Problems Solved:

41	42	43	44	45

Name:

Matrikel-Nr.:

Problem 41. Let $L = \left\{ww^{-1} \,\middle|\, w \in \{0,1\}^*\right\}$ be the language of palindromes. Here w^{-1} denotes the "mirrored" word, i. e., if $w = l_1 l_2 \cdots l_r$ then $w^{-1} = l_r \ldots l_2 l_1$.

- Describe (informally) a Turing machine M with $L(M) = L$.
- Analyse the time and space complexity of M.

Problem 42. 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 43. 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 44. 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 int foo[] = ... // array of big enough size
2 procedure P(int a, int b)
3     if (a + 1 < b) {
4         int h = floor( (a + b) / 2 );
5         if foo[h] >= 0 then P(a, h)
6         if foo[h] <= 0 then P(h, b)
7     }
8 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 45. 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 N such that for all $n \geq N$ 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.