Algorithms Design
Laboratory Assignment 2

Computers and Information Technology Department
Faculty of Automatics, Computers and Electronics

Abstract

This document introduces the goals and methodology for developing the Algorithms Design assignment. The document contains also a description of the lab homework deliverables. The document targets 1^{st} year students in their study of algorithms.

1 Introduction

The goal of your assignment is to develop a software for experimenting with algorithms. The lab homework 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 lab homework you must produce a set of deliverables including:

- Technical Report, .doc/.docx or typeset using LaTeX [1] (Overleaf tutorial).
- Source Code in C language (mandatory) and Python language (optional for bonus points) for both problems.
- Experimental non-trivial data[1].
- Source Code in C language (mandatory) and Python language (optional for bonus points) for data generation algorithms

2 General aspects

2.1 Assignment problems

Problem 1 (50p): N children play the well-known ala-bala-portocala game. They are placed on a circle and start counting from 1 to k. The k-th child is removed from the game. The process continues with the rest of the children, each time counting from 1 up to k, until all the children are removed from the

[1] large and very large data sets randomly generated

game. Implement an efficient algorithm for printing out the order in which the children are removed from the game using a circular list.

Problem 2 (50p): A fortress has multiple types of towers: attack and defense towers. Between any pair of towers, there may or may not be a tunnel. Suppose we have n towers and we have a list of r pairs of towers for which there are tunnels. Implement an O(n+r)-time algorithm that determines whether it is possible to assign some of the towers as attack and the remaining ones as defense towers such that each tunnel is between an attack and a defense tower. If it is possible to perform such a designation, your implemented algorithm should produce it.

2.2 Grading

The grading of your laboratory assignment 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 assignment.

3 Deliverables

For this assignment you have to produce three types of deliverables:

i) Technical report

ii) Source code for both problems

iii) Experimental data and results

iv) Source code for data generation algorithms

3.1 Technical report

The technical report should describe briefly, concisely and clearly your work and achievements for the lab homework. The description must contain your assignment tasks (problems), 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. 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 3.6.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 10 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.

3.4 Homework Delivery

To allow for an easier check of every delivered project/homework the following steps must be followed as well:

- The homework must be delivered in an archive of type and with extension .zip whose name will be as follows:
  - English section: AD-CEN1.1A-LastName-FirstName-Laboratory-Assignment-2.zip
  - Romanian section: PA-CR1.1A-Nume-Prenume-Laboratory-Assignment-2.zip
- The name of the above archive is made from the following elements:
  - Abreviation of the course name:
    - AD: Algorithm Design
    - PA: Proiectarea Algoritmilor
  - Section, year and group:
    - CEN: Computer English
    - CR: Calculatoare Romana
  - Last Name and First Name:
    - Your Last and First Name (And middle name as well where available) as it appears in your Faculty registration papers
  - Labs for which the homework is delivered or the name of the homework (current homework is Laboratory Assignment 2). The full and complete name of the homework must be used as assigned to you in Google Classroom).

Archive examples:

  - AD-CEN1.1A-Popescu-Ion-Laboratory-Assignment-2.zip
  - PA-CR1.1A-Popescu-Ion-Laboratory-Assignment-2.zip

- The zip archive must be uploaded in Google Classroom (without any external links or other storage where the homework might be available). Every homework is stored in Google Drive (when using Google Classroom):
  - After the homework is uploaded it is necessary to press the "Submit" button (otherwise the homework doesn't appear as delivered)
  - Not delivering the homework until the deadline leads to an automatic grade of 0
  - You can upload the homework and change it as many times as needed until the deadline (but only for urgent changes)
- Every archive must contain one directory/folder per problem (P1, P2, etc.). Every directory/folder will contain (please also see the other sections on how to deliver the assignment):
  - Source code (compilable and executable) for the respective problem, fulfilling the requirements mentioned above and below this chapter and containing also the full solution for the respective problem.
  - Test code (and data generation) must also be delivered in order to see how different scenarios where tested.
  - Comments and generated documentation (if case and if requested)
  - In the Technical report you must also mention all your observations regarding the algorithms you have chosen to implement the problem, other algorithms your thought about but proved to be inefficient, implemented solution, methods in which the code was tested, how you developed/thought answers to wrong/good input, limitations of your algorithm, etc.
  - Determining the time dependency variation chart on the problem dimension
  - There should not be any other main directory(ies) in the archive except the P1 and P2 directories and all files must reside in these 2 directories (P1 and P2) as per problem implementation
  - There must be exactly 2 technical reports. Each problem must have its own technical report. Thus, the technical report for the 1^{st} problem must be in P1 directory while the technical report for the 2^{nd} problem must be in P2 directory
  - In case technical reports are implemented using LaTeXthen for each technical report the LaTeX source must reside in the respective problem folder (P1 and respectively P2)

