Problems Solved:

46	47	48	49	50

Name:

Matrikel-Nr.:

Problem 46. Consider the program

```
f(n) ==
    return g(n, 0, 0, 0)
g(n, m, v, s) ==
    if m > n then
        return s
    else
        return g(n, m + 1, 2 * (v + (2 * m + 1) * 2^m), s + v)
```

which computes a function $f : \mathbb{N} \to \mathbb{N}$.

1. Show that
$$v = 2^m m^2$$
holds true for every call $g(n, m, v, s)$ to function g in the execution of $f(n)$. *Hint:* Use induction on the number of (nested) function calls to g.

2. Show that
$$s = \sum_{k=0}^{m-1} k^2 2^k$$
holds true for every call $g(n, m, v, s)$ in the execution of $f(n)$.

   Hint: Again, use induction on the number of calls to g. In the induction step, you may want to use the result of Part 1.

3. From Part 2, one may deduce $f(n) = \sum_{k=0}^{m} k^2 2^k$.

   Show by induction on n that $f(n) = 2^{n+1}(3 - 2n + n^2) - 6$.

Problem 47. Take that recursive program

```
f(n,b) ==
     if n < 1 then return 0
     d := floor(n/3)
     return b + f(d,1) + 2*f(d,2)
```

Let $C(n)$ be the number of comparisons executed in the first line of the function body while running $f(n, 0)$ for some positive integer n.

1. Write down a recurrence for C and determine enough initial values.

2. Solve that recurrence for the given initial values and arguments n of the form $n = 3^m$.

3. Prove by induction that your solution is correct.

Problem 48. Prove or disprove the following:

1. $O(g(n))^2 = O(g(n)^2)$
2. $2^{O(g(n))} = O(2^g(n))$

Hint: First transform above equations into a form that does not involve the O-notation on the left hand side, then prove the correctness of the resulting formulas.

Problem 49. An n-bit binary counter counts in $2^n - 1$ steps from $(00\ldots0)_2 = 0$ to $(11\ldots1)_2 = 2^n - 1$ and in one more step back to $(00\ldots0)_2 = 0$. The cost of a step is the number of bits changed at that step. (For instance, the cost of increasing a 4-bit counter from `1011` to `1100` is 3 since 3 bits are modified.)

1. Consider how often bit position i changes in the 2^n cycles and compute the sum of the number of changes of all positions. The amortized cost is this sum divided by the number of cycles.
2. Compute the amortized cost by applying the potential method.

   Use as the potential $\Phi(a_i)$ of counter a_i after the i-th application of the increment operation $\Phi(a_i) = b(a_i)$ where $b(a_i)$ is the number of 1s in the binary representation of the counter.

   For the computation of an upper bound of the amortized cost $\hat{c}_i$ derive inequalities $b(a_i) \leq \ldots$ and $c_i \leq \ldots$ using the notion $t(a_i)$ for the number of bits reset from 1 to 0 by the i-th increment operation.

Problem 50. Consider a RAM program that evaluates the value of $\sum_{i=1}^{n} i^2$ in the naive way (by iteration). Analyze the worst-case asymptotic time and space complexity of this algorithm on a RAM assuming the existence of operations operation `ADD r` and `MUL r` for the addition and multiplication of the accumulator with the content of register r.

1. Determine a Θ-expression for the number $S(n)$ of registers used in the program with input n (space complexity).
2. Compute a Θ-expression for the number $T(n)$ of instructions executed for input n (time complexity in constant cost model),
3. Assume the logarithmic cost model of a RAM, i.e., the cost of an operation is proportional to the length of the arguments involved. Example: If a is the (bit) length of the accumulator and l is the (bit) length of the content of register r then `MUL r` costs $a + l$ and `ADD r` costs $\max a, l$.

   Compute the assymptotic costs $C(n)$ (using O-notation) of the algorithm for input n.