NOTE: This file contains sample solutions to the quiz together with the marking scheme and comments for each question. Please read the solutions and the marking schemes and comments carefully. Make sure that you understand why the solutions given here are correct, that you understand the mistakes that you made (if any), and that you understand *why* your mistakes were mistakes.

Remember that although you may not agree completely with the marking scheme given here, it was followed the same way for all students. We will remark your quiz only if you clearly demonstrate that the marking scheme was not followed correctly.

For all remarking requests, please submit your request **in writing** directly to your instructor. For all other questions, please don't hesitate to ask your instructor during office hours or by e-mail.

GENERAL MARKING SCHEME:

- ***A****: All Correct*, except maybe for very few minor errors.
- ***B****: Mostly Correct*, but with a few serious errors, or many small errors.
- ***C****: Mostly Incorrect*, but with a few important elements, or many small elements, done correctly.
- ***10%****: Completely Blank*, or clearly crossed out.
- ***D****: All Incorrect*, except maybe for very few minor elements done correctly.

MARKER'S COMMENTS:

- Many students confused the notions of "lower bound on the worst-case running time" and "best-case running time." [*Instructor's Comment: If you are unsure about this distinction, please take the time to ask and find out!*]

1. Give a tight bound on the worst-case running time of the following algorithm. Include justifications for your upper bound and your lower bound.

```
# Precondition: L is a non-empty list of numbers.
for i := 1,2,...,len(L)-1:
    j := i
    while j > 0 and L[j] < L[j-1]:
        swap L[j] and L[j-1]
        j := j - 1
# Postcondition (even though this is not necessary for the question):
# L is sorted in non-decreasing order.
```

Upper Bound: Let $n = \text{len}(L)$.

The loop for i iterates $n - 1$ times.

The loop for j iterates no more than i times, which is $\leqslant n - 1$ times.

The body of the loop for j takes constant time.

So the loop for j takes worst-case time $\mathcal{O}(n) = \mathcal{O}(n \times 1)$.

So the loop for i takes worst-case time $\mathcal{O}(n^2) = \mathcal{O}(n \times n)$.

Lower Bound: Let $n = \text{len}(L)$ and $L = [n, n-1, \ldots, 2, 1]$.

Then for every value of i, the initial value of $L[i]$ is strictly smaller than every value in $L[0 \ldots i-1]$.

So the loop for j will iterate exactly i times because the condition $L[j] < L[j-1]$ will remain true.

Since the body of the loop for j takes constant time, the loop for j performs at least i steps.

Over all the values of i, this means that the algorithm performs at least $1+2+\cdots+n-1 = n(n-1)/2$ steps.

So the algorithm takes time $\Omega(n^2)$ in the worst-case.