Important rules and remarks:

- You are allowed to use any source of information; **communication with third parties is forbidden**
- You must be able to explain your code and you must be able to perform modifications if asked; otherwise the exam is not passed
- If there is anything unclear, **ask** us
- You must use the technologies from the lab/lecture
- There are NO PARTIAL points, only the ones specified in the problem statement
- **The requirements MUST be solved in the order they appear in the problem statement; the points for any feature are given only if all previous features are completely correct**
- **After solving each feature, mark that feature as solved in the attendance file**
- **You will have to use a PRIVATE git repository and, after solving each feature, there must be a commit containing the feature name in the comment (F1, F2, etc); other commits are allowed**
- **after solving each GROUP of features notify the teachers for evaluation; you must present each GROUP of feature before proceeding to the next one; the GROUPS of features are the following:**
    - **F1, F2**
    - **F3, F4, F5, F6**
    - **F7, F8, F9**
    - **F10, F11**

Approval Voting

In an approval voting system, voters select all candidates they approve i.e. all candidates that are acceptable for them; *it is possible to select no candidate*.
After the voting stations are closed, in order to speed up the vote counting process, each voting station uses a **scanner,** a special device that scans each voting bulletin and sends some data to a **county**-level server. Each county level server processes the data from the corresponding scanners and sends updates to a nation-wide vote counting **monitor**. The Central Electoral Bureau is monitoring the received data (at the nation-wide vote counting **monitor**).
Write an application that simulates this process for a *fixed number of 3 candidates*. The project will be a modular one and will be composed of the following modules: scanner, county, monitor-core, monitor-web. There will also be a *webapp* (Angular) that may or may not be part of monitor-web.

The **scanner** is a console application that starts by reading from the console the scanner (i.e. voting station) *name* (assumed to be unique) and a *county-id* (assumed to be unique, *should* be greater than 1023 - no validation needed). In order to simulate a bulletin scanning process, the application randomly generates a tuple of the following form (a, b, c), where a, b, c may be either 0 or 1 (values of 1 show the selected/voted candidates). For example: (1,0,1) means that a voter has selected/voted CandidateA and CandidateC.

A delay is introduced after generating each tuple, e.g., Thread.sleep(new Random().nextInt(3000)).
The scanner keeps track of the number of processed bulletins and prints the following information on one line: <bulletin-count> - <scanner-name> <a> <b> <c>
Only the following data is sent to the corresponding county-level server: scanner-name, a, b, c.
The scanner app continuously sends bulletin data to the county-level server using tcp sockets.
There may be several scanner apps (distinct processes) corresponding to each county-level server.

The **county**-level server starts by reading from the console the county *name* (assumed to be unique) and the *county-id* (assumed to be unique, *should* be greater than 1023 - no validation needed). It concurrently receives data from several scanners and, as soon as new data arrives, it is printed in the console (scanner-name, a, b, c); afterwards, a delay is simulated, e.g., Thread.sleep(new Random().nextInt(4000)). **[F1 25p]**
At *fixed* intervals (e.g., Thread.sleep(5000);), each county-level server possibly sends the following information to the nation-wide monitor: county-name, a, b, c, nr; where a, b, c represent the total number of votes up to this point in time, respectively received in county *county-name* by CandidateA, CandidateB, and CandidateC. The *nr* value represents the current total number of votes from all stations/scanners in county *county-name.* The data is sent via a post request to the "/api/voting" rest endpoint using the Spring RestTemplate. The data is continuously sent at fixed intervals.
The data is only sent if there are changes with respect to the previous update. If there are no changes then the following message is printed in the console: "data not changed". There may be several county-level apps (distinct processes). **[F2 30p]**

The **monitor-web** module exposes a rest-api and when receiving data from a county-level server at the "/api/voting" endpoint (post request), it prints the received data in the console: county-name, a, b, c, nr. **[F3 30p]**
The data (county-name, a, b, c, nr) is saved in a database table called *countyvote* using services and SpringDataJpa repositories from a **monitor-core** module (same architecture from the lab). Besides the aforementioned information (county-name, a, b, c, nr), the *countyvote* table will only have a primary key called id (the entity identifier). **[F4 30p]**
The "/api/voting/all" endpoint gets all data from the *countyvote* table. **[F5 30p]**
At the address http://localhost:4200/voting/all (Angular) all information from the "/api/voting/all" endpoint is displayed, one item per line. **[F6 30p]**

The "/api/voting/latest" rest-api endpoint returns (get request) the latest voting data (*countyvote* table) from each county; so if there are two counties then a list with two elements is returned, each element representing the latest voting data from a county. **[F7 35p]**
At the address http://localhost:4200/voting/results, the data from the "/api/voting/latest" endpoint is displayed, one item per line. **[F8 35p]**

The list is updated at regular intervals as new data is sent by the scanners (page reload not allowed). **[F9 35p]**

Below the list, the three candidates (CandidateA, CandidateB, CandidateC) with their current number of votes are displayed, one candidate per line, descendingly sorted by the total number of votes from all counties. The format of each line is: <candidate-name>:<votes> **[F10 35p]**
The candidate ranking list is automatically updated (the votes are updated, and the list is sorted) at regular intervals as is sent by the scanners (page reload not allowed). **[F11 45p]**