Work on these exercises before the tutorial. You don’t have to come up with complete solutions before the tutorial, but you should be prepared to discuss them with your TA.

IMPORTANT: Where applicable, you must use the proof structures and format of this course.

For this exercise, we will be using the following algorithm:

```python
def meaning_of_life(A):
    """ A function that takes a list A and outputs t """
    # Precondition: _______________________________________

    n = len(A)
    t = 0
    if A[0] % 2 == 1:
        i = 0
        while i < n**2:
            t += A[i % n]
            i += 1
    else:
        i = n-1
        while i >= 0:
            t += A[i]
            i -= 1
    return t
```

1. Is there a precondition for meaning_of_life? Think about how a precondition for an algorithm relates to $B' \in \mathbb{N}$ for run-time proofs, and whether one is necessary in this case.

Solution:
The precondition should be: A contains $n>0$ numbers, where $n = \text{len}(A)$. While your run-time formula (re: Q2) and analysis (re: Q3) may work without this requirement, you must consider this when choosing your value for $B'$. Otherwise, you could be mathematically correct, but practically wrong—meaning_of_life will return an error if you try to run it on an empty list A, at least in Python. You should always keep these things in mind when analysing an algorithm.
2. How many steps will meaning_of_life take for A = [1, 2, 3]? A = [2, 1, 3]?

Solution: A = [1, 2, 3]

```python
n = len(A) # n = 3 1 step
t = 0 1 step
if A[0] % 2 == 1: # true 1 step
    i = 0 1 step
    while i < n**2: # 0 < 9 1 step
        t += A[i % n] # t = 0 + 0 = 0 1 step
        i += 1 # i = 0 + 1 = 1 1 step
    # 1 < 9, 2 < 9, ..., 8 < 9 8*3 more steps
    # 1 more step for the closing loop condition, 9<9
else: # irrelevant 0 steps
    i = n-1 0 steps
    while i >= 0: 0 steps
        t += A[i] 0 steps
    i -= 1 0 steps
return t 1 step
```

Therefore, this will take 33 steps.

Solution: A = [2, 1, 3]

```python
n = len(A) # n = 3 1 step
t = 0 1 step
if A[0] % 2 == 1: # false 1 step
    i = 0 0 steps
    while i < n**2: # 0 < 9 0 steps
        t += A[i % n] # t = 0 + 0 = 0 0 steps
        i += 1 # i = 0 + 1 = 1 0 steps
    else: by definition of step 4; so, 0 (extra) steps
        i = n-1 # i = 2 1 step
    while i >= 0: # 2 >= 0 1 step
        t += A[i] # t = 0 + 2 = 2 1 step
    i -= 1 # i = 1 1 step
    # 1 >= 0, 0 >= 0 6 more steps
    # 1 more step for the closing loop condition, -1 <= 0
return t 1 step
```

Therefore, this will take 15 steps.
3. What is the formula for the running time of `meaning_of_life`? What is the formula for the worst-case running time of `meaning_of_life`?

If you’re unsure of what the difference is, recall Q3 from Tutorial 6.

**Solution:**

From Q2, we can see that lines 5, 6, 7, and 17 (as in the original question), take exactly 1 ‘time’ each, no matter the input—as long as the precondition holds. These correspond to assigning values (e.g. \( t = 0 \)), checking an if condition, and returning a value.

These correspond to 4 steps.

Now, we have two cases, one in which \( A[0] \) is odd, and one in which \( A[0] \) is even.

**Case 1:** \( A[0] \) is odd. Then the outer loop will need \( 3n^2 + 2 \) steps.

**Case 2:** \( A[0] \) is even. Then the outer loop will need \( 3n + 2 \) steps.

Therefore, the **running time function** of `meaning_of_life` is:

\[
\text{meaning_of_life}(n) = \begin{cases} 
3n^2 + 6, & \text{if } A[0] \text{ is odd} \\
3n + 6, & \text{if } A[0] \text{ is even}
\end{cases}
\]

Based on the above, the **worst-case running time function** is \( 3n^2 + 6 \).

