A prerequisite for the course is that you already know basic data types and algorithms. In this first tutorial, we go over some simple examples. You should be able to implement them for yourself in your favourite programming language. Check the relevant chapters of *Introduction to Algorithms* to refresh your memory. You can ask me any questions and clarifications.

This course doesn’t impose a specific programming language. We study algorithms in an informal way, using pseudo-code. But you must put the theoretical knowledge in practice by implementing everything in a language of your choice. The languages that are acceptable for these tasks are: Haskell, Python, Java, C/C++.

My own favourite language is Haskell. During the course I will sometimes give you my own realization of some of the algorithms in Haskell. If some of you come up with nicely written solutions in other languages, we may share those with everybody. Use the Moodle Forum to discuss and share solutions.

If you’re taking the Project Module COMP4040, you must use Haskell: The project consists in three programming assignments in that language.

## 1 Sorting Algorithms

You should already know the basics of programming with lists and various sorting algorithms. Review and refresh your knowledge by doing the following exercises.

- Make sure that you know how to implement (linked) lists in your chosen programming language;
- Review the implementation of basic operations on lists: appending an element at the beginning or end of a list, concatenating lists, inserting a new element into an ordered list;
- Implement *Selection Sort*;
- Implement *Insertion Sort*;
• Implement Merge Sort;
• Implement Quick Sort.

If you don’t know or don’t remember how these algorithms work, check the relevant chapters of [IA]. You can also ask me or the teaching assistants to explain them to you.

2 The Maximum Subarray Problem

Imagine that you’re given an array of numbers that can be either positive or negative. For example:

[13, −3, −25, −20, −3, −16, −23, 18, 20, −7, −5, −22, 15, −4, 7]

You must find the subarray (sequence of consecutive numbers) with the maximum sum. A subarray is specified by giving the initial and final indices, for example (3, 7) specifies the subarray [−20, −3, −16, −23, 18] (the indices start from 0). Its sum is −44: definitely not the maximum we can get. The indices (10, 13) specify the subarray [12, −5, −22, 15] with sum 0. Which is the subarray with the largest sum?

• Implement the naive algorithm that was explained in the lecture;
• Implement a divide-and-conquer algorithm, as was described in the lecture and given in pseudocode in the book IA, Section 4.1;
• Design and implement a linear-time algorithm, following the hints given in the lecture and in Exercise 4.1-5 of IA.

Experiment with the algorithms that you implemented: generate large lists of random numbers and check how fast each of the three runs. Think about how to determine the complexity of each algorithm theoretically.