Contents

Project

Build a small library.
Use it to solve a concrete problem.

Part 1: Build a small library

Define an ADT

Design and implement the DTs corresponding to the ADT over two data structures (DS)

- some restriction related to DS are given (that are given with the problem)

Part 2: Use the library

Given a problem, solve it by using the DT defined before.

Part 3: What did I learn from this project?

Remark:
Working programs are required in order to consider the project

Grading

1p automatically

Part 1

Build a small library
(5p)

1p **data abstraction**
ADT
Abstract class corresponding to ADT

4p **representation, operation design & implementation**
Remark: implementations should **respect ADT !**

2 x (0.25p.) **DS**
- choose appropriate DS (as appropriate as possible under the given restrictions)

If other ADTs are used: specify everything you need to use
(only what is needed)

2 x (0.5p) **operation design**
- complexity (for all operations)
- algorithm design (Pseudocode)

for nontrivial subalg. (at least the most complex 4 subalg.)

2 x (1p) **implementation**
source code – delivered in *electronic format only*
- use classes and choose appropriate OOP features

2 x (0.25p) **unit testing**
- source code (for unit testing) – delivered in electronic format only

Reason: DT implementation should be tested
independent to the problem to be solved

Part 2

Use the library and solve a problem!
(3p)

(0.5p) **problem: input/output & test data**

- input/output specification & example
  - from/to file when appropriate
- appropriate test data: consider black box testing
  - test sets (input/output sets)

(2p) **application design** **&** correct program

- **correctness and intuitive justification**
  present the idea / method used to solve the problem;
  - justify why the program should work correctly;
  - apply justification to some appropriate examples
  - if appropriate: present (maybe not standard) pseudocode for main
- other ADTs (specification & no pseudocode - only what is needed)
- source code structure:
  - names of the files
  - short indication of the content of each file

(0.5p) **execution time**

determine the (approximate) execution time for, at least, 3 sets of test data, **suitable selected**.

Part 3

What did I learn from this project?
(1p)

- Which DSs are best for given ADT?
- Which ADTs are best to be used to solve the given problem?
- Why did we studied DS? Why ADT are important?

For example:

Which DSs are best

You can consider:

- compare 2 DS

  For example, you can present/compare:
  - memory usage (*concrete, in your problem*)
  - complexity of operations
  - …
- Argue about advantages and disadvantages.

Remark: Please describe logic and correct arguments.
The accent is not on presenting all (possible) arguments.

General remarks

1. ADTs should contain all the specific operations (and will not contain the problem specific operations); they will be completely specified and implemented (independent of the problem)
2. Use iterators whenever they are appropriate !
3. Subalgorithms should be written independent from ADT representation by using only operations from ADT interface.

4. **Style** will be considered all along the project !

It refers to organization of the program and documentation.
For example:

- modularity, encapsulation
- input data from file (where appropriate) , …
- suitable a data types for elements
- suggestive selection of names, indentations, …
- comments
- etc.

5. You should use some sort of heading to specify each line of the specification refers to which of the above requirements. (Think of an appropriate way to emphasize this.) **There will be no points for specifications that have no clear indication to which requirement they refer to.**

6. The final exam subjects will not consist of such complex problems, nor with programming techniques.

Delivery aspects

Documentation + program

What kind of documentation?

Documentation can be written by hand or in electronic format (file).

Written documentation

A grouping containing the manual written documentation
and an electronic device containing source code of the project
+ instructions for compiling the project
Can be delivered without other discussions with the teacher

Electronic format documentation

documentation printed or in electronic format
Personal delivery:
It should be delivered by personal presentation
- Show how the program works (Run & test the program!)
- Answer to some questions related to the project
It is necessary to come with any *devices* needed to present the project.

Delivery schedule

There will be only one project delivery!

Delivery date:

last seminar of your group
If delivered later, with another group seminar class: **1 point off**.
If delivered during session then
- **3 points off**.
- only **personal delivery**
during (session) special office hours that will be announced

Cheating is not accepted !

Cheating leads to a grade of 0 (zero) for the cheated project.

Do your own project!
A project made by someone else means cheating.
Justifications like:

> ***I don’t know to answer to this question ;***
> ***someone helped me doing this project !***

are not accepted!