4. Prove or disprove: `meaning_of_life(n) \in \Omega(n^3)`.

**Solution:**

Based on the above, the worst-case running time function is \( 3n^2 + 6 \), so we will only consider the worst case, i.e. when the first element of \( A \) is odd. Then the claim is false, so we need to prove \( \forall c \in \mathbb{R}^+, \forall B \in \mathbb{N}, \exists n \in \mathbb{N}, n \geq B \land \text{meaning_of_life}(n) < cn^3 \). Now:

\[
\lim_{n \to \infty} \frac{3n^2 + 6}{n^3} = 0
\]

Then, we know the following:

\[
\forall \epsilon \in \mathbb{R}^+, \exists n' \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq n' \implies \frac{3n^2 + 6}{n^3} < \epsilon
\]

Assume \( c \in \mathbb{R}^+, B \in \mathbb{N} \)

We know that \( \exists n' \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq n' \implies \frac{3n^2 + 6}{n^3} < c \quad \# \text{from definition, with } \epsilon = c \)

// Prove the first part of your statement, i.e. \( n \geq B \)

Let \( n_1 \) be such that \( \forall n \in \mathbb{N}, n \geq n_1 \implies \frac{3n^2 + 6}{n^3} < c \)

Let \( n_0 = \max(B, n_1) \); then \( n_0 \in \mathbb{N} \)

Then, \( n_0 \geq B \quad \# \text{definition of max} \)

// Now, prove that \( \text{meaning_of_life}(n) < cn^3 \)

Then, \( n_0 \geq n_1 \quad \# \text{definition of max} \)

Then, \( \frac{3n_0^2 + 6}{n_0^3} < c \quad \# \text{follows from limit definition} \)

Then \( 3n_0^2 + 6 < cn_0^3 \), and so \( \text{meaning_of_life}(n_0) < cn_0^3 \)

Then, \( n_0 \geq B \land \text{meaning_of_life}(n_0) < cn_0^3 \quad \# \text{simple conjunction} \)

Then \( \forall c \in \mathbb{R}^+, \forall B \in \mathbb{N}, \exists n \in \mathbb{N}, n \geq B \land \text{meaning_of_life}(n_0) < cn_0^3 \)

So, \( \text{meaning_of_life}(n_0) \notin \Omega(n^3) \)
5. The following algorithm was also discussed in tutorial:

```python
def order(L):
    i = 1
    while i < len(L):
        j = i
        while j > 0 and L[j] < L[j-1]: # mention that you consider this 1 step
            # or 3 steps; I choose 1 for simplicity
            L[j], L[j-1] = L[j-1], L[j]
            j -= 1
        i += 1
```

The outer loop iterates over \( i = 1, 2, 3, \ldots, n-1 \), and for each \( i \), the inner loop iterates over \( j = i, i-1, \ldots, 2, 1 \), as long as \( L[j] < L[j-1] \). So in the worst case, there are \( 1 + 2 + 3 + \cdots + n-1 = n(n-1)/2 \) swaps (line 7).

For each value of \( j \), the algorithm performs 3 steps, so over all \( j \), there are \( 3i \) steps. There are also 3 steps for the lines in the inner loop for each \( i \), and an additional step to evaluate the last inner loop condition; each iteration of the outer loop, then, takes \( 3i + 4 \) steps.

The total number of steps for the algorithm is then:

\[
\left( \sum_{i=1}^{n-1} (3i + 4) \right) + 2 = 3 \left( \sum_{i=1}^{n-1} i \right) + 4 \left( \sum_{i=1}^{n-1} 1 \right) + 2 \\
= 3 \left( \frac{n(n-1)}{2} \right) + 4(n-1) + 2 \\
= 3n^2 + 5n - 4
\]