Problems Solved:

41	42	43	44	45

Name:

Matrikel-Nr.:

Problem 41. Let $T(n)$ be total number of calls to `tick()` resulting from running `P(n)`.

```
procedure P(n)
    k = 0
    while k < n do
        tick()
        P(k)
        k = k + 1
    end while
end procedure
```

1. Compute $T(0), T(1), T(2), T(3), T(4)$.
2. Give a recurrence relation for $T(n)$. (It is OK if your recurrence involves a sum.)
3. Give a recurrence relation for $T(n)$ that does not involve a sum. (*Hint:* Use your recurrence relation (twice) in $T(n+1) - T(n)$.)
4. Solve your recurrence relation. (It is OK to just guess the solution as long as you prove that it satisfies the recurrence.)

Problem 42. Let $T(n)$ be given by the recurrence relation

$$T(n) = 3T(\lfloor n/2 \rfloor).$$

and the initial value $T(1) = 1$. Show that $T(n) = O(n^\alpha)$ with $\alpha = \log_2(3)$. *Hint:* Define $P(n) :\iff T(n) \leq n^\alpha$. Show that $P(n)$ holds for all $n \geq 1$ by induction on n. It is not necessary to restrict your attention to powers of two.

Problem 43. Let $T(n)$ be number of times that line 2 is executed in the worst case while running $P(a, b)$ where $n := b - a$.

```
1 procedure P(int a, int b, int foo[])
2     if (a + 1 < b) {
3         int h = floor( (a + b) / 2 );
4         if foo[h] >= 0 then P(a, h)
5         if foo[h] <= 0 then P(h, b)
6     }
7 end procedure
```

1. Compute $T(1)$, $T(2)$, $T(3)$ and $T(4)$.
2. Give a recurrence relation for $T(n)$.
3. Solve your recurrence relation for $T(n)$ in the special case where $n = 2^m$ is a power of two.

4. Use the Master Theorem to determine asymptotic bounds for $T(n)$.

Note that `floor` denotes the function that returns the biggest integer value that is smaller than or equal to the argument.

Problem 44. Let X be a monoid. Device an "algorithm" (as recursive/iterative pseudo-code in the style of Chapter 6 of the lecture notes) for the computation of x^n for $x \in X, n \in \mathbb{N}$ that uses less multiplications than the naive algorithm of n times multiplying x to the result obtained so far. Determine the complexity as $M(n)$, i.e., the number of multiplications of your "algorithm" depending on the exponent n.
Hint: Note that x^8 can be computed with just 3 multiplications while the naive algorithm would use 7 multiplications. Based on this observation, the algorithm can be based on a kind of "binary powering" strategy.

Problem 45. Let M be a Turing machine over the alphabet $\{0, 1\}$ that takes as input a string $b_1 b_2 \dots b_n$ ($b_i \in \{0, 1\}$), prepends an additional 1 to the string and then interprets the result $1b_1 b_2 \dots b_n$ as the binary representation of a number k. M then writes out the unary representation of k (consisting of a string of k letters 1) onto the tape and stops.
Note that in the above description it is not said *how* M computes the result. In particular M need not be the most efficient Turing machine fulfilling the above specification.

1. Give a reasonably close asymptotic lower-bound for the space- and time-complexity $S(n)$ and $T(n)$ for the execution of the task and justify these bounds (without giving a detailed construction of M). Choose adequate Landau-symbols for formulating the bounds.

2. Give an informal description of a (reasonably efficient) Turing machine M' that performs the task described above. Analyze the space and time complexity $S(n)$ and $T(n)$ and write down an upper/exact asymptotic bound for these complexities. Again choose adequate Landau symbols for formulating the bounds.

   Hint: Let M' apply the *binary powering* strategy.