COURSE OUTLINE

(1) GENERAL

SCHOOLS	**ENGINEERING, NATURAL SCIENCES**		
ACADEMIC UNIT/UNITS	**DEPARTMENT OF COMPUTER ENGINEERING AND INFORMATICS, DEPARTMENT OF MATHEMATICS**		
TITLE OF MASTER'S DEGREE	***MSC in Data Driven Computing and Decision Making***		
LEVEL OF STUDIES	Postgraduate		
COURSE CODE	**DDCD 108**	SEMESTER	Spring
COURSE TITLE	**ADVANCED ALGORITHM ENGINEERING**		

INDEPENDENT TEACHING ACTIVITIES *if credits are awarded for separate components of the course, e.g. lectures, laboratory exercises, etc. If the credits are awarded for the whole of the course, give the weekly teaching hours and the total credits*	WEEKLY TEACHING HOURS	CREDITS
Lectures, Tutorials, Laboratory	2(L), 2(T),4(Lab)	7.5
Add rows if necessary. The organisation of teaching and the teaching methods used are described in detail at (d).		

COURSE TYPE *general background, special background, specialised general knowledge, skills development*	Specialized general knowledge, skills development
PREREQUISITE COURSES:	Recommended prerequisite knowledge (undergraduate level): Object-Oriented Programming, Programming in C/C++, Algorithms, Data Structures, Combinatorial Optimization, or equivalent.
LANGUAGE OF INSTRUCTION and EXAMINATIONS:	Greek (English if there are foreign students)
IS THE COURSE OFFERED TO ERASMUS STUDENTS	Yes
COURSE WEBSITE (URL)	https://www.ceid.upatras.gr/webpages/faculty/zaro/teaching/adv-alg-eng/index.html

(2) LEARNING OUTCOMES

Learning outcomes
The course learning outcomes, specific knowledge, skills and competences of an appropriate level, which the students will acquire with the successful completion of the course are described. *Consult Appendix A* • *Description of the level of learning outcomes for each qualifications cycle, according to the Qualifications Framework of the European Higher Education Area* • *Descriptors for Levels 6, 7 & 8 of the European Qualifications Framework for Lifelong Learning and Appendix B* • *Guidelines for writing Learning Outcomes*

Upon conclusion of the course the students ought to be able to:

- Understand techniques, properties, implementations and applications of fundamental and advanced algorithms and data structures.
- Apply techniques for the efficient implementation of fundamental and advanced algorithms and data structures.
- Apply the scientific experimental methodology in empirically and comparatively assessing algorithms and data structures.
- Use algorithmic software platforms and libraries for developing new efficient implementations.
- Develop implementations of complex algorithms and data structures with practical usability and applicability.
- Understand the process of converting user requirements to efficient and useful algorithmic software.

Upon conclusion of the course the students are expected to have the following skills/competences:

- Considerably improve their skills in the efficient and effective implementation of algorithms and data structures.
- Use advanced algorithm engineering techniques.
- Understand and apply properly the scientific experimental methodology in empirically and comparatively assessing algorithms and data structures.
- Develop efficient and effective implementations of complex algorithms and data structures.
- Apply effectively the process of converting user requirements to efficient and useful algorithmic software.

General Competences

Taking into consideration the general competences that the degree-holder must acquire (as these appear in the Diploma Supplement and appear below), at which of the following does the course aim?

Search for, analysis and synthesis of data and information, with the use of the necessary technology	*Project planning and management*
Adapting to new situations	*Respect for difference and multiculturalism*
Decision-making	*Respect for the natural environment*
Working independently	*Showing social, professional and ethical responsibility and sensitivity to gender issues*
Team work	*Criticism and self-criticism*
Working in an international environment	*Production of free, creative and inductive thinking*
Working in an interdisciplinary environment	*……*
Production of new research ideas	*Others…*
	…….

- Search for, analysis and synthesis of data and information, with the use of the necessary technology
- Adapting to new situations
- Decision-making
- Working independently
- Production of new research ideas
- Criticism and self-criticism
- Production of free, creative and inductive thinking

(3) SYLLABUS

- Motivation, peculiarities and importance of algorithm engineering. The algorithm engineering life cycle.
- Development of software platforms and libraries that ease the efficient implementation and experimental evaluation of algorithms and data structures.
- Methodological issues regarding the empirical research of algorithms and data structures. Application of the scientific experimental method. Reproduction of experiments. Creating and searching for proper data sets.
- Program correctness checking. Certifying algorithms and programs.
- Efficient implementation and experimental evaluation of classic and advanced algorithms and data structures. Case studies: graph algorithms, shortest path algorithms, maximum flow algorithms.
- Heuristic methods for enhancing implementations and boosting their performance.
- Arithmetic precision errors and methods for their handling.
- Methodological issues in the process of converting user requirements to efficient and useful algorithmic software.
- Implementation environment: C++ along with the algorithmic software platforms/libraries LEDA, STL and BOOST.
- Special topics: algorithm animation, generative programming, template metaprogramming, policy-based class design.

