Reminder: \( \log n \) denotes the binary logarithm, i.e., \( \log n = \log_2 n \).

1 **O-notation: Part I**

Give formal proofs of the following statements using the definition of Big-O from the lecture.

1. \( 10 \in O(1) \).
2. \( 5n \in O(n) \).
3. \( n^2 + 10n \in O(\frac{1}{10}n^3) \).
4. \( \sum_{i=1}^{n} i \in O(4n^2) \).

2 **Racetrack Principle**

1. Use the racetrack principle to prove the following statement:
   \[ n \leq e^n \text{ holds for every } n \geq 1. \]

2. Use the racetrack principle and determine a value \( n_0 \) such that
   \[ \frac{2}{\log n} \leq \frac{1}{\log \log n} \text{ holds for every } n \geq n_0. \] (Difficult!)

   *Hint:* Transform the inequality and eliminate the log-function from one side of the inequality before applying the racetrack principle.
   Recall that \( (\log n)' = \frac{1}{n \ln(2)} \). The inequality \( \ln(2) \geq 1/2 \) may also be useful.

3 **O-notation: Part II**

Give formal proofs of the following statements using the definition of Big-O from the lecture.

1. \( f \in O(h_1), g \in O(h_2) \) then \( f + g \in O(h_1 + h_2) \).
2. \( f \in O(h_1), g \in O(h_2) \) then \( f \cdot g \in O(h_1 \cdot h_2) \).
3. \( 2^n \in O(n!) \)
4 Fast Peak Finding

Consider the following variant of Fast-Peak-Finding where the “≥” sign in the condition in instruction 4 is replaced by a “<” sign:

1. if $A$ is of length 1 then return 0
2. if $A$ is of length 2 then compare $A[0]$ and $A[1]$ and return position of larger element
3. if $A[\lfloor n/2 \rfloor]$ is a peak then return $\lfloor n/2 \rfloor$
4. Otherwise, if $A[\lfloor n/2 \rfloor - 1] < A[\lfloor n/2 \rfloor]$ then return Fast-Peak-Finding($A[0, \lfloor n/2 \rfloor - 1]$)
5. else return $\lfloor n/2 \rfloor + 1 +$ Fast-Peak-Finding($A[\lfloor n/2 \rfloor + 1, n - 1]$)

1. Give an example instance of length 8 on which this algorithm fails.

2. Consider now the correct version of Fast-Peak-Finding given in the lecture. Suppose that we replaced the $\lfloor n/2 \rfloor$ by $\lfloor n/10 \rfloor$ throughout the algorithm. Would the algorithm still work? Would it be more or less efficient?

5 Finding Two Peaks (optional and very difficult!)

We are given an integer array $A$ of length $n$ that has exactly two peaks. The goal is to find both peaks. We could do this as follows: Simply go through the array with a loop and check every array element. This strategy has a runtime of $O(n)$ (requires $c \cdot n$ array accesses, for some constant $c$). Is there a faster algorithm for this problem (e.g. similar to Fast-Peak-Finding)? If yes, give such an algorithm. If no, justify why there is no such algorithm.