**Problems Solved:**

| 36 | 37 | 38 | 39 | 40 |
|---|---|---|---|---|

**Name:**

**Matrikel-Nr.:**

**Problem 36.** Is there a Turing machine $M$ over the alphabet $\Sigma = \{0, 1\}$ that can multiply two 128 bit integers in $O(1)$ steps? The Turing machine would get the binary representations of two integers, i. e., two words of length 128, as input and has to produce the product in binary form as output. Justify your answer. If the answer is *yes*, describe how the multiplication is done and find an upper bound for the number of steps, if the answer is *no*, explain why it is not possible.

**Problem 37.** True or false?

1. $5n^2 + 7 = O(n^2)$
2. $5n^2 = O(n^3)$
3. $4n + n \log n = O(n)$
4. $(n \log n + 1024 \log n)^2 = O(n^2 (\log n)^3)$
5. $3^n = O(9^n)$
6. $9^n = O(3^n)$

Prove your answers based on the formal definition of $O(f(n))$, i. e., for all functions $f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$ we have

$$g(n) = O(f(n)) \iff \exists c \in \mathbb{R}_{>0} : \exists N \in \mathbb{N} : \forall n \geq N : g(n) \leq c \cdot f(n).$$

**Problem 38.** Let $f, g, h : \mathbb{N} \to \mathbb{R}_{\geq 0}$. Prove or disprove based on Definition 45 from the lecture notes.

1. $f(n) = O(f(n))$
2. $f(n) = O(g(n)) \implies g(n) = O(f(n))$
3. $f(n) = O(g(n)) \wedge g(n) = O(h(n)) \implies f(n) = O(h(n))$

**Problem 39.** Write a LOOP program in the core syntax (variables may be only incremented/decremented by 1) that computes the function $f : \mathbb{N} \to \mathbb{N}$, $f(n) = 2^n$.

1. Count the number of variable assignments (depending on $n$) during the execution of your LOOP program with input $n$.
2. What is the time complexity (the asymptotic complexity of the number of variable assignments) of your program (depending on $n$)?
3. Is it possible to write a LOOP program with time complexity better than $O(2^n)$? Give an informal reasoning of your answer.

4. Optional. Let $l(k)$ denote the bit length of a number $k \in \mathbb{N}$. Let $b = l(n)$, i.e., $b$ denotes the bit length of the input. What is the time complexity of your program depending on $b$, if every variable assignment $x_i := x_j + 1$ costs time $O(l(x_j))$?

   *Hint:* You must determine an $O$-notation for $s(n) = \sum_{k=0}^{2^n-1} l(k)$. Split this sum into $s(n) = \sum_{k=0}^{2^{n-1}-1} l(k) + \sum_{k=0}^{2^{n-1}-1} l(2^{n-1}+k)$. The number of bits of each term of the second sum is easy to determine. Compare the first sum with $s(n-1)$. Then continue by expanding $s(n-1)$ in the same way.

**Problem 40.** Define *concrete* languages $L_i$ $(i = 1, \ldots, 4)$ over the alphabet $\Sigma = \{0, 1\}$ such that $L_i$ has infinitely many words and $L_i \neq \Sigma^*$. The following properties must be fulfilled.

(i) There exists a (deterministic) Turing machine $M_1$ with $L_1 = L(M_1)$ such that there is a constant $K \in \mathbb{N}$ and every word $w \in L_1$ is accepted in less then $K$ steps.

(ii) Every (deterministic) Turing machine $M_2$ with $L_2 = L(M_2)$ needs at least $n$ steps to accept a word $w \in L_2$ with $|w| = n \in \mathbb{N}$.

(iii) Every (deterministic) Turing machine $M_3$ with $L_3 = L(M_3)$ needs at least $n^2$ steps to accept a word $w \in L_3$ with $|w| = n \in \mathbb{N}$.

(iv) Every (deterministic) Turing machine $M_4$ with $L_4 = L(M_4)$ needs at least $2^n$ steps to accept a word $w \in L_4$ with $|w| = n \in \mathbb{N}$.

By *concrete* language it is meant that your definition defines an explicit set of words (preferably of the form $L_i = \{w \in \Sigma^* \mid \ldots\}$) and not simply a class from which to choose. In other words,

> Let $L_1 \neq \Sigma^*$ be an infinite language such that (i) holds.

does not count as a *concrete* language.
In each case (informally) argue why your language fulfills the respective conditions.
Note that the exercise asks about acceptance of a word, not the computation of a result.