Problems Solved:

26	27	28	29	30

Name:

Matrikel-Nr.:

Problem 26. Consider the following term rewriting system:

$$\begin{array}{ll} \text{(Rule 1)} & p(x, s(y)) \to p(s(s(x)), y) \\ \text{(Rule 2)} & p(x, 0) \to s(x) \end{array}$$

1. Show that

$$p(0, s(p(0, s(0)))) \xrightarrow{*} s(s(s(s(s(s(s(s(s(0)))))))))$$

   by a suitable reduction sequence. For each reduction step, underline the subterm that you reduce, and indicate the reduction rule and the matching substitution σ used explicitly.

2. Prove or disprove (an informal argument suffices)

$$p(0, p(0, p(0, p(0, s(0))))) \xrightarrow{*} \underbrace{s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(}_{16 \times s} 0)))))))))))))))).$$

Problem 27. Construct a DFSM recognizing $L(G)$ where $G = (\{A, B\}, \{a, b, c\}, P, A)$ with the production rules P given by

$$\begin{aligned} A &\to aA|bA|cA|bB|cB, \\ B &\to cA|cB|b|c. \end{aligned}$$

Hint: Start by a constructing a NFSM N. Then turn N into a DFSM D such that $L(G) = L(N) = L(D)$.
"Construct" means to explain how you turn the grammar into a DFSM. Simply writing down a DFSM D with the required property, does not count as a solution unless you *prove* that $L(G) = L(D)$.

Problem 28. Let $Q(x) = \left\{ y \in \mathbb{N} \,\middle|\, x \leq y^2 \right\} \subseteq \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{N}$ be the (partial) function

$$f(x) = \begin{cases} \min Q(x) & \text{if } Q(x) \neq \emptyset, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

1. Is f LOOP-computable?
2. Is f a primitive recursive function?
3. Is f a WHILE-computable function?
4. Is f a μ-recursive function?

In each case justify your answer. If it is *yes*, give a corresponding program and/or an explicit definition as a (primitive/μ-) recursive function.
Remark: When defining f, you are allowed to use the Definition 29 and 30 from the lecture notes and the primitive recursive functions (respectively loop programs computing these functions)

$$m : \mathbb{N}^2 \to \mathbb{N}, \quad (x, y) \mapsto x \cdot y$$

$u : \mathbb{N}^2 \to \mathbb{N}$,

$$u(x, y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{if } x \neq y. \end{cases}$$

and $IF : \mathbb{N}^3 \to \mathbb{N}$,

$$IF(x, y, z) = \begin{cases} y & \text{if } x = 0 \\ z & \text{otherwise.} \end{cases}$$

Other functions or rules are forbidden.

Problem 29. According to Definition 32 of the lecture notes, there are no natural numbers in Lambda calculus. However, natural numbers can be encoded (known as Church encoding) as “Church numerals” (see below), i.e., as functions $\mathbf{n}$ that map any function f to its n-fold application $f^n = f \circ \ldots \circ f$. Note that we denote such a “natural number” representation via boldface symbols in order to emphasize that these are lambda terms. In other words, we define Church numerals as follows. By letting “application” bind stronger than “abstraction”, we avoid writing parentheses where appropriate.

$$\begin{aligned}
\mathbf{0} &= \lambda f.\lambda x.x \\
\mathbf{1} &= \lambda f.\lambda x.fx \\
\mathbf{2} &= \lambda f.\lambda x.f(fx) \\
\mathbf{3} &= \lambda f.\lambda x.f(f(fx)) \\
\mathbf{4} &= \lambda f.\lambda x.f(f(f(fx))) \\
&\vdots \\
\mathbf{n} &= \lambda f.\lambda x.\underbrace{f(\cdots(f}_{n\text{-fold}}\,x)\cdots)
\end{aligned}$$

1. Define a lambda term add that represents addition of “Church numerals”.
2. Show the intermediate steps of a reduction from ((add $\mathbf{2}$) $\mathbf{1}$) to $\mathbf{3}$.

Hint: a bit of literature research may help.

Problem 30. Let $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and

$$e = \sum_{k=0}^{\infty} \frac{1}{k!}.$$

Furthermore let for a natural number $n > 0$ the word w_n be given as the first n digits after the comma in the decimal expansion of e and $L = \{w_n \mid n \in \mathbb{N} \setminus \{0\}\}$.

(a) Is there a grammar G with Σ as the set of terminal symbols such that $L(G) = L$?

(b) Is L a context-free language?

Prove your answers!
Hint: Please note that e can be computed with arbitrary precision by evaluating a sufficiently large number of summands in the summation term.
For the proof of the second part of the task you have to look into additional literature. In particular, look for the Pumping Lemma for context-free languages.