- To implement your homework:
  - Read the requirements multiple times
  - Deliver all requirements (both problems in this case, the technical report, all observations)
  - Don't postpone the delivery until the final moment
- Note:
  - Your archive names must be exact as mentioned above. Any other type/format of upload will not be taken into consideration
  - Only one archive must be uploaded in Google Classroom per assignment. Multiple files will not be taken into consideration
  - Not delivering the assignment will lead to an automatic generated grade of 0
  - Any homework/assignment must be individually made. Do not copy from other colleagues, sources from the Internet, etc. All deliveries must be personally developed, implemented, created and tested.

4 Technical report

The technical report must be typeset using LaTeXor .doc/.docx and provided in printed, as well as in electronic form (PDF and sources in case of LaTeX). 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) Algorithms

iv) Experimental data

v) Results & Conclusions

4.1 Cover page

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

4.2 Problems statement

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

4.3 Algorithms

This section should contain the pseudo-code description of the algorithms employed throughout your assignment. The pseudo-code should use the format introduced in [2]. An example is shown in figure 1.

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

Figure 1: Example for typesetting an algorithm.

Moreover, this section must discuss each algorithms' memory and computational complexity, in the best and worst case scenarios. Any written explanations, exemplifications or figures regarding the algorithms should appear here.

4.4 Experimental data

Your application must also address the method for automated generation of non-trivial input test data. Non-trivial data can be obtained by randomly generating large and very large data-sets, e.g., vectors of length $10^3 - 10^8$. For some details on random generators in C language, see *Random Generators*, *C Randomization* or *String Random Generators* . Any algorithm used for the generation of the data-sets should be presented here.

When describing the algorithm used for the generation of the data-sets, you must also describe the logic behind the algorithm generating the data and how you ensured that the data is meaningful for the tests.

4.5 Results & Conclusions

Figures comparing the average times (cpu time) on multiple runs on each data-set should be present here. The figures can be created with Microsoft Excel, Matlab, R, etc. The results should be commented and explained.

This section must contain your own conclusions after performing the lab homework. 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 lab homework in short and long term, a.o.

5 Source code

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

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

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

- The lab homework code must successfully compile and build; when using GCC the build process should use the *make* tool and a Makefile. It should also be easy to open/import and build/run the delivered Code::Blocks project.
- 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 follow the C coding style, imposed for this course.
- The code should be commented (every function and important variables and code blocks inside functions).
- 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 Python source code. They will directly influence the grading of your lab homework.

- The lab homework 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).
- The code must be modular (minimum 2 *.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.
- Tests performed with other implementations you have thought for (and implemented) for the same problem (time comparison, memory comparison, etc.)
- 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.

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 to-do list

- ☐ Cover page
  - ☐ Title of the assignment (Assignment 1)
  - ☐ Student's name
  - ☐ Student's group
  - ☐ Speciality CE or CR
- ☐ Problem statement/Clear description of assignment's tasks.
- ☐ Pseudocode description of all the algorithms used in the lab homework.
- ☐ Conclusions
  - ☐ summary of achievements
  - ☐ brief description of the most challenging/interesting achievements
  - ☐ future directions for extending the lab homework
    - ☐ short term directions
    - ☐ long term directions

B Source code to-do 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
  - ☐ the code should be commented
  - ☐ the code must be modular, at least 2 *.c* files and 1 *.h* file for each Code::Blocks project
  - ☐ successfully build and compile
  - ☐ 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 be commented

C Experiments and results to-do 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 lab homework (optional)