Problems Solved:

21	22	23	24	25

Name:

Matrikel-Nr.:

RAMRegular

Problem 21. Let $\Sigma = \{a, b\}$. We encode a and b on the input tape of a RAM by 1 and 2 and a word $w \in \Sigma^*$ by a respective sequence of 1's and 2's.
We say that a RAM R *accepts* a word $w \in \Sigma^*$ if R starts with the coded word w on its input tape and terminates after having written a non-zero number on its output tape. We define $L(R) := \{w \in \Sigma^* \mid R \text{ accepts } w\}$.
Let F be a RAM that terminates for every input and whose program does not contain "loops", i.e., each instruction is executed at most once.
Derive answers for the following questions. (Give ample justifications, just saying 'yes' or 'no' is not enough.)

1. Is $L(F)$ as a language over Σ finite?
2. Is $L(F)$ as a language over Σ regular?

Solution of Problem 21:

1. No. The Program (`LOAD #1; OUT; JUMP 0`) accepts all possible words.
2. Yes. The RAM can only read finitely many letters from the input tape, since there are only finitely many commands that can be executed. Let's say n is the number of IN commands. So after reading at most n letters, the RAM must have decided whether or not the word is accepted. There are only finitely many words of length at most n. Let $w_1, w_2, \ldots, w_k$ be the words of length at most n that will be accepted. Among those are words w.l.o.g. $w_1, w_2, \ldots, w_r$ with $r \leq k$ for which the RAM will read a space on the input tape after the word. So the input word has ended. For the words $w_{r+1}, \ldots, w_k$ it is not clear whether some further letters follow. So the corresponding regular expression is.

$$w_1 + \cdots + w_r + (w_{r+1} + \cdots + w_k) \cdot (a + b)^*.$$

RAMBinary

Problem 22. Write a RAM program that from a given natural number n prints its binary representation. In order to simplify the problem the output shall be in low positions first format, i.e., the number 8_{10} is 0001_2 but not 1000_2.
Hint: please note that the computation of the quotient respectively remainder of a division by 2 can be implemented by the repeated subtraction of 2.

Solution of Problem 22:

```
START:  LOAD #0    ; Load 0 to the accum
        STORE 2    ; write 0 to R[2], d := 0
        IN         ; R[1] := n
DIV2:   BEQ 0, WRITE
        BEQ 1, WRITE
```

```
        SUB #2    ; n := n-2
        STORE 1   ; R[1] := n
        LOAD 2    ; load d into accum
        ADD #1    ; d := d+1
        STORE 2   ; R[1] := d
        LOAD 1    ; load new n into accum
        JUMP DIV2
WRITE:  OUT       ; write 0 or 1 to the output
        LOAD 2    ; load d into accum
        BEQ 0, END
        BEQ 1, PRINTd
        STORE 1   ; n := d
        LOAD #0
        STORE 2   ; d := 0
        LOAD 1    ; load n into accum
        JUMP DIV2
PRINTd: OUT       ; write 1 to the output
END:    JUMP 0
```

LoopDiv

Problem 23. In the following use *only* the definition of a *loop program* as given in Def. 23 of the lecture notes, Section 3.2.2. Note that it is not allowed to use abbreviations like

$x_i := x_j - x_k$;
$x_i := x_j + x_k$;

Furthermore, the variables in a loop program are only $x_0, x_1, \dots$

1. Show that the function

$$s(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 < x_2, \\ 0 & \text{otherwise} \end{cases}$$

   is loop computable. I.e. give an explicit loop program for s.

2. Write a loop program that computes the function $d : \mathbb{N}^2 \to \mathbb{N}$ where $d(x_1, x_2)$ is $k \in \mathbb{N}$ such that $k \cdot (x_2 + 1) = x_1 + 1$ if such a k exists. The result is $d(x_1, x_2) = 0$, if a k with the above property does not exist.

   For simplicity in the program for d, you are allowed to use a construction like the following (with the obvious semantics) where P is an arbitrary loop program.

   IF $x_i < x_j$ **THEN** P **END**;

   Note: Only $<$ is allowed in the condition and there must be a variable before and a variable after the $<$ sign and $1 < x_j$ is not allowed. Note also that there is no "ELSE" branch.

Solution of Problem 23:

$s(x_1, x_2) ==$

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

$d(x_1, x_2) ==$

```
x1 := x1+1;
x2 := x2+1;
x3 := 0;
x0 := 0;
LOOP x1 DO
    IF x3 < x1 THEN
        LOOP x2 DO x3 := x3 + 1; END;
        x0 := x0 + 1;
    END;
END;
IF x1 < x3 THEN
    x0 := 0;
END;
```

LoopSum

Problem 24. 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.
You are only allowed to use the constructs given in Definition 23 of the lecture notes.

Solution of Problem 24:

```
x0 := 0;
LOOP x1 DO
    LOOP x1 DO LOOP x1 DO x0 := x0 + 1 END END;
    LOOP x1 DO x0 := x0 + 1 END;
    x1 := x1 - 1
END
```

WhileKleene

Problem 25. 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
```

Solution of Problem 25:

First the translation into the goto-program.

```
x0 := 0
L10: IF x1 = 0 GOTO L11;
     x1 := x1 - 1;
     x2 := x1;
L20: IF x2 = 0 GOTO L21;
     P;
     GOTO L20;
L21: GOTO L10
L11: x0 := x0 + 1
```

Replace goto by assignment to x_c and add more labels.

```
L1: x0 := 0
L10: IF x1 = 0 THEN xc := 11;
L2: x1 := x1 - 1;
L3: x2 := x1;
L20: IF x2 = 0 THEN xc := 21;
L4: P;
L5: xc := 20;
L21: xc := 10;
L11: x0 := x0 + 1
```

Add the while wrapper.

```
xc :=
WHILE xc DO
    IF        xc =  1 THEN xc := 10; x0 := 0
    else IF   xc = 10 THEN xc :=  2; IF x1 = 0 THEN xc := 11; END;
    else IF   xc =  2 THEN xc :=  3; x1 := x1 - 1;
    else IF   xc =  3 THEN xc := 20; x2 := x1;
    else IF   xc = 20 THEN xc :=  4; IF x2 = 0 THEN xc := 21; END;
    else IF   xc =  4 THEN xc :=  5; P;
    else IF   xc =  5 THEN xc := 21; xc := 20;
    else IF   xc = 21 THEN xc := 11; xc := 10;
    else IF   xc = 11 THEN xc :=  0; x0 := x0 + 1;
    END IF;
END;
```