Universidad de Alcalá

Laboratory Assignments

Data Structures

Fall 2021

Terms and conditions:

Assignment should be done in groups of 2 people.

Students have to present every assignment to the teacher and answer the teacher's questions about it on the corresponding lab session. You can find deadlines and days of presentation in the course presentation. These presentations should be supported with documents well suited for projector use.

Submissions that do not meet the deadlines and/or the requirements may not be reviewed. It is responsibility of the students to meet the requirements and it is not responsibility of the lecturer to warn students and go after them when such requirements are not met. Exceptions will not be considered (except for those included in the examination regulation-"*normativa de examenes*").

Students that do not pass the practical part of the course (i.e. the lab) on the January call may be required to complete a different assignment for the extraordinary call (also named Spring call or June call).

Important note: On these assignments only basic requirements are described. Students need to make additional assumptions and design decisions based on the given requirements. Students also need to decide on the input data and on the way in which that data is inputted into the system. Sample data needs to be provided to facilitate testing. Students are free to implement the solution in the way they prefer. Any decision will be reasonable providing that (1) it is explained on the written document and commented on the source code, and (2) it does not contradict blatantly the requirements stated here.

Content to upload to BlackBoard Platform:

- Assignment shall be delivered as a GitHub repository shared with the teacher in the appropriate GitHub assignment.
- Assignment shall include the source files/projects as well as a document (.doc or .PDF). The document must also include the IDs and names of all the members of the group.
- Assignment shall include the documentation used to support the explanations of the presentation day. PDF format is the only accepted format.

Documentation of every assignment

The documentation **must** contain at least the sections of the following index:

Index

1. Analysis
    1. Data flow from user or external files to system, internally in the system between every relevant component or storage, and from system to user or external files.
    2. ADT specifications:
        1. Explanation of the adaptations made to the standard specification of the ADT to fit requirements. If any ADT is used more than once, its description must be repeated for every case or stated to be the same than previous.
        2. Definition of operations of the ADT
            1. Name, arguments and return values

2. Design
    1. Diagram of the representation of the ADT in the memory of the computer (box diagrams) and explanation of every operation. Those explanations must be detailed only in case of operations that require it.
    2. Use case diagrams (UML notation)
    3. Class diagram
    4. Explanation of classes
        1. Explanation of significant methods. Including those created for operations and any other needed
    5. Explanation of the behavior of the program

3. Implementation
    1. Diagram with source files and their relations
    2. Explanation of every difficult section of the program

4. Review
    1. Running time of the solution (using big-Oh notation)
    2. Possible enhancements for the solution (in terms of efficiency and others)
    3. Reasoning on convenience or need of use of Dynamic DS in every case, comparing with Static DS characteristics.

5. References (Bibliography)

Common advice for all assignments

If you consider that any of these advice don't apply to your assignment, please answer teachers to ensure it.

1. Students will provide the necessary files to test the programs if the requested program are able to manage them (more than one file and with different sizes). The number of elements must be enough to test all operations.

2. File management must not be part of the data structures internal implementation.

3. The I/O operations, like printing and requesting data from user must not be part of any DS implementation.

4. The implementation of every assignment must not use high level libraries which implements data structures and their operations.

5. The implementation of assignments must use dynamic memory storage (pointers) for dynamic DS implementation.

6. Creating a running program and a proper documentation, will result in a mark enough to pass the course. The better is the solution and the documentation the higher will be the mark.

7. Failing to provide a working solution, an appropriate documentation or a clear presentation will result in low marks, not enough to pass the course.

8. Every program must produce enough output to show its internal behavior. Verbose programs are better, but the information must be organized to simplify the analysis, i.e. a list of 100 items, each one in a new line is worse than a list of items in 2-3 lines with some text explaining it.

Assignment 1 A: Linear data structures and operations

Students are asked to implement an O.O. C++ program to simulate a statistical data processor.

To do so, the program will execute two phases and then end.

1. The first phase will be in charge of data generation and will use interaction with the data engineer until certain conditions are met, moving to the next phase.
2. The second phase will be in charge of data exploitation, offering options to the user to manipulate data or to finish execution.

Data generation phase

The program will generate a random amount of data numbers based on user input until the amount of data numbers is > 100,000.

The cycle of data generation will follow these steps:

1. It will request two integer numbers to the user, A and B, both between 1 and 100.
2. It will calculate FB = (B * 500 * rand(0-1)) + 4000 * rand(0-1) – 6000* rand(0-1)
3. It will generate FB data numbers, where every number is calculated as: N_i =A +(1000*i) – (1000 * i * rand(0-1))
4. It will store every generated number in a queue, a stack and a list.
5. It will print A, B and FB and the accumulated amount of data numbers.
6. It will repeat if the accumulated amount of data numbers is < 100,000

Once the required number of elements is reached, the program will ask for a file name and will store all data in a text file with this name and ".txt" extension, with every number in a different line.

Data exploitation phase

In this phase, a menu is offered to the user to allow the use of the data stored in the three DS. After executing an option, the menu will be shown again until the user selects "exit".

The options of the menu will be:

1. Sorting algorithms
2. Search algorithms

3. Add data number
4. Remove data number
5. Exit.

The sorting option will execute 2 different sorting algorithms of DS. The students have to select the algorithms. The algorithm will create a separate DS of every type with the content of the originals DS sorted. For the creation of the sorted queue only the data of the original queue must be used and only queues can be used as temporal. The same apply for stacks and list, no DS mix is allowed. After sorting, the result of every DS will be stored in a file with the name provided in Phase 1 with the adding of every DS name.

The search option will search for the maximum value on every original (not sorted) DS, and also the 100 lower values of every DS. Found values of every search will be printed in the screen.

In both options, the time used for every operation must be showed, so they can be compared with the estimated running time.

The Add data number option will request a number to the user and introduce this number in every not sorted DS.

The Remove data number will request a number to the user and try to remove from all not sorted DS, printing the result of the operation (success/not success).

B: Extra (optional part).

Once the program is created and tested, it is requested to increase the number of data numbers to 1,000,000M (and multiplying FB by ten) to run the program and get bigger time measures, writing the results in the documentation and including this as a second project.

If time measures, and calculated running time make it possible, increase the data multiplying by 10 again and repeat the operations.

Assignment 2: Linear and non-linear data structures and operations

In this second assignment, we want.