Exercise sheet 3 2023-11-01

Due date: 2023-11-09 16:59

The goal of this exercise sheet is to make basic usage of the C++ standard library and its basic data containers.

The standard library is just code that makes use of C++ language features in order to provide you with code helper utilities. It is probably installed with your compiler, so you can find the standard library header files in `/usr/include/c++/` (on Debian/Ubuntu/Arch/... systems).

Exercise 1:

Implement a **contact list program**. We want to store names and phone numbers. We defined a `struct storage` to store two lists (`std::vector`): one for names, and one for numbers.

The names and numbers lists have to remain synchonized!

```
name of first person    - number of first person
name of second person   - number of second person
name of third person    - number of third person
...                     - ...
```

Yes, the default storage design is not ideal, but this "enterprise-grade" code now is what it is. If you want to improve it (index structures, better storage, ...), go ahead! But like any relevant software, the **API** needs to **remain compatible** (i.e. how all the functions behave, since that is what we test).

You can test your contact list with the `run.cpp`, executed with `runhw03`.

The following functionality is expected:

- Add a contact. Disallow empty or duplicate names and `return false`. `return true` for success.
- A function that returns how many contacts are currently stored
- Get the number for a given name - `return -1` if no such name is found.
- Add a function which returns the contacts list as string (so one can, for example, print it).
  - For each contact line, use this format: `name - number`. You may do pretty alignment padding with spaces!
  - The "oldest" contact should come at the top, the newest at the bottom. Except when the contact list was sorted of course.
- Remove a contact. Does nothing and `return false` if requested name was not part of the list. `return true` for success.
- Sort the contact list by name - watch out to keep the number list synchronized!
- Add a function to get the name that matches a number. `return ""` when not found.

Write your code in `contact_list.cpp` (it is already registered in `CMakeLists.txt`).