Correctness of Selection Sort:

→ Name : Sanjukktha
UTA ID: 1002239323

Pseudo Code :

```
n = A.length
for i = 0 to n-2
    min = i
    for j = i+1 to n-1
        if A[j] < A[min]
            min = j
    if min ≠ i
        swap A[i] w/ A[min]
```

Loop invariant for inner loop (4-6)

__Initialization__ : Before 1st iteration of inner loop (when $j = i+1$), $min = i$ smallest element in subarray '$A[i \ldots 1]$' → single element.

__Maintenence__ : Each iteration of inner loop $A[j]$ is smaller than $A[min]$, algo updates min to j. min = always smallest in $A[i \ldots j]$

Termination: When inner loop terminates (when $j = n$), min points to index of smallest in $A[i \dots n-1]$

Outerloop (2-8):

Initialization: Before 1st iteration of outerloop (when $i = 0$) the subarray $A[0 \dots i-1]$ is empty, An empty array is sorted.

Maintenence:

Subarray $A[0 \dots i]$: smallest '$i+1$' elements in sorted order, each element in $A[0 \dots i]$ is less than equal to each element in $A[i+1 \dots n-1]$
Thus loop invariant is maintained.

Termination:

When the outer loop terminates after $i = n-1$, the loop invariant guarantees that the subarray

A[0.... n-2] is sorted and that every element A[0 .. n-2) is less than or equal to remaining element A[n-1]. Since A[n-1] is already the largest element in the array, the entire array is sorted

Time Complexity:

worst case } ⇒ $O(n^2)$ Outerloop runs n-1 times & the inner loop performs a linear search over the unsorted part of the array for each iteration of the outer loop.
