Homework Assignment #4
Due: February 7, 2020 at 2:30 p.m.

1. Consider the following recursive sorting algorithm. The arguments are an array A of n integers and an integer lo.

```
1  SORT(A[1..n], lo)
2      Precondition: 1 ≤ lo ≤ n and A[1..lo − 1] is sorted in non-decreasing order and
                         if lo > 1 then A[lo − 1] is less than each element of A[lo..n].
3      m ← min(A[lo..n])
4      for i ← lo..n
5          if A[i] = m then
6              swap the contents of A[lo] and A[i]
7              lo ← lo + 1
8          end if
9      end for
10     if lo ≤ n then
11         SORT(A[1..n], lo)
12     end if
13     Postcondition: A[1..n] is sorted in non-decreasing order
14 end SORT
```

(a) State a loop invariant for the loop at lines 4–9. You do not have to prove your invariant is an invariant, but it should be strong enough so that you could prove it, and also use it to answer the rest of this question.

(b) What is a good measure of the size of the input, for the sake of proving the algorithm is correct?

(c) Assuming that your statement in part (a) is a loop invariant, prove SORT is correct.

(d) To sort the whole array A, you can call SORT$(A[1..n], 1)$. If A contains m distinct values, give a good asymptotic bound on the worst-case running time of SORT, stating your answer in terms of m and n. Explain briefly why your answer is correct.

(e) Can you think of a situation (i.e., a class of inputs) for which this algorithm would have better asymptotic worst-case behaviour than any of the sorting algorithms we have mentioned in class (selection sort, insertion sort, bubble sort, merge sort, heap sort, quick sort, counting sort, radix sort, bucket sort)? Justify your answer.