Problems Solved:

46	47	48	49	50

Name:

Matrikel-Nr.:

Problem 46. Consider the program

```
static void permutations(char[] a, int i)
  {
    if (i >= a.length)
    {
      System.out.println(new String(a));
      return;
    }
    for (int j=i; j<a.length; j++)
    {
      swap(a, i, j);
      permutations(a, i+1);
      swap(a, i, j);
    }
  }
```

where a call `permutations(s.toCharArray(), 0)` prints all permutations of string s.

1. Draw a recursion tree for a call `permutations(<abcd>,0)` (where `<abcd>` denotes the character array with elements a, b, c, d). What is the height of the tree? What is the number of nodes in every level of the tree? What is the total number of nodes in this tree?

2. Give a recurrence for the total number $S(n)$ of calls of `swap` in an execution of `permutations(a,i)` where $n =$ `a.length-i`.

3. Give a summation formula for $S(n)$ by adding the number of calls in every level of the recursion tree.

4. Give a complexity estimation $S(n) = O(T(n))$ for some closed formula n (justify your estimation semi-formally).

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. 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 49. Consider the following program as an informal sketch of an underlying RAM program which is to be analyzed in the logarithmic cost model. Analyze the time and space complexity:

```
n = read()
p = 1
while n > 0
    p = 2 * p
    n = n - 1
q = 1
while p > 0
    q = 2 * q
    p = p - 1
write(q)
```

Specify the asymptotic time and space complexity of the program depending on the input N by Θ-notation.

Problem 50. Consider a RAM program that evaluates the value of $n! = \prod_{i=1}^{n} i$ 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. Determine a Θ-expression for the number $T(n)$ of instructions executed for input n (time complexity in constant cost model),

3. Determine a O-expression for the asymptotic time complexity $C(n)$ of the algorithm for input n assuming the logarithmic cost model for a RAM (with cost $al * rl$ for operation `MUL r` where al is the digit length of the content of the accumulator and rl is the digit length of the content of register r).

   Hint: approximate in the asymptotic analysis summation $\sum_{i=a}^{b} T$ by integration $\int_a^b T \mathrm{d}i$ and solve this integral; you may use a computer algebra system or WolframAlpha for this purpose.