# Laboratory 3-4

## Requirements

- Each student will be given one of the problems below.
- Write a **C application** to solve the requirement.
- The problem should be solved in 2 iterations, the first one is due in **Week 3** and the second in **Week 4**:
  - Iteration 1 should solve at least the first 2 requirements (a and b). In this first iteration the vector used in your repository can be *statically allocated*.
  - Iteration 2 should solve all the problem requirements. You must define a vector structure (including specific operations), which should use a *dynamically allocated array*. The solution in Iteration 2 should be *modular*.
  - The source code must be specified and contain tests for all functions except UI code.
  - *Make sure your programs do not have any memory leaks!*
- Use a layered architecture for your application (domain, repository, controller, UI).
- The application will provide a console based user interface.
- The user interface, domain and data access elements will be stored in different modules.
- The user interface module will only contain the user interface part (reads/writes).
- To be able to test your program, when the application starts, you must have at least 10 records.
- 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**.
  - 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.
- After delivering this entire assignment (both iterations), you must send an **.zip** archive containing your source code files (without any .obj or executable files) to the email address: commit@scs.ubbcluj.ro.

## Additional requirements - bonus possibility (0.1/0.3 p)

Implement any of the two requirements below for 0.1 bonus. If you implement both, you will receive 0.2 bonus points. To receive the bonus, the requirements must be implemented correctly, **by week 4/5** (as mentioned in each requirement) and the application must function properly.

1. Implement the following two extra requirements using function pointers (deadline: **week 4,** bonus**: 0.1p**):
   - For your problem, requirement b, add a different type of filtering (your choice).
   - For your problem, requirement c, add descending sorting. The user should choose the type of sorting and the program should show the list of entities accordingly.
2. For iteration 2, provide 2 different implementations for the undo/redo functionality: one using a list of operations and one using the "list of lists" approach. Implement your dynamic array generically, such that it can contain any type of elements (use *void**). Use this structure for your repository, as well as to implement the "list of lists" approach for the multiple undo/redo functionality (deadline: **week 5,** bonus**: 0.2p**).

## 1. Pharmacy

*Image source: http://www.apotekonline.biz/den-medicinska-varden/*

**John** is the administrator of the "**Smiles" Pharmacy**. He needs a software application to help him manage his pharmacy's medication stocks. Each **Medication** is stored in the following way: name, concentration, quantity and price. **John** wants his new application to help him in the following ways:

a. He wants to be able to add, delete or update a medication. A medicine is uniquely identified by its name and concentration. If a product that already exists is added, its quantity will be updated (the new quantity is added to the existing one).
b. He wants to be able to see all the available medications containing a given string (if the string is empty, all the available medications will be considered), sorted ascending by medication name.
c. He wants to be able to see only those medications that are in short supply (quantity less than **X** items, where the value of **X** is provided by **John**).
d. The application must provide multiple undo and redo functionality. Each step will undo/redo the previous operation performed by the user.

## 2. Bakery

**Mary** runs her family's bakery, "**Bread'n Bagel".** Every day she struggles with keeping up to date with available stocks of raw materials and would like a program to help her manage the business more effectively. Each **Material** used in the bakery must have: a name, a supplier, a quantity and the expiration date. **Mary** wants a software application that helps her in the following ways:

*Image source: http://www.clipartof.com/portfolio/colematt/illustration/bakery-shop-1151182.html*

a. The application must allow adding, deleting and updating a material. A raw material is uniquely identified by its name, supplier and expiration date. If a material that already exists is added, its quantity will be modified (the new quantity is added to the existing one).

b. Mary wants to be able to see all the available materials that are past their expiration date, containing a given string (if the string is empty, all materials past their expiration date will be considered).
c. The application must be able to display all materials from a given supplier, which are in short supply (quantity less than a given value), sorted ascending by their quantities.
d. The application must provide multiple undo and redo functionality. Each step will undo/redo the previous operation performed by the user.

