Programming Techniques project, Spring 2016

Costin Bădică and Alex Becheru

April 6, 2016

Professor:	Costin Bădică
Where:	Room S2
When:	Wednesday 12:00
Professor:	Alex Becheru
Where:	Room S6N
When:	Monday 10:00
Web:	Programming Techniques

Abstract

This document introduces the goals and methodology for developing the Programming Techniques project. The document contains also a description of the Programming Techniques project deliverables. The document targets 1st year students of the Computers in English speciality.

1 Introduction

The goal of your project is to develop a software for experimenting with algorithms. The project is focused on the development of skills for good programming of basic algorithms, covering coding, design, documentation and presentation of results. At the end of your project you must produce a set of deliverables including: a technical report, source code and experimental data.

This document is typeset using LaTeX [1].

2 General aspects

2.1 Project selection & allocation

Student's that are part of the same subgroup are not allowed to develop the same project. Before deciding on a specific project, check if it was not already chosen by someone else in your subgroup, see project allocation document. In order to select a project you have to complete the project allocation form.

The allocation is done by the professors based on the students' selection, see project allocation document. As a rule of thumb, in case of multiple selections of the same project by students of the same subgroup, the first to register the selection will be the the one to be allocated the project. The others will have to choose other projects.

2.2 Grading

The grading of your project will take into account all the elements presented in this document:

- The structure and content of the technical report should follow the requirements stated in section 4.
- The structure and content of the source code should follow the requirements stated in section 5.
- The content of the experimental data and results should comply with the requirements stated in section 6.

Additional points will be added for a correct Python implementation of your project.

3 Deliverables

For the project you have to produce three types of deliverables:

i) Technical report

ii) Source code

iii) Experimental data and results

3.1 Technical report

The technical report should describe briefly, concisely and clearly your work and achievements for the project. The description must contain your project task (problem), how did you develop the software and the outcome of your work. More details are given in section 4.

3.2 Source code

The C source code should contain the source modules of your software (*.c* and *.h* files). The software should be developed using ANSI C . The source files will not use compiler specific extensions. The source code should be accompanied by command lines for building the executables, as well as by *makefiles*. More details are given in section 5.

The Python source code should be developed in a modular way, such that multiple .py files are developed and used together. The software should be developed in either Python 2.7.x or Python 3.5.x. The source code should be accompanied by command lines for building the executables.

3.3 Experimental data and results

You should check your software on non-trivial input data. You should provide at least 5 non-trivial input data sets, and the corresponding outputs produced by your software. The source code for generating non-trivial input data will be provided as it is described in subsection 3.2. More details are given in section 6.

4 Technical report

The technical report must be typeset using LaTeXand provided in printed, as well as in electronic form (PDF and sources). For documentation about LaTeX you can consult reference [3].

The technical report must be divided into a number of sections, including:

i) Cover page

ii) Problem statement

iii) Pseudocode

iv) Application design

v) Conclusions

vi) References

4.1 Cover page

This is a one-page section containing the title of the project, the name of the student, the group, year and section where the student is enrolled.

4.2 Problem statement

This section is the introduction of your technical report. It should describe clearly the task of your project.

4.3 Pseudocode

This section should contain the pseudocode description of your algorithm. The pseudocode should use the format introduced in [2]. An example is shown in figure 1.

4.4 Application design

This section must contain a detailed description of the application design containing the following items:

- the high level architectural overview of the application

```
INSERTION-SORT($A$)
1. for $j \leftarrow 2, length[A]$ do
2.     $key \leftarrow A[j]$
3.     $\triangleright$ Insert $A[j]$ into the sorted sequence $A[1 \ldots j-1]$
4.     $i \leftarrow j - 1$
5.     while $i > 0$ and $A[i] > key$ do
6.         $A[i+1] \leftarrow A[i]$
7.         $i \leftarrow i - 1$
8.     $A[i+1] \leftarrow key$
```

Figure 1: Example for typesetting an algorithm.

- the specification of the input
- the specification of the output
- the list of all the modules in the application and their description
- the list of all the functions in the application, grouped by modules; for every function the following details must be provided:
  - the description of what the function does
  - a description of every parameter, and
  - the meaning of the return value

Your application design must also address the method for automated generation of input test data.

4.5 Conclusions

This section must contain your own conclusions after performing the project work. Some suggestions about what to include in the list of conclusions are: a summary of your achievements, which were the most challenging and interesting parts and why, future directions for extending the project in short and long term, a.o.

4.6 References

You should include here a list of bibliographic references that are cited in the text of the technical report. They can include: textbooks, journal articles, conference proceedings papers, and Web references.

5 Source code

The C source code must contain the complete source of your application, including *.c* files, header files (*.h*), and makefiles.

The Python source code must contain all the complete source of your application, *.py* files and any other libraries used in your project.

The following requirements must be met by the C source code. They will directly influence the grading of your project.

