Prerequisites

Guide for Linux / Mac users. If you are running Linux or Mac machine, there are good chances you have your software development environment already set up. Open terminal and check if your gcc is installed by checking its version: 'gcc -v'. If this does not resolve into an error, most probably you are all set for this problem set, otherwise either contact teaching staff or install dependencies on your own using your package manager. You will need support for 'gcc' and 'make'.

Guide for Windows users. If you are running Windows machine, a suggestion is to install a Linux terminal emulator, such as **msys2** or **cygwin**. Easy to follow instructions on the environment setup are availabe here.

Compiling / Modifying / Running challenges

Each challenge has the following folder structure:

```
challenge
- inc       # include files (headers)
- lib       # libraries (dependencies)
- src       # source files
- res       # files related to the input/output of the application
- out       # folder holding the compiled elf
- Makefile  # "compilation rules"
```

To compile the project open your terminal, go to the challenge folder (where the Makefile resides) and execute 'make' command. If the compilation is successful it should make folder called "out" with an executable "main.elf". The following are the makefile commands at your disposal.

```
make        # compile the project
make clean  # "clean" the project compile (delete object files and executable)
```

You can and must modify the source code to comply with the challenge's requirements, but please read the comments, they might have some helpful hints.

Submission

The submission process begins starting from the publishing of the problem set. The solution should be provided as an archive (zip) of the code base. Tables or any other non-code devilerables can be filled manually and submitted as a photo or pdf. There are two dates you must be aware of: the soft and hard deadlines.

- **Soft deadline** - is in 2 weeks from the start of the problem set submission and is a mechanism of providing you with a valuable feedback. This submission will be reviewed and you will be given a feedback to improve your submission for the hard deadline.
- **Hard deadline** - is in 3 weeks from the start of the problem set submission and after this date the solutions are no longer granted with points.

BUT PLEASE NOTE THAT THE SUBMISSION OF PROBLEM SETS IS MANDATORY EVEN AFTER THE HARD DEADLINE.

Challenge 1: File Input/Output (3 points)

Write a program which would take two arguments: *path to the input file* and *path to the output file*. Assume that the input file holds some text containing a set of ASCII characters and develop a program which would read the input file, convert all characters to uppercase and write them into the output file. Please consider the provided comments in the software project.

Challenge 2: Signal filtering (7 points)

You are given two files wich contain "recorded" signals. One file contains the original sginal, while the other one has the same signal but with added noise. All sgnal values are provided in floating point format and separated by a new line character. The input/output of the application is already handled for you. Your goal is to develop a set of 1D averaging and median filters with a length of 3, 5, 7, 9 and 11 signal values. Please use files which correspond to your student ID.

- Averaging filtering

- Median filtering
- Fill the comparison table

Signal ID = <ID>	Averaging filter	Median filter
Filter length = 3		
Filter length = 5		
Filter length = 7		
Filter length = 9		
Filter length = 11		

Challenge 3: Linked lists and sorting (5 bonus points)

You are given a Comma Separater Value (CSV) file with the following population information for some of the Latvian cities (the order is preserved):

- year;
- city;
- population size;
- population before working age;
- population in working age;
- population above working age.

Each line represents a single entry in the table (or database). In this challenge you will use file to form a linked list with the respective entries, further please develop an API to display entries and order data depending on the different fields.
The challenge can be split into the following subtasks:

- write a procedure to read file and form linked list structure;
- write a procedure to print entire linked list;
- write a set of procedures to reorder list by year, population sizes;
- write a procedure to deinitialize (free) the linked structure.