## 3. Tourism Agency

Image source: http://www.news.com.au/travel/travel-updates/australias-cleanest-beach-named/story-e6frfq80-1111117516994

The employees of "**Happy Holidays"** need an application to manage all the offers that the agency has. Each **Offer** has a type (may be *seaside*, *mountain* or *city break*), a destination, a departure date and a price. The employees need the application to help them in the following ways:

a. The application must allow adding, deleting and updating a tourism offer. An offer is uniquely identified by its destination and departure date.
b. The application should offer the possibility to display all the tourism offers whose destinations contain a given string (if the string is empty, all destinations are considered) and they will be shown sorted ascending by their price.
c. The application should be able to display all offers of a given type, starting from a given date.
d. The application must provide multiple undo and redo functionality. Each step will undo/redo the previous operation performed by the user.

## 4. Real Estate Agency

**Evelyn** owns a real estate agency. Being also the single employee, she needs an application to help her manage all the real estates of her clients. Each **Offer** has a type (can be *house*, *apartment* or *penthouse*), an address, a surface and a price. **Evelyn** needs the application to help her in the following ways:

Image source: https://fishsouthbay.wordpress.com/category/cleaning-tips/

a. The application must allow adding, deleting or updating a real-estate. A real-estate is uniquely identified by its address.
b. The application should offer the possibility to display all offers whose address contains a given string (if the string is empty, all offers will be considered), sorted ascending by their price.

c. **Evelyn** would like to be able to see all offers of a given type, having the surface greater than a given value.
d. The application must provide multiple undo and redo functionality. Each step will undo/redo the previous operation performed by the user.

## 5. Intelligent Refrigerator

Image source: http://viralgadgets.net/wp-content/uploads/2016/01/Samsung-Family-Hub-Smart-Fridge-01-320x190.jpg

The company "**Home SmartApps**" has decided to design a new intelligent refrigerator. Besides the hardware, they need a software application to manage the refrigerator. Each **Product** has a name, a category (may be *dairy*, *sweets*, *meat* or *fruit*), a quantity and an expiration date.

a. The application must allow adding, deleting and updating a product. A product is uniquely identified by name and category. If a product that already exists is added, its quantity will be updated (the new quantity is added to the existing one).
b. The application should offer the possibility to display all the products whose names contain a given string (if the string is empty, all products from the refrigerator are considered) and they will be shown sorted ascending by their quantities.
c. The application should be able to display all products of a given category (if the category is empty, all types of food will be considered) whose expiration dates are close (expire in the following given **X** days).
d. The application must provide multiple undo and redo functionality. Each step will undo/redo the previous operation performed by the user.

## 6. World Population Monitoring

The **World Population Monitoring Organisation** needs an application to help keeping track of countries' populations. Each **Country** has a unique name, the continent it belongs to (may be *Europe*, *America*, *Africa*, *Australia* and *Asia*) and a population (stored in millions). The employees of the organisation need the application to help them in the following ways:

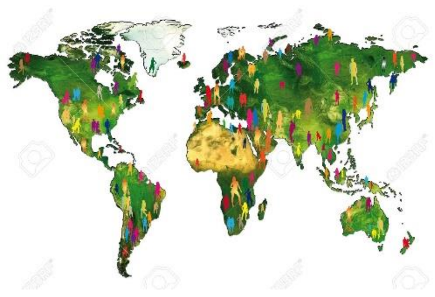
Image source: http://es.123rf.com/imagenes-de-archivo/poblacion_mundial.html

a. The application must allow adding, deleting or updating a country. Updating must also consider the case of migration: a given number of people leave one country to migrate in another.
b. The application should offer the possibility to display all the countries whose names contain a given string (if the string is empty, all the countries should be considered).

c. The application should allow displaying all the countries on a given continent (if the continent is empty, all states should be considered), whose populations are greater than a given value, sorted ascending by population.
d. The application must provide multiple undo and redo functionality. Each step will undo/redo the previous operation performed by the user.