CSCE 411: Mastery Quiz 1

Name: _____

UIN: _____

Consider the following algorithm:

```
SillySort(A[1...n]):
  For i from 1 to n − 1:
      For j ∈ {1} ∪ {i + 1, ..., n}:
          count ← 0
          For k ∈ {1} ∪ {i + 1, ..., n}:
              If A[k] < A[j]:
                  count ← count + 1
          If count = 1:
              Swap A[j] with A[i + 1]
  Return A
```

Prove that for every positive integer n and every array of n **distinct** integers, SillySort(A) correctly sorts A.

Solution: Before looking at the solution on the next page, consider the following equivalent version of the algorithm (where the inner two for loops have been replaced with two lines of pseudocode) and see if you can figure out a loop invariant that you could use to prove its correctness.

```
SillySort(A[1...n]):
  For i from 1 to n − 1:
      j ← the index of the second smallest element from A[1, i + 1, i + 2, ..., n]
      Swap A[j] with A[i + 1]
  Return A
```

Note that this is equivalent to the original version because the original pseudocode iterates j through all of the values in $\{1\} \cup \{i+1, \ldots, n\}$ and for each value of j, the inner loop counts the number of values from $A[1, i+1, i+2, \ldots, n]$ which are smaller than $A[j]$. If this number is 1 (i.e. if $A[j]$ is the second smallest element of $A[1, i+1, i+2, \ldots, n]$) then the algorithm swaps $A[j]$ with $A[i+1]$.

Translating the original psuedocode into the version above was not supposed to be the hard part of the problem, but apparently it was the biggest challenge for a lot of you. If you can prove the correctness of the simplified version of the algorithm, then you understand the core induction problem that the quiz was asking.

If you do not understand the notation in a quiz or exam, it's worth raising your hand and asking about it. Sometimes, the notation may be part of the point of the question (particularly on comprehension quizzes), but other times we would be happy to explain it to your or even issue a clarification to the whole class. The worst I'll say is "I'm sorry, I can't tell you that."

Solution: Let A be an arbitrary array of distinct numbers. We will prove, by induction on i, that after i iterations of the algorithm's outer for loop, $A[2, \ldots, i+1]$ contains the 2nd smallest through $(i+1)$th smallest values in A in sorted order.

- Base case: Suppose $i = 0$. Vacuously, after 0 iterations of the loop, the empty subarray $A[2:1]$ contains 0 values in sorted order.
- Induction hypothesis: Suppose that $\forall\, \ell \in \mathbb{Z}_{\geq 0}$ with $\ell < i$, after ℓ iterations of the for loop the 2nd smallest through $(\ell+1)$th smallest elements of A are stored in positions 2 through $\ell + 1$ in sorted order.
- Induction Step: We will prove that after i iterations, the 2nd smallest through $(i+1)$th smallest elements are stored in positions 2 through $i+1$ in sorted order. By our induction hypothesis, after iteration $i-1$ of the loop, $A[2, \ldots, i]$ contained the 2nd smallest through ith smallest smallest elements of A in sorted order.

  We observe that in iteration i of the loop, the algorithm iterates through $A[1, i+1, i+2, \ldots, n]$, finds the second smallest element of this subarray, and swaps it to position $i+1$. Since $A[2, \ldots, i]$ already contained the 2nd smallest through ith smallest elements of A, the second smallest element of $A[1, i+1, i+2 \ldots, n]$ will be the $(i+1)$'th smallest element of the overall array. Therefore, since this loop iteration does not modify $A[2, \ldots, i]$, at the end of the i'th iteration we have that $A[2, \ldots, i+1]$ contains the 2nd smallest through $(i+1)$'th smallest values in the array and they are in sorted order.

We have now shown that after $n-1$ iterations of the outer for loop (i.e. at the end of the algorithm), $A[2, \ldots, n]$ contains the 2nd smallest through nth smallest elements of the array in sorted order. The only remaining element is the smallest element and the only remaining position is $A[1]$, so the smallest element must be in $A[1]$ at this point, meaning that the entire array is sorted. Therefore, SillySort correctly sorts A.