# Programming Assignment 2

**Due Date:** May 16, 2021, 23:59

**Note.** Please note that this semester all assignments are group assignments. Further note that for the grading we will apply a "10%" rule, i.e. the maximum number of points for this assignments is 110, but 100 will be counted as 100%. Points that exceed 100 will be stored in a separate counter and used later for compensation of lost points in other assignments or programming assignments or (if not used up this way) the final exam.

The task of this programming assignment is to extend your C++ implementation from Programming Assignment 1 for a **registration, queueing and reporting system** for mass medical treatment (such as vaccination). The extensions concern the efficient implementation of index structures for relations that are represented by lists of blocks, to implement queries on these relations in an efficient way and to produce graphs as output.

Exercise 1. Define and implement the basic data structure for the storage of relations:

- **(i)** Design a relational database schema with relations Person, Medical_Status, Registration and Treatment capturing all data that are to be collected during registration, scheduling and treatment processing.
- **(ii)** In addition, assume that there are three types of treatments a person may register for, each with different priority rules.
- **(iii)** Represent each of these relations in a data structure that uses an ordered sequence of blocks, where each block is realised by an array.
- **(iv)** Permit an additional overflow block for each block in the sequence.
- **(v)** Implement operations
  - for the insertion and deletion of records;
  - for the reorganisation of blocks by merging, sorting and splitting;
  - for the retrieval of a record from a relation.

Exercise 2. Define and implement the basic data structure for the storage of relations:

- **(i)** Modify the operations for registration, priority calculation, appointment scheduling and reporting using the database schema.
- **(ii)** Create file output for the various reports.

In a third step you are to realise index structures that will support the fast access to the relations to facilitate the processing of the queries. For this you have to identify primary and secondary keys and implement B+-trees, B-trees or hash tables.

Exercise 3. Implement index structures on the relations above and integrate them into the query processing:

**(i)** Identify for each of the relations above a primary key and organise the data structure in Exercise 1 accordingly.

**(ii)** Implement index structures for your primary keys using B+-trees or hash tables. Use at least once a B+-tree.

**(iii)** Realise your operations from Exercise 1 using your index structure.

**(iv)** Identify useful secondary keys and implement at least two index structures for them, using at least once a B-tree and at least once a hash table.

**(v)** Integrate your index structures into the processing of the queries in Exercise 2.

EXERCISE 4. Finally, consider two queries that require an iteration until a fixed-point is reached. Define a DISTANCE relation among addresses, and define two persons to be *neighbours* if the distance between their addresses is below a threshold $\theta$. For every $n \in \mathbb{N}$ this gives rise to clusters of people within a distance of $n \cdot \theta$. Fix a treatment, e.g. a vaccination, and determine for each $n \in \mathbb{N}$ the percentage of clusters, where at least $p$ percent of the people in the cluster have received the treatment.

As clusters can be represented by a person, create a graph as the output of this query.

**total points: 110**