This assignment is **due on March 12**. All submitted work must be *done individually* without consulting someone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

As a first step go to the last page and read the section: Advice on how to do the home work.

Problem 1. Using O-notation, upperbound the running time of the following algorithm, where A is an array containing n integers.

```
function SUM(A)
    $answer \leftarrow 0$
    for $i \leftarrow 0; i < n;$ i++ do
        if $A[i] < i$ then
            $answer \leftarrow answer + A[i]$
    return $answer$
```

Problem 2. We want to extend the queue that we saw during the lectures with an operation GETAVERAGE() that returns the average value of all elements stored in the queue. This operation should run in $O(1)$ time and the running time of the other queue operations should remain the same as those of a regular queue.

a) Design the GETAVERAGE() operation. Also describe any changes you make to the other operations, if any.

b) Briefly argue the correctness of your operation(s).

c) Analyse the running time of your operation(s).

Problem 3. We are given a *sorted* array A containing n positive integers, and a value k. We wish to compute how many indices i and j ($i \neq j$) there are such that the product of the ith and the jth element of A is at most k, i.e., $A[i] * A[j] \leq k$. For full marks your algorithm needs to run in $O(n)$ time.

Example:
$A : [1, 1, 3, 5]$, $k = 4$: return 3

a) Design an algorithm that solves the problem.

b) Briefly argue the correctness of your algorithm.

c) Analyse the running time of your algorithm.

Advice on how to do the home work

- Assignments should be typed and submitted as pdf (no handwriting)
- When designing an algorithm or data structure, it might help you (and us) if you briefly describe your general idea, and after that you might want to develop and elaborate on details. If we don't see/understand your general idea, we cannot give you points for it.
- Be careful with giving multiple or alternative answers. If you give multiple answers, then we will give you marks only for "your worst answer", as this indicates how well you understood the question.
- Some of the questions are very easy (with the help of the lecture notes or book). You can use the material presented in the lecture or book without proving it. You do not need to write more than necessary (see comment above).
- When giving answers to questions, always prove/explain/motivate your answers.
- When giving an algorithm as an answer, the algorithm does not have to be given as (pseudo-)code.
- If you do give (pseudo-)code, then you still have to explain your code and your ideas in plain English.
- Unless otherwise stated, we always ask about worst-case analysis, worst case running times etc.
- As done in the lecture, and as it is typical for an algorithms course, we are interested in the most efficient algorithms and data structures.
- If you use further resources (books, scientific papers, the internet,...) to formulate your answers, then add references to your sources.
- If you refer to a result in a scientific paper or on the web you need to explain the results to show that you understand the results, and how it was proven.