MODULE OUTLINE

1. GENERAL INFORMATION

SCHOOL	SCHOOL OF SCIENCE AND TECHNOLOGY			
PROGRAM COURSE	INFORMATICS			
LEVEL OF STUDY	UNDERGRADUATE			
MODULE CODE	PLI-30	**YEAR OF STUDY**	3rd	
MODULE TITLE	Foundations of Computer Science			
INDEPENDENT TEACHING ACTIVITIES *in case credits are awarded for separate components/parts of the course, e.g. in lectures, laboratory exercises, etc. If credits are awarded for the entire course, give the weekly teaching hours and the total credits*			**HOURS**	**CREDIS**
Weekly teaching hours x 32 weeks			14-16	18 ECTS
COURSE TYPE *Background knowledge, Scientific expertise, General Knowledge, Skills Development*	Scientific expertise Compulsory			
PREREQUISITE MODULES:	Students can choose PLI12 and PLI20, or PLI20 and PLI30 together, if they have chosen PLI12 or PLI20 respectively during the previous academic year and repeat it with the obligation of only final exams.			
LANGUAGE OF INSTRUCTION AND EXAMS	GREEK			
THE MODULE IS OFFERED TO ERASMUS STUDENTS	No (due to the annual duration of the module)			
MODULE WEBSITE (URL)	https://www.eap.gr/education/undergraduate/computer-science/topics/#themeliwseis_epistim_yp Each module has its own space in the Learning Management System of EAP (http://study.eap.gr), with controlled access (use of code) for students and teaching staff.			

2. LEARNING OUTCOMES

Learning Outcomes
● *The course learning outcomes, specific knowledge, skills and competences of an appropriate (certain) level, which students will acquire upon successful completion of the course, are described in detail. It is necessary to consult:*
The PLI30 module consists of three distinct parts: 1) Algorithms and Complexity, 2) Theory of Computation, and 3) Automata and Formal Languages. The learning outcomes are distinguished into A) Knowledge and Understanding, B) Intellectual (Thinking) Skills, C) Analysis & Synthesis Skills.

A) Knowledge and Understanding.

On successful completion of the module, the students will be able to :

1. Describe simple algorithms in pseudocode, explain its functionality, use asymptotic notation, determine the worst-case running time, formulate and solve recursive equations, describe the generic algorithmic methods of Divide-and-Conquer, Greediness and Dynamic Programming, as well as the graph searching techniques of Depth-First Search (DFS) and Breadth-First Search (BFS).
2. Describe and define formally a Turing machine and its related concepts (Computation, Function, Grammar without limitations, M-recursive function), record the sequential steps of computation, define intuitively and formally the concept of an algorithm, define the halting problem, describe the universal machine Turing, refer some well-known intractable problems, describe the dove-tailing technique, define the DTIME and NTIME complexity classes as well as the classes P, NP, and EXP, describe the class NP via a polynomial verifier and a short certificate, describe the meaning of complete, tractable and hard problems, describe the role and use of reductions, define the classes of polynomial and exponential space complexity PSPACE and EXPSPACE, describe the space and time constructed functions, define and prove theorems of space and time hierarchy, define an approximation algorithm, describe the probabilistic Turing machine, and distinguish between the features of Monte Carlo and Las Vegas algorithms.
3. Describe the concept of a language, its fundamental operations and what is a regular expression, state the characteristics of a finite state machine, describe the process of accepting an input string, define a finite automaton and explain what is the transition function, describe the language acceptable by a deterministic and a non-deterministic finite automaton, state the pumping lemma for regular languages and its use, define a context-free grammar and its variables, terminal symbols and derivations, state the basic features of a push down automaton (PA), describe the recognition process of an input string and explain its transition function, state the two types of string recognition and the corresponding languages that are acceptable by PAs, define a deterministic push down automaton, describe how a grammar can be transformed into an equivalent one that does not contain unitary rules ass well as state and explain how the pumping lemma is used for context-free grammars.

