This assignment is **due on March 24** and should be submitted on Gradescope. All submitted work must be *done individually* without consulting someone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

Before you read any further, go to the last page of this document and read the Written Assignment Guidelines section.

Problem 1. (20 points)
Consider the following algorithm that given an array A of length n produces another array B of length $n-k$ where $B[i] = \sum_{j=i}^{i+k-1} A[j]$.

```
1: function ALGORITHM(A, k)
2:     n ← length of A
3:     B ← new array of size n − k
4:     for i ∈ [0 : n − k] do                ▷ i ranges from 0 to n − k − 1
5:         B[i] ← 0
6:         for j ∈ [0 : k] do                ▷ j ranges from 0 to k − 1
7:             B[i] ← B[i] + A[i + j]
8:     return B
```

Assuming that $k < n/2$ (but not necessarily a constant), your task is to:

a) Use O-notation to upperbound the running time of the following algorithm in terms of **both** n and k.

b) Come up with a more efficient algorithm that runs in $O(n)$ time independent of the value of k.

Problem 2. (40 points)
We want to build a queue for integer elements that in addition to the operations we saw during the lecture, also supports a SEESAW() operation that on a given queue holding the elements $Q_0, Q_1, \ldots, Q_{n-1}$ returns

$$Q_0 + \frac{1}{Q_1} + Q2 + \frac{1}{Q_3} + \cdots$$

All operations should run in $O(1)$ time. Your data structure should take $O(n)$ space, where n is the number of elements currently stored in the data structure.
Your task is to:

a) Design a data structure that supports the required operations in the required time and space.

b) Briefly argue the correctness of your data structure and operations.

c) Analyse the running time of your operations and space of your data structure.

Problem 3. (40 points)
We want to build a stack for integer elements. In addition to the usual stack operations, we want to support a SILLYPROD() operation that on given stack holding the elements $S_0, S_1, \ldots, S_{n-1}$ returns

$$\sum_{i<j:S_i<S_j} S_i S_j.$$

All standard operations should run in worst-case $O(1)$ time, and SILLYPROD() should run in amortized $O(n)$ time (the amortization is over all stack operations). Your data structure should take $O(n)$ space, where n is the number of elements currently stored in the data structure.

a) Describe your algorithm in plain English.

b) Briefly argue the correctness of your data structure and operations.

c) Analyse the running time of your operations and space of your data structure.

Written Assignment Guidelines

- Assignments should be typed and submitted as pdf (no pdf containing text as images, no handwriting).
- Start by typing your student ID at the top of the first page of your submission. Do **not** type your name.
- Submit only your answers to the questions. Do **not** copy the questions.
- When asked to give a plain English description, describe your algorithm as you would to a friend over the phone, such that you completely and unambiguously describe your algorithm, including all the important (i.e., non-trivial) details. It often helps to give a very short (1-2 sentence) description of the overall idea, then to describe each step in detail. At the end you can also include pseudocode, but this is optional.
- In particular, 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 marks 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 slides 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 and explain it in your own words. Only citing a source doesn't show your understanding and will thus get you very few (if any) marks. Copying from any source without reference is considered plagiarism.