LABORATORY 5-7

1. KEEP CALM AND ADOPT A PET

The **"Keep calm and adopt a pet"** shelter needs a software application to help them find adoptive parents for the dogs[1] they are taking care of. The application can be used in two modes: administrator and user. When the application is started, it will offer the option to choose the mode.

Administrator mode: The application will have a *database*[2], which holds all the dogs in the shelter at a given moment. The shelter employees must be able to update the database, meaning: add a new dog, delete a dog (when the dog is adopted) and update the information of a dog. Each **Dog** has a breed, a name, an age and a photograph. The photograph is memorised as a link towards an online resource (the photograph on the presentation site of the centre). The administrators will also have the option to see all the dogs in the shelter.

User mode: A user can access the application and choose one or more dogs to adopt. The application will allow the user to:

a. See the dogs in the database, one by one. When the user chooses this option, the data of the first dog (breed, name, age) is displayed, along with its photograph.
b. Choose to adopt the dog, in which case the dog is added to the user's adoption list.
c. Choose not to adopt the dog and to continue to the next. In this case, the information corresponding to the next dog is shown and the user is again offered the possibility to adopt it. This can continue as long as the user wants, as when arriving to the end of the list, if the user chooses next, the application will again show the first dog.
d. See all the dogs of a given breed, having an age less than a given number. If the breed is empty, then all the dogs will be considered. The same options (a, b and c) apply in this case.
e. See the adoption list.

NB! PLEASE CONSULT THE OBSERVATIONS AT THE END OF THIS DOCUMENT.

[1] Feel free to use any other animal you prefer (cat, parrot, penguin, etc.).
[2] Please see the observations at the end of this document.

2. LMDB – Local Movie Database

So many movies, so little time… To make sure you do not miss any good movies, you absolutely need a software application to help you manage your films and create watch lists. The application can be used in two modes: administrator and user. When the application is started, it will offer the option to choose the mode.

Administrator mode: The application will have a *database*[3], which holds all the movies. You must be able to update the database, meaning: add a new movie, delete a movie and update the information of a movie. Each **Movie** has a title, a genre, a year of release, a number of likes and a trailer. The trailer is memorised as a link towards an online resource. The administrators will also have the option to see all the movies in the database.

User mode: A user can create a watch list with the movies that he wants to watch. The application will allow the user to:

a. See the movies in the database having a given genre (if the genre is empty, see all the movies), one by one. When the user chooses this option, the data of the first movie (title, genre, year of release, number of likes) is displayed and the trailer is played in the browser.
b. If the user likes the trailer, he/she can choose to add the movie to his/her watch list.
c. If the trailer is not satisfactory, the user can choose not to add the movie to the watch list and to continue to the next. In this case, the information corresponding to the next movie is shown and the user is again offered the possibility to add it to the watch list. This can continue as long as the user wants, as when arriving to the end of the list of movies with the given genre, if the user chooses next, the application will again show the first movie.
d. Delete a movie from the watch list, after the user watched the movie. When deleting a movie from the watch list, the user can also rate the movie (with a like), and in this case, the likes of the movie in the repository will be increased.
e. See the watch list.

NB! Please consult the observations at the end of this document.

[3] Please see the observations at the end of this document.

3. Proper Trench Coats[4]

Trench coats are cool. Everyone should own a trench coat. The **"Proper Trench Coats"** store sells fashionable, elegant trench coats and the store needs a software to allow their customers to "order online". The application can be used in two modes: administrator and user. When the application is started, it will offer the option to choose the mode.

Administrator mode: The application will have a *database*[5], which holds all the available trench coats at a given moment. The store employees must be able to update the database, meaning: add a new trench coat, delete a trench coat (when it is sold out) and update the information of a trench coat. Each **Trench Coat** has a size, a colour, a price, a quantity and a photograph. The photograph is memorised as a link towards an online resource (the photograph on the presentation site of the store). The administrators will also have the option to see all the trench coats in the store.

User mode: A user can access the application and choose one or more trench coats to buy. The application will allow the user to:

a. See the trench coats in the database, having a given size, one by one. If the size is empty, then all the trench coats will be considered. When the user chooses this option, the data of the first trench coat (size, colour, price, quantity) is displayed, along with its photograph.
b. Choose to add the trench to the shopping basket. In this case, the price is added to the total sum the user has to pay. The total sum will be shown after each purchase.
c. Choose not to add the trench coat to the basket and to continue to the next. In this case, the information corresponding to the next trench coat is shown and the user is again offered the possibility to buy it. This can continue as long as the user wants, as when arriving to the end of the list, if the user chooses next, the application will again show the first trench coat.
d. See the shopping basket and the total price of the items.