B) Intellectual (Thinking) Skills.

On successful completion of the module, the students will be able to:

1. Use the asymptotic analysis in determining the complexity of iterative and recursive algorithms, determine exact asymptotic estimations for solving recursive equations, apply the “divide and conquer” method for solving problems of intermediate difficulty, use the method of dynamic programming and the greedy method to solve computational problems of mix difficulty, store a graph using the adjacency list and the adjacency matrix representation, apply the techniques of Depth-First Search (DFS) and Breadth-First Search (BFS for exploring a graph.
2. Prove that the halting problem is undecidable using the diagonalization method, prove the important properties of Turing decidable and Turing recognizable languages, prove that the satisfiability (SAT) problem is NP-complete, prove that a problem is Turing decidable or undecidable, decide whether a given problem belongs to the class P, or NP, or NP-complete, prove the existence of PSPACE-complete problems through polynomial time reductions (problem QSAT), classify problems to the classes of polylogarithmic complexity.

3. Prove the closure properties of regular languages under the operations of union, intersection, concatenation and Kleene star, prove that two expressions belong to the same language, explain why every finite language is regular, prove whether a given language is regular or not, transform a non-deterministic finite automaton to a finite one, prove whether a given language is context-free, design pushdown automata, apply the pumping Lemma for context-free languages.

C) Analysis & Synthesis Skills.
On successful completion of the module, the students will be able to:
1. Select the most suitable algorithm for a given set of input instances, compare the exponent of two functions using asymptotic notation, prove that a greedy algorithm computes the optimal solution, use effectively the methods of Divide-and-Conquer, Greediness and Dynamic Programming to design new algorithms, reason about complicated algorithms and compute their complexity, synthesize or modify existing algorithms for problem solving.
2. Design simple Turing machines that execute requested computations or recognize or decide a given language, distinguish between decidable and undecidable problems, relate the time complexity between variants of the Turing machine, synthesize effectively basic Turing machines in order to create more complex ones, identify the usefulness of the dove-tailing technique, reduce a given problem of known complexity to another and thus determining the complexity of the second problem, relate the two space complexity classes and the way they are correlated via the Savitch theorem.
3. Identify the regular expression that corresponds to a language, justify why regular languages are closed under the operations of intersection, union, concatenation and Kleene star, construct a finite automaton for a given regular expression, construct the regular expression corresponding to the language of a given finite automaton, transform a non-deterministic finite automaton to a deterministic one that accepts the same language, explain when a certain context-free grammar is ambiguous, transform an automaton to a regular language and vice versa, justify the recognition of a context-free language by a pushdown automaton, develop algorithms for solving decision problems for regular and context free languages.

General Competences

Taking into consideration the general competences that students/graduates must acquire (as those are described in the Diploma Supplement and are mentioned below), at which of the following does the course attendance aim?

Search for, analysis and synthesis of data and information by the use of appropriate technologies,
Adapting to new situations
Decision-making
Individual/Independent work
Group/Team work
Working in an international environment
Working in an interdisciplinary environment
Introduction of innovative research
Project planning and management
Respect for diversity and multiculturalism
Environmental awareness
Social, professional and ethical responsibility and sensitivity to gender issues
Critical thinking
Development of free, creative and inductive thinking
……
(Other…….citizenship, spiritual freedom, social awareness, altruism etc.) …….

- Search for, analysis and synthesis of data and information by the use of appropriate technologies
- Adapting to new situations
- Decision-making
- Individual/Independent work

- Critical thinking
- Social, professional and ethical responsibility and sensitivity to gender issues
- Development of free, creative and inductive thinking

3. MODULE CONTENT

The main objective of the module is to introduce students to the foundational principles of computation in Computer Science and to computational thinking. The goal is to learn fundamental algorithmic techniques for problem solving as well as the limits of computation. In particular, the module studies fundamental issues of computability that include the solvability or unsolvability of problems by computers, the estimation of computational resources that are required to solve a particular problem, the concept of algorithm, as well as the process of designing, analyzing, and implementing algorithms to the point that they can be useful in practice.

