Problems Solved:

36	37	38	39	40

Name:

Matrikel-Nr.:

Problem 36. Which of the following statements are true? Justify your answer.

1. $\log(n^{100})$ is $O(\sqrt{n})$
2. $\varphi(n^{-100})$ is $O(n)$ where $\varphi(x) = 10^x$.
3. $n^2 - 2n$ is $O(n)$
4. For all $\varepsilon > 0$: $\sqrt{e^n}$ ist $O(e^{\varepsilon n})$
5. There exists $\varepsilon > 0$ and $k \in \mathbb{N} \setminus \{0\}$ such that $\mathrm{e}^{\varepsilon n}$ is $O(n^k)$.
6. For all $\varepsilon > 0$ and for all $k \in \mathbb{N} \setminus \{0\}$: $\mathrm{e}^{\varepsilon n}$ is $O(k^n)$.
7. 2^n is $O(8^n)$
8. 8^n is $O(2^n)$

Prove at least one of 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 37. 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 38. 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 39. Let $\Sigma = \{0, 1\}$ and let $L \subseteq \Sigma^*$ be the set of binary numbers divisible by 3, i.e.,

$$L = \{x_n \dots x_1 x_0 \ : \ 3 \text{ divides } \sum_{k=0}^{n} x_k 2^k\}.$$

(By convention, the empty string ε denotes the number 0 and so it is in L too.)

1. Design a Turing machine M with input alphabet Σ which recognizes L, halts on every input, and has (worst-case) time complexity $T(n) = n$. Write down your machine formally. (A picture is not needed.) *Hint:* Three states q_0, q_1, q_2 suffice. The machine is in state q_r if the bits read so far yield a binary number which leaves a remainder of r upon division by 3. The transition from one state to another represents a multiplication by 2 and the addition of 0 or 1.

2. Determine $S(n)$, $\bar{T}(n)$ and $\bar{S}(n)$ for your Turing machine.

3. Is there some faster Turing machine that achieves $\bar{T}(n) < n$? (Justify your answer.)

Problem 40. Define *concrete* languages L_i $(i = 1, \dots, 4)$ 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 (deterministic) Turing machine M_1 with $L_1 = L(M_1)$ such that every word $w \in L_1$ is accepted in $O(1)$ steps.

(ii) Every (deterministic) Turing machine M_2 with $L_2 = L(M_2)$ needs at least $O(n)$ 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 $O(n^2)$ 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 $O(2^n)$ 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 $L_i = \{w \in \Sigma^* \mid \dots\}$) 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.