(4) TEACHING and LEARNING METHODS - EVALUATION

DELIVERY *Face-to-face, Distance learning, etc.*	Face-to-face. Tutorials and laboratory sessions with exemplary solutions of exercises. Introductory tutorials in C++. Hands-on experience in programming laboratory.
USE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY *Use of ICT in teaching, laboratory education, communication with students*	ICT methods are used in both teaching and communication with the students. Lecture slides and supplementary material are uploaded in the course's web site.
TEACHING METHODS *The manner and methods of teaching are described in detail.* *Lectures, seminars, laboratory practice, fieldwork, study and analysis of bibliography, tutorials, placements, clinical practice, art workshop, interactive teaching, educational visits, project, essay writing, artistic creativity, etc.* *The student's study hours for each learning activity are given as well as the hours of non-directed study according to the principles of the ECTS*	see table below

Activity	*Semester workload*
Lectures	2*13=26
Tutorials (exercises)	2*13=26
Laboratory practice	4*13=52
Individual study, preparation and problem solving	3*13=39
Weekend study	2*13=26
Study during the 3 "empty weeks" (2 weeks of vacation and 1 week of exam preparation)	6*3=18
Course total (25 hours per ECTS unit)	**187**

STUDENT PERFORMANCE EVALUATION *Description of the evaluation procedure* *Language of evaluation, methods of evaluation, summative or conclusive, multiple choice questionnaires, short-answer questions, open-ended questions, problem solving, written work, essay/report, oral examination, public presentation, laboratory work, clinical examination of patient, art interpretation, other* *Specifically-defined evaluation criteria are given, and if and where they are accessible to students.*	The language of instruction and examination is Greek. Special provisions (lecture notes and examinations in English) can be made for foreign students. Evaluation (criteria can be found in the web site of the course): • Programming exercises (30% of final mark), aiming at familiarizing students with: ▪ The efficient and effective implementation of non-trivial algorithms and data structures in C++. ▪ The use of the algorithmic software platforms and libraries LEDA and Boost. ▪ The use of the algorithm engineering techniques taught in the course. ▪ The proper experimental evaluation of implemented algorithms and data structures. ▪ The proper interpretation of experimental results and the errors they may contain. • Examination on the presentation of a state-of-the-art topic (20% of final mark), after thorough study and analysis of relevant bibliography, aiming at familiarizing students with state-of-the-art technology. • Final examination (50% of final mark): oral and written examination on a programming project assigned individually to each student. Examination on the code of the implementation as well as on the written report containing the details of the implementation and the results of the experimental evaluation.

(5) ATTACHED BIBLIOGRAPHY

- Suggested bibliography:

- K. Mehlhorn and S. Naeher, *LEDA: A platform for combinatorial and geometric computing*, Cambridge University Press, 1999.
- M. Mueller-Hannemann and S. Schirra, *Algorithm Engineering - Bridging the Gap between Algorithm Theory and Practice*, Springer 2010.
- C.C. McGeoch, *A Guide to Experimental Algorithmics*, Cambridge University Press, 2012.
- A.Alexandrescu, *Modern C++ design: Programming and Design Patterns Applied*, Addison-Wesley, 2001.
- M.A. Weiss, *Data structures and problem solving with C++*, 2nd Edition, Addison-Wesley, 2000.
- A. Koenig and B.Moo, *Accelerated C++*, Addison-Wesley, 2000.
- S.B. Lippman and J. Lajoie, *C++ Primer*, 3rd Edition, Addison-Wesley, 1998.
- N.Jossutis, *The C++ standard library: a tutorial and a reference*, Addison Wesley, 1999.
- J.Siek, A.Lumsdaine, and L.Lee, *The Boost Graph Library: User Guide and Reference Manual*, Addison Wesley, 2002.
- Gamma, R. Helms, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.
- Lecture notes and slides uploaded in the web site of the course.

- Related academic journals:

- ACM Journal of Experimental Algorithmics
- Algorithmica, Springer
- Journal of Discrete Algorithms, Elsevier

-Related academic conferences:

- European Symposium on Algorithms (ESA)
- Algorithm Engineering and Experiments (ALENEX) conference
- Int’l Symposium on Experimental Algorithms