bubbleSort(A):

```
1.  n = length(A)
2.  for i = 0 to n-1:
3.    for j = 0 to n-i-1:
4.      if A[j] > A[j+1]:
5.        swap(A[j], A[j+1])
```

Loop invariant: At the beginning of each iteration of the outer for loop, the last i elements of the array are in their correct, sorted positions.

Proof:

- Initialization: At the beginning of the first iteration of the outer for loop, i = 0, so the last 0 elements of the array are already in their correct, sorted positions.
- Maintenance: Suppose that at the beginning of some iteration of the outer for loop, the last i elements of the array are in their correct, sorted positions. During the ith iteration, the inner for loop performs n-i-1 comparisons and swaps, ensuring that the (n-i)-th largest element is moved to its correct, sorted position. Therefore, at the end of the ith iteration, the last i+1 elements of the array are in their correct, sorted positions.
- Termination: When the outer for loop terminates, i = n-1, so the last n-1 elements of the array are in their correct, sorted positions. By the loop invariant, this means that the entire array is sorted.

linearSearch(A, x):

```
1.  for i = 0 to length(A)-1:
2.    if A[i] == x:
3.      return i
4.  return -1
```

Loop invariant: At the beginning of each iteration of the for loop, x has not been found in the subarray A[0:i].

Proof:

- Initialization: At the beginning of the first iteration of the for loop, i = 0, so the subarray A[0:i] is empty and x has not been found.
- Maintenance: Suppose that at the beginning of some iteration of the for loop, x has not been found in the subarray A[0:i]. During the ith iteration, we check if A[i] == x. If it is, we return i. Otherwise, we increment i and continue to the next iteration. By the loop invariant, x has not been found in the subarray A[0:i+1].
- Termination: If x is in the array, it must be found by some iteration of the for loop. If x is not in the array, the for loop terminates after i = length(A)-1, and we return -1. Therefore, the algorithm is correct.

findFirst(A, x):

1. **for i = 0 to length(A)-1:**
2. **   if A[i] == x:**
3. **   return i**
4. **return -1**

Loop invariant: At the beginning of each iteration of the for loop, x has not been found in the subarray A[0:i].

Proof:

- Initialization: At the beginning of the first iteration of the for loop, i = 0, so the subarray A[0:i] is empty and x has not been found.
- Maintenance: Suppose that at the beginning of some iteration of the for loop, x has not been found in the subarray A[0:i]. During the ith iteration, we check if A[i] == x. If it is, we return i, which is the index of the first occurrence of x in the array. Otherwise, we increment i and continue to the next iteration. By the loop invariant, x has not been found in the subarray A[0:i+1].
- Termination: If x is in the array, it must be found by some iteration of the for loop. If x is not in the array, the for loop terminates after i = length(A)-1, and we return -1. Therefore, the algorithm is correct.