2. Selection sort correctness.

Selection sort is a simple comparison-based sorting algorithm that works by dividing the input list into two parts: a sorted sub-list and an unsorted sub-list. The sorted sub-list is filled by the algorithm iteratively choosing the smallest (or largest, depending on the sorting order) element from the unsorted sub-list and moving it to the end. Until the entire list is sorted, this process is repeated.

In order to argue for selection sort's correctness, we must first prove two important points:

1. **Initialization :** Starting from the beginning of each iteration, the part of the list that comes before the current position is sorted, while the part that comes after is left unsorted. Because it is the only element in the list, the first element can be seen as having already been sorted, which makes this true even before the first iteration. All of the list is therefore the unsorted sub-list, and the method begins appropriately with an empty sorted sub-list.
2. **Maintenance :** The smallest member from the unsorted sub-list is moved to its proper location in the sorted sub-list at the end of each iteration. This preserves the invariant that the unsorted sub-list drops with each iteration while the sorted sub-list remains correctly sorted and increases in size.

Selection sort gradually sorts the entire list because it continues to select the smallest element from the unsorted sub-list and adding it to the end of the sorted sub-list.

To sum up, selection sort makes sure that the smallest (or largest) element from the unsorted sub-list is appropriately inserted into the sorted sub-list at each stage, thus accurately maintaining the invariant of having a sorted sub-list and an unsorted sub-list. This procedure keeps going until the full list is sorted, proving that the selection of sort is correct.