Problems Solved:

46	47	48	49	50

Name:

Matrikel-Nr.:

Problem 46. Consider the following Java methods:

```
1  int a(int n) { return b(n); }
2  int b(int n) {
3    if (n<1) {return -1;}
4    int p = b(n-1);
5    int q = b(n-2);
6    int r = b(n-2);
7    return p+q+r; }
```

Let f_n be the number of calls to method `b` which result from evaluating `a`(n) where $n \geq 0$. (We assume that no optimizations are made. In particular, *both* of the function calls `b(n-2)` in lines 5 and 6 are executed.)

1. Compute f_n for $n = 0, 1, \ldots, 5$.
2. Give a recurrence relation for f_n.
3. Let
$$F(z) = \sum_{n=0}^{\infty} f_n z^n = 1 + z + 4z^2 + \ldots$$
be the *generating function* of the sequence f_n. (Assume that this sum converges; for $|z| < 1/2$ it does.) Show that the generating function $F(z)$ satisfies the equation
$$F(z) = zF(z) + 2z^2 F(z) + \frac{1}{1-z} - z. \tag{1}$$
Hint: Multiply both sides of your recurrence equation by z^n; then sum on n. Rewrite your resulting sums in terms of $F(z)$.
4. Solve the equation above for $F(z)$ an perform a partial fraction decomposition on your result. *Hint:* The correct result is
$$F(z) = \frac{1}{1-2z} + \frac{1}{2} \cdot \frac{1}{1+z} - \frac{1}{2} \cdot \frac{1}{1-z}.$$
Your answer should show your calculation to get this result.
5. Find a closed form expression (i.e. a formula) for f_n.
Hint: Apply the geometric series
$$\frac{1}{1-az} = \sum_{k=0}^{\infty} a^k z^k.$$
to each summand in the solution for $F(z)$ in Part 4 with $a = 2$, $a = -1$ and $a = 1$ and bring the result in a form that matches
$$F(z) = \sum_{n=0}^{\infty} f_n z^n$$
in order to find f_n.

Problem 47. Does there exist for every finite language $L \subseteq \{0,1\}^*$ a Turing machine D such that D

1. takes as input the code $\langle M \rangle$ of a Turing machine M that stops on every input and
2. decides whether $L \subseteq L(M)$ holds?

Justify your answer.

Problem 48. 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 program assuming the existence of operations `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 a simplified version of the logarithmic cost model of a RAM where the cost of every operaton is proportional to the length of the arguments involved. In particular, 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 asymptotic costs $C(n)$ (using O-notation) of the program for input n.

Problem 49. Show that the language

$$L'_{u,\varepsilon} = \{\langle M \rangle \mid M \text{ does not accept } \varepsilon\}$$

is not recursively enumerable.

Problem 50. Consider the following pseudo code of an implementation of a FIFO (first in first out) queue with two functions enqueue and dequeue.

```
input  := EMPTYLIST
output := EMPTYLIST
function enqueue(e, input, output) { push(e, input) }
function dequeue(input, output) {
    if isempty(output) {
        while not isempty(input) { push(pop(input), output) }
    }
    pop(output)
}
```

Analyze its amortized cost of these functions by (a) the aggregate method and (b) the potential method.
Here,

- `push(e, L)` is the operation of adding an element e to the front of a list L,
- `isempty(L)` returns `TRUE` if the list L is empty,
- `pop(L)` is the operation that removes the first element of a list L and returns it.

All these operations are assumed to cost constant time.
In the code above, a queue is represented by a pair (`input`, `output`). Putting a new element into the queue via `enqueue`, first puts it to the front of `input`. Only when an element is requested via a call to `dequeue`, elements are moved from `input` to `output` list, thus effectively reversing `input` so that in total the queue returns its elements in a FIFO principle.
Hint: For the potential method you might want to consider the function Φ such that for a queue q that is represented by the pair (`input`, `output`) of two lists, $\Phi(q)$ is the size of the `input` list.