NB! PLEASE CONSULT THE OBSERVATIONS AT THE END OF THIS DOCUMENT.

[4] Feel free to use any other clothing item you prefer (dresses, ties, suits, shirts, boots etc.).
[5] Please see the observations at the end of this document.

4. <u>Master C++</u>[6]

You are very passionate about programing (otherwise you wouldn't be here, reading this) and C++ is a language so close to your heart. On your way to becoming a C++ guru, you study a lot and watch many tutorials. To make sure you do not miss any good tutorials, you absolutely need a software application to help you manage your tutorials and create watch lists. The application can be used in two modes: administrator and user. When the application is started, it will offer the option to choose the mode.

Administrator mode: The application will have a *database*[7], which holds all the tutorials. You must be able to update the database, meaning: add a new tutorial, delete a tutorial and update the information of a tutorial. Each **Tutorial** has <u>a title</u>, <u>a presenter</u> (name of the presenter person), <u>a duration</u> (minutes and seconds), <u>a number of likes</u> and <u>a link</u> towards the online resource containing the tutorial. The administrators will also have the option to see all the tutorials in the database.

User mode: A user can create a watch list with the tutorials that he wants to watch. The application will allow the user to:

a. See the tutorials in the database having a given presenter (if the presenter name is empty, see all the tutorials), one by one. When the user chooses this option, the data of the first tutorial (title, presenter, duration, number of likes) is displayed and the tutorial is played in the browser.
b. If the user likes the tutorial, he/she can choose to add it to his/her tutorial watch list.
c. If the tutorial seems uninteresting, the user can choose not to add it to the watch list and to continue to the next. In this case, the information corresponding to the next tutorial is shown and the user is again offered the possibility to add it to the watch list. This can continue as long as the user wants, as when arriving to the end of the list of tutorials with the given presenter, if the user chooses next, the application will again show the first tutorial.
d. Delete a tutorial from the watch list, after the user watched the tutorial. When deleting a tutorial from the watch list, the user can also rate the tutorial (with a like), and in this case, the likes of the tutorials in the repository will be increased.
e. See the watch list.

NB! PLEASE CONSULT THE OBSERVATIONS AT THE END OF THIS DOCUMENT.

[6] Feel free to use any other domain you prefer (cooking, essay writing, musical instrument playing etc.).
[7] Please see the observations at the end of this document.

OBSERVATIONS:

- Design the solution to this problem using the OOP paradigm.
- The application must be implemented in C++ and use layered architecture.
- The application must provide a console based user interface.
- Please provide tests for non-trivial functions.
- Please provide specifications for your functions.
- For problems 3 and 4, if you choose a different entity from the one that is specified in the problem requirement, the entity must have at least the same characteristics.
- The *database* of entities will be represented by a memory repository. Please add at least 5 entities in your memory repository (source code).
- Please add basic data validation.
- Please handle the following situations:
  - If an entity that already exists is added, a message will be shown and the entity **will not be stored**. You must decide what makes an entity unique.
  - If the user tries to delete an entity that does not exist, a message will be shown and there will be no effect on the list of entities.

- The problem should be solved in 2 iterations, the first one is due in **Week 5** and the second in **Week 7**:
  - Iteration 1 should solve at least the requirements related to the **administrator mode**. You should define a user defined type **DynamicVector**, which provides the (offering the specific operations: add, remove, length, etc). The array of elements in the DynamicVector must be dynamically allocated.
  - Iteration 2 should solve all the problem requirements. The **DynamicVector** should be modified to use templates.

Extra requirements (for bonus)

- Overload operators "+" and "-" for the DynamicVector:
  - "+" should offer the possibility to add an element to the vector in the ways presented below. Considering that v is a DynamicVector and e a generic element, we should be able to write both of the following:
    - v = v + e
    - v = e + v
  - "-" should offer the possibility to delete an element from the vector in the way presented below. Considering that v is a DynamicVector and e a generic element, we should be able to write the following:
    - v = v - e