1. (Consider the python function below (read the comments carefully) **1 / 1 point**

```
1
2
3
4
5
6
7
8
9
```

What is the overall time cost of calling the "foo" function, in terms of $n, c1, c2, c3, c4$ and $c5$?

$n * n * (c2 + c3 + c4 + c1) + c1 + c5$

$(n + 1) * c1 + n * (c2 + c3 + c4) + c5$

$c1 + n * (c2 + c3 + c4) + c5$

$c1 + c2 + c3 + c4 + c5$

Correct

2. Consider the following array, which is *almost* sorted in ascending order. There are just two elements (3 and 7) out of place. **1 / 1 point**

$A = [1, 2, 7, 4, 5, 6, 3, 8, 9]$

Select all the true facts about running insertion sort on A from the list below. Ensure that no wrong choices are selected.

During the execution of insertion sort, when the element 7 is to be inserted into the sorted portion $[1, 2]$, no swap operation will occur because $2 < 7$.

Correct

After 7 has been inserted, the insertion of elements 4, 5 and 6 will incur one swap operation each, with the number 7 remaining at the end of the sorted portion of the array.

Correct

Insertion of the element 3 into the sorted portion $[1, 2, 4, 5, 6, 7]$ involves 4 swap operations, with 4, 5, 6 and 7 respectively.

Correct

3. Consider this array of size n sorted in descending order: $[n, n-1, \ldots, 1]$. **1 / 1 point**

Suppose we ran insertion sort to sort this array in *ascending* order.

Select all the correct options from the list below.

After i steps, suppose the sorted portion is $[n-i+1, \ldots, n]$ and the element to be inserted is $(n-i)$. We will need to perform i swaps to ensure that $n-i$ is inserted in the correct place.

Correct

The total number of swaps is given by:

$$1 + 2 + \cdots + (n-1) = \frac{n(n-1)}{2}$$

Correct

Consider a different array `a: [a1, a2, ..., an]` that satisfies the property that `a[i] < a[i+1]` for all but one place `a[j]` wherein `a[j] > a[j+1]`. Insertion sort as presented in lecture will run in $\Theta(n)$ time for such an "almost" ascending sorted array.