1. If (2) holds, the desired set T is $T_2 \cup \{s_{n-1}\}$.

Soln:

```
Algorithm RecSubsetSum(S, k)
    Input: S = {s0, s1, . . ., sn-1} positive integers,
           k nonnegative integer
    Output: T ⊆ S for which sum(T) = k
    //base case
    if S.size() = 1 then
        if k = 0 then return {}
        else if k = s0 then return {s0}
        else return NULL
    (S, last) ← S.removeLast()
    T ← RecSubsetSum(S, k)
    if T not NULL then
        return T
    T ← RecSubsetSum(S, k – last)
    if T not NULL then
        return T ∪ {last}
    return NULL
```