- The project code must successfully compile and build; when using GCC the build process should use the *make* tool and a makefile.
- The C code must follow the C99 standard (e.g. no compiler-specific extensions).
- The code must be portable; it must compile under two different compilers (e.g. Visual C++ and GCC)
- The code must use an indent style and address at least the following issues:
  - blocks must be indented using 4 spaces (no tabs)
  - every control construct (e.g. *if*, *else*, *while*, *for*. *switch*) must be braced – even those with single-line bodies:
  - the variable and function names should be descriptive, but not verbose
  - the use of global variables should be kept to a minimum
  - the use of preprocessor macros should be kept to a minimum
- The code should be commented (every function and important variables and code blocks inside functions). You must use the commenting style proposed by Doxygen for C-like programming languages. Your source code must be accompanied by code documentation in LaTeXautomatically generated using Doxygen, based on the comments that you added to your code.
- The code must be modular (minimum 2 *.c* files and minimum one *.h* file)
- You must use an automated method for generating non-trivial input test data

The following requirements must be met by the C source code. They will directly influence the grading of your project.

- The project code must be successfully interpreted and executed.
- The code must use the PEP 0008 indent style and address at least the following issues:

  - the variable and functions names should be descriptive, but not verbose
  - the use of global variable should be kept to a minimum

- The code should be commented (every function and important variables and code blocks inside functions). You must use the commenting style proposed by Doxygen for Python programming language. Your source code must be accompanied by code documentation in LaTeXautomatically generated using Doxygen, based on the comments that you added to your code.
- The code must be modular (minimum 3 *.py* files)
- You must use an automated method for generating non-trivial input test data.

6 Experiments and results

In this section you must explain the method that you used for testing your application, as well as the experimental results.

The experimental results will contain:

- A description of the output data that you obtained by running your algorithm, as well as the method that you used to test that this output is correct according to the algorithm specification.
- Optionally, the execution time of your algorithm, for each input data set.

You must also present the results (and optionally the recorded execution time) of your experiments in a meaningful way for the reader. The presentation method is left for your choice.

7 Schedule

Your project activity must be done by the end of the semester. Although there is no intermediary milestone, we highly advise you to show your work in progress to the teachers and/or request advices during the development period.

References

[1] Leslie Lamport, *LaTeX: A Document Preparation System*. Addison Wesley, Massachusetts, 2nd Edition, 1994.

[2] Thomas H. Cormen and Charles E. Leiserson and Ronald L. Rivest and Clifford Stein, *Introduction to Algorithms*. MIT Press, 3rd Edition, 2009.

[3] LaTeXproject site, `http://latex-project.org/`, accessed in April 2013.

A Technical report todo list

- ☐ Cover page
    - ☐ Title of the project
    - ☐ Student's name
    - ☐ Student's group
    - ☐ Speciality CE or CR
- ☐ Problem statement/Clear description of project's tasks.
- ☐ Pseudocde description of all the algorithms used in the project.
- ☐ Appplication Design
    - ☐ high level arhitectural overview of the application
    - ☐ specification of the input
    - ☐ specification of the output
    - ☐ the list of all the modules in the application and their description
    - ☐ the list of all the functions in the application, grouped by modules. For every function the following detailes must be provided:
        - ☐ the description of the function's tasks
        - ☐ a description of every parameter
        - ☐ the meaning of the return value
    - ☐ description of the method used to generate automatically test data
- ☐ Conclusions
    - ☐ summary of achievements
    - ☐ brief description of the most challenging/interesting achievements
    - ☐ future directions for extending the project
        - ☐ short term directions
        - ☐ long term directions
- ☐ References, each reference should include the following:
    - ☐ author's name/web-site name
    - ☐ name of the book, article, web-article
    - ☐ publisher's name and ISBN/ISSN/DOI
    - ☐ publishing year/ web-site visiting date

B Source code todo list

- ☐ C code:
    - ☐ the code respects C99 standard
    - ☐ the code respects an indent style, at least the following must be respected:
        - ☐ block indentation using 4 spaces
        - ☐ each control construct (e.g. *if, for*) is braced
        - ☐ descriptive but not verbose variable or function names
        - ☐ minimal use of global variables
        - ☐ minimal use of preprocessor macros
    - ☐ the code should is commented using Doxygen commenting style for C programmes
    - ☐ code documentation in LaTeXautomatically generated using Doxygen
    - ☐ code portability was demonstrated:
        - ☐ successful compilation under GCC compiler
        - ☐ successful compilation under Visual C compiler
    - ☐ the code must be modular, at least 2 *.c* files and 1 *.h* file
    - ☐ makefile creation
    - ☐ successfully build and compilation using the makefile
    - ☐ a program to create automated non-trivial input test data
- ☐ Python code (optional):
    - ☐ the code is modular (minimum 3 *.py* files)
    - ☐ PEP 0008 indent style is respected
        - ☐ block indentation using 4 spaces
        - ☐ descriptive but not verbose variable or function names
        - ☐ minimal use of global variables
    - ☐ the code was successfully interpreted and executed
    - ☐ the code should is commented using Doxygen commenting style for Python
    - ☐ code documentation in LaTeXautomatically generated using Doxygen

C Experiments and results todo list

- ☐ description of the output data
- ☐ description of the testing method
- ☐ execution time for each input data set (optional)
- ☐ comparison of execution times between Python and C implementations of the algorithms used in the project (optional)