Problems Solved:

16	17	18	19	20

Name:

Matrikel-Nr.:

Problem 16. Let $\Sigma = \{1\}$. Is the function $f : \Sigma^* \to \Sigma^*$ where $f(1^n) = 1^{2n}$ Turing-computable? If yes, construct a Turing machine $M = (Q, \Sigma, \Gamma, q_0, F, \delta)$ which computes f and explain its flow. If no, i.e., f is not Turing-computable, justify your answer.
Note that 1^n means a sequence of n symbols 1.

Problem 17. Write a RAM program that computes the integer absolute value function $f : \mathbb{Z} \to \mathbb{Z}$ defined by

$$f(n) = \begin{cases} n, & \text{if } n \geq 0; \\ -n, & \text{otherwise} \end{cases} \quad (1)$$

such that the program reads an integer n from the input tape and (upon termination) has written $f(n)$ to its output tape.
Note that an easy way to test whether an input value is negative is by subtracting 0 from it and testing whether the result is 0.

Problem 18.

Definition 1 (RAM computable)**.** We say that a partial function $f : \mathbb{N} \to_P \mathbb{N}$ is *RAM computable* if there exists a RAM R that such that

- R terminates for input $n \in \mathbb{N}$ if and only if $n \in \mathit{domain}(f)$;
- R terminates for input $n \in \mathbb{N}$ with output n' if and only if $n' = f(n)$.

Show that every loop computable function is also RAM computable by describing how the loop program computing the function can be translated to a RAM program.
This amounts to translating every construct in the language of loop programs into a corresponding RAM program fragment. Here every variable of the loop program is represented by one register in the RAM. Furthermore, every loop requires an additional register to keep track of the number of iterations still to be performed (but consider that loops may be nested!). Give your constructions assuming that the loop program requires variables $x_0, ...x_{m-1}$ and has a "loop depth" of at most d.

Problem 19. Provide a loop program that computes the function $f(n) = \sum_{k=1}^{n} k(k+1)$, and thus show that f is loop computable.

Problem 20. Suppose P is a while-program that does not contain any `WHILE` statements, but might modify the values of the variables x_1 and x_2.
Transform the following program into Kleene's normal form.
Hint: first translate the program into a goto program, replace the `GOTO`s by assignments to a control variable, and add the `WHILE` wrapper.

```
x0 := 0
WHILE x1 DO
    x1 := x1 - 1;
    x2 := x1;
    WHILE x2 DO
        P;
    END;
END;
x0 := x0 + 1
```