1. Show, using the definition of $\Theta$, that $100n + 10n \log_2 n + n^2 = \Theta(n^2)$.

2. Find the order of growth (up to $\Theta$) of $10 \sum_{i=0}^{n-1} (i^5 - i^4)$, showing your work.

3. Find the order of growth (up to $\Theta$) of $\sum_{i=1}^{n} (4 \log_2 i)$, showing your work.

4. Prove that, for every $a$ greater than zero and every $b$, $(n + b)^a = \Theta(n^a)$.

5. Prove that, for every $a$ greater than one and every $b$ greater than zero, $\log_a (n + b) = \Theta(\log_a n)$.

6. Consider the following pseudocode:

   **Input:** $A$, a set of numbers
   **Input:** $B$, a set of numbers

   1. let $s \leftarrow 0$
   2. for $a \in A$ do
   3.   for $b \in B$ do
   4.     let $s \leftarrow s + ab$
   5.   end
   6. end
   7. return $s$

   **a.** What are $n$ and $m$, the input sizes for this pseudocode?
   **b.** For the purposes of analyzing time complexity, what are the basic operation(s) in this pseudocode?
   **c.** What is this algorithm’s asymptotic time complexity? Show your work.

7. Consider the following pseudocode:

   **Input:** $A$, a list of numbers

   1. let $s \leftarrow 0$
   2. let $i \leftarrow 0$
   3. for $a \in A$ do
   4.   let $i \leftarrow 1$
   5.   while $i < |A|$ do
   6.     let $s \leftarrow s + 1$
   7.     let $i \leftarrow 2i$
   8.   end
   9.   let $s \leftarrow s + 1$
   10. end
   11. for $x \in [0, \ldots, i - 1]$ do
   12.   let $s \leftarrow s + 1$
   13. end
   14. return $s$

   **a.** What is $n$, the size of the input, for this pseudocode?
   **b.** For the purposes of analyzing time complexity, what are the basic operation(s) in this pseudocode?
   **c.** What is this algorithm’s asymptotic time complexity? Show your work.
   **d.** Where are the bottleneck(s)? I.e., which loop(s) dominate the runtime for large $n$?
8. Consider the following pseudocode:

Input: A, a list of numbers
1. let \( i \leftarrow 1 \)
2. while \( i < |A| \) do
3.   if \( i < A[i] \) then
4.     return \( A[i] \)
5.   end
6.   let \( i \leftarrow i + 1 \)
7. end
8. return \( \perp \)

a. What is the best-case runtime of this algorithm? Show your work.

b. What is the worst-case runtime of this algorithm? Show your work.

9. Part of the application you are working on needs to find the two smallest values in a list. Currently the implementation is:

Input: A, a list
1. let \( r \leftarrow (\infty, \infty) \)
2. for \( i \in [0 \ldots |A| - 1] \) do
3.   for \( j \in [0 \ldots |A| - 1] \) do
4.     if \( i \neq j \land (A[i], A[j]) < r \) then
5.       let \( r \leftarrow (A[i], A[j]) \)
6.     end
7. end
8. end
9. return \( r \)

where tuples are compared lexicographically. One of your colleagues proposes the following revision:

Input: A, a list
1. if \( |A| \geq 2 \) then
2.   let \( B \leftarrow \text{sorted}(A) \)
3.   return \( (B[0], B[1]) \)
4. end
5. return \( (\infty, \infty) \)

Take comparisons to be the basic operations, and assume that sorting a list requires \( \Theta(n \log n) \) comparisons. Is the suggestion asymptotically faster than the original algorithm? Show your work.

10. Part of the application you are working on needs to find the two smallest values in a list.

a. Prove by contradiction that every algorithm that solves this problem must access each list element at least once.

b. Because there must be at least a linear number of accesses, any algorithm that solves this problem in linear time is asymptotically optimal. Give such an algorithm and prove that it runs in linear time.