The key subjects of the module are:
1. Algorithms and Complexity
2. Theory of Computation
3. Automata and Formal Languages

4. TEACHING METHODS--ASSESSMENT

MODES OF DELIVERY *Face-to-face, in-class lecturing, distance teaching and distance learning etc.*	Distance education with five Group Counseling Meetings (OSS) during the academic year on weekends.
USE OF INFORMATION AND COMMUNICATION TECHNOLOGY *Use of ICT in teaching, Laboratory Education, Communication with students*	We use : - Remote meetings tools - Presentation software (e.g., power point), - Specialized software in the subjects under study (simulators, etc.). Additionally, the students use office automation tools, web browsers and e-reader for digital books.
MODULE DESIGN *Description of teaching techniques, practices and methods: Lectures, seminars, laboratory practice, fieldwork, study and analysis of bibliography, tutorials, Internship, Art Workshop, Interactive teaching, Educational visits, projects, Essay writing, Artistic creativity, etc*	

Activity	***Annual Workload***
5 OSS (x 4 hours)	20
Solving Exercise Assignments (5 assignments x 13 hours)	65
Examination	7
Individual study	356-420

The study hours for each learning activity as well as the hours of selfdirected study are given following the principles of the ECTS.	**Total module workload (hours)**	**448-512**
STUDENT PERFORMANCE EVALUATION/ASSESSMENT METHODS *Detailed description of the evaluation procedures.* *Language of evaluation, assessment methods, formative or summative (conclusive), multiple choice tests, short- answer questions, open-ended questions, problem solving, written work, essay/report, oral exam, presentation, laboratory work, other……etc.* *Specifically defined evaluation criteria are stated, as well as if and where they are accessible by the students*	Elaboration of written exercise assignments during the academic year; the average grade of exercises participates in the formation of the final grade of the module by 30%, if there is a passable grade in the final or resit examinations. The grade of the final written exams participates in the formation of the final grade of module by 70%. All the criteria are posted, both in each written assignment (in the LMS study.eap.gr), as well as in the general regulation of HOU at: https://www.eap.gr/education/study-regulations/	

5. SUGGESTED BIBLIOGRAPHY

- *Suggested bibliography:*

- Book A: Algorithms and Complexity, HOU, Patras, 2001.
- Book B: Theory of Computation, HOU, Patras, 2003.
- Book C: Automata and Formal Languages, HOU, Patras, 2002.
- M. Sipser (2019), Introduction to the Theory of Computation, Greek translation, Crete University Press, 2nd edition.
- J. Kleinberg and E. Tardos (2008), Algorithm Design, Greek translation, Klidarithmos, 1st edition (Chap. 4: pp. 147-186, 197-214, 220-229, 230-243; Chap. 5: pp. 247-273, 282-285; Chap 6: pp. 291-325, 332-340, 351-374).
- Supplementary Material (66 pages): Guide of Study for the book «M. Sipser (2019), Introduction to the Theory of Computation, Crete University Press, 2nd edition».
- Supplementary Material (66 pages): Guide of Study for the book «Kleinberg and E. Tardos (2008), Algorithm Design, Klidarithmos, 1st edition».
- T. Cormen, C. Leiserson, R. Rivest, and C. Stein (2016). Introduction to Algorithms, Greek translation, Crete University Press, 3rd edition.
- K. Mehlhorn and P. Sanders (2014). Algorithms and Data Structures – The Basic Toolbox. Greek translation, Klidarithmos, 1st edition.

Additional complementary digital material (and multimedia) is located within the LMS study.eap.gr education platform

-Related scientific Journals:

1) ACM Transactions on Algorithms
2) Algorithmica, Springer
3) Algorithms, MDPI
4) Information and Computation
5) Journal of Discrete Algorithms, Elsevier
6) Journal of the ACM
7) SIAM Journal on Computing
8) Theoretical Computer Science, Elsevier