Consider the following algorithm:

```python
def order(L):
    """ (list of numbers) -> None
    Order L from smallest to largest. L is changed in-place. """
    i = 1
    while i < len(L):
        j = i
        while j > 0 and L[j] < L[j-1]:
            L[j], L[j-1] = L[j-1], L[j]  # swap L[j] and L[j-1]
            j = j - 1
        i = i + 1
```

1. Compute the number of "swaps" (executing the line that says swap) performed by the algorithm in the worst-case, on any list $L$ of length $n$.

   The `def` line can be ignored: it is part of the syntax to define a function, but not something that actually gets executed every time we call the function. So we count only the steps in the body of the function.

   The outer loop iterates over $i = 1, 2, 3, \ldots, n-1$.

   For each value of $i$, the inner loop iterates over $j = i, i-1, \ldots, 2, 1$, as long as $L[j] < L[j-1]$. In the worst-case (when $L$ is initially sorted in reverse order), this happens for every value of $j$.

   For each value of $j$, the algorithm swaps once: 1 time when $i = 1$ (for $j = 1$), 2 times when $i = 2$ (for $j = 2$ and $j = 1$), \ldots, $n-1$ times when $i = n-1$ (for $j = n-1$ and \ldots and $j = 1$).

   So in total, the algorithm performs exactly $1 + 2 + \cdots + (n-1) = n(n-1)/2 = n^2/2 - n/2$ swaps, in the worst-case.
2. Compute the number of “steps” (basic operations) performed by the algorithm in the worst-case, on any list $L$ of length $n$. Count a step each time a line is visited.

As before, we count only the lines in the body. The outer loop iterates over $i = 1, 2, 3, \ldots, n-1$. For each value of $i$, the inner loop iterates over $j = i, i-1, \ldots, 2, 1$, in the worst-case (as argued in the first question).

For each value of $j$, the algorithm performs 3 steps. So over all values of $j$, a total of $3i$ steps.

In addition, for each value of $i$, there are steps performed outside of the inner loop: 3 steps for the lines outside the inner loop, and an additional 1 step to evaluate the last inner loop condition — when the condition becomes False. So each iteration of the outer loop performs $3i + 4$ steps.

Together with the first line, and the extra 1 step for the last outer loop condition, the number of steps performed by the algorithm is exactly:

$$
\left( \sum_{i=1}^{n-1} (3i + 4) \right) + 2 = 3 \left( \sum_{i=1}^{n-1} i \right) + 4 \left( \sum_{i=1}^{n-1} 1 \right) + 2 \\
= 3n(n - 1)/2 + 4(n - 1) + 2 \\
= 3n^2 + 5n - 4
$$