Problems Solved:

36	37	38	39	40

Name:

Matrikel-Nr.:

Problem 36. 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 you have to show that your language fulfills the respective conditions.
Note that the exercise asks about acceptance of a word, not the computation of a result.

Problem 37. Is there a Turing machine M over the alphabet $\Sigma = \{0, 1\}$ that can multiply two 64 bit integers in $O(1)$ steps? The Turing machine would get the binary representations of two integers, i. e., two words of length 64, as input and has to produce the product in binary form as output. Justify your answer, i. e., if the answer is *yes*, describe how the multiplication is done, if the answer is *no*, explain why it is not possible.

Problem 38. Given two algorithms A and B for computing the same problem. For their time complexity we have

$$t_A(n) = \sqrt{n} \quad \text{and} \quad t_B(n) = 2^{\sqrt{\log_2 n}}.$$

1. Construct a table for $t_A(n)$ and $t_B(n)$. Can you give a value for n after which one of the algorithms always seems faster than the other one?
2. Based on your result of the question above, you may conjecture $t_A(n) = O(t_B(n))$ and/or $t_B(n) = O(t_A(n))$. Prove your conjecture(s) formally on the basis of the O notation.

Hint: remember that for all $x, y > 0$ we have

$$x = 2^{\log_2 x}$$

$$\log_2 x^y = y \cdot \log_2 x$$

$$\sqrt{x} = x^{\frac{1}{2}}$$

$$x \leq y \ \Rightarrow \ 2^x \leq 2^y$$

which may become handy in your proof.

Problem 39. 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 40. Analyze the time and space complexity of the following program:

```
n = read()
p = 1
while n > 0
    p = 2 * p
    n = n - 1
q = 1
while p > 0
    q = 2 * q
    p = p - 1
write(q)
```

Specify the asymptotic time and space complexity of the program depending on the input N by Θ-notation.
Note: The time complexity is considered to be the number of lines executed, and the space complexity is the number of variables used during the execution.