|  |
| --- |
| **ΜΑΘΗΜΑ «Λογικός Προγραμματισμός» CSE803** |
| **code**  | **CSE803** |
| **title** | Logic Programming |
| **type (compulsory/optional)** | Optional (Elective) |
| **cycle (first/second/third)** | first |
| **year of study when the component is delivered (if applicable)** |  |
|  **semester/trimester when the component is delivered** | spring |
| **number of ECTS credits allocated** | 5 |
| **name of lecturer(s), with information about how, when and where to contact them.** | Ilias Sakellariou |
|  **learning outcomes** | Logic Programming and Constraint Logic programming are among the most interesting programming schools, that significantly differ from the "classical" schools of imperative and object oriented programming. Upon successful completion of the course the student will be able to: (1) understand the different approach of declarative programming in algorithm implementation, (2) understand the advantages and disadvantages of Logic Programming compared to imperative programming, (3) identify types of applications or modules of larger software systems that could be developed in significantly reduced time using declarative programming, (4) describe and be able to exploit procedures such as term unification and use higher order predicates, (5) design and implement logic programs, exploiting the execution mechanism of the language, unification, and techniques such as recursion and procedural abstraction, (6) explain the nature of a constraint variable, its domain and constraints as relations that express partial information for the problem, (7) describe and explain constraint solving techniques, (8) model problems as constraint satisfaction problems and develop the corresponding implementations in a CLP system. |
| **mode of delivery (face-to-face/distance learning etc.)** | face to face |
|  **prerequisites and co-requisites (if applicable)** | The student should be familiar with basic notions of programming such as variables, execution, etc.  |
| **course content** | Introduction to Logic Programming. Declarative Programming, First Order Predicate Logic and Logic Programs. Prolog Syntax, facts, rules. Program Execution-queries. Logic Variables and Scope. Terms and Unification. Resolution. Execution Mechanism. Debugging. Recursion. Prolog Arithmetic. Lists, cut and control of execution. Higher Order predicates (all solutions, variable call, negation as failure, term composition and decomposition, Prolog DB). Files. Graphs. Constraint Satisfaction Problems. The notion of constraints over Variables. Domains. Solving Constraint Satisfaction problems. Filtering algorithms on binary and higher order constraints. The Eclipse programming language constraint System. Example problem classes (scheduling, resource allocation) and the related global constraints. |
|  **recommended or required reading and other learning resources/tools** | English TextBooks • Apt, Krzystof R. ;Wallace, Mark G. "Constraint Logic Programming Using ECLiPSe", Cambridge University Press, 2007.  • Bratko, Ivan. Prolog Programming for Artificial Intelligence, (4th edition), Addison Wesley, 2012.  • Kowalski, Robert. Logic For Problem Solving, North-Holland, 1983 (from author's web page) (additional material will be available from the course web page).In case the student is fluent in Greek:41958366 Prolog: Programming in Logic for Artificial Intelligence, Type: textbook (In Greek), Manolis Marakakis, 2014, New Technology Publications, ISBN: 978-960-6759-98-7  |
|  **planned learning activities and teaching methods** | During the semester, there will be 26 hours of lectures and 13 hours of Laboratory work. Each week will include a 2 hour lecture, followed by a 1 hour laboratory class on the delivered topic.  |
| **assessment methods and criteria** | Final written Examination (70%), weekly Coursework (10%), Practicals (20%)Weekly coursework will involve programming exercises in Prolog and practicals the development of Prolog programs on more challenging problems. Final written examination involves a set of exam question aiming to evaluate student understanding on basic notions, such as execution and unification, and the ability to use recursion, unification, backtracking and constraints to solve small problems.  |
| **language of instruction** | English  |