<table>
<tr><td>Course Title</td><td colspan="5">Theory of Computation</td></tr>
<tr><td>Course Code</td><td colspan="5">ACSC301</td></tr>
<tr><td>Course Type</td><td colspan="5">Compulsory</td></tr>
<tr><td>Level</td><td colspan="5">BSc (Level 1)</td></tr>
<tr><td>Year / Semester</td><td colspan="5">2^{nd}, 3^{rd} (Fall)</td></tr>
<tr><td>Teacher's Name</td><td colspan="5">Harris Papadopoulos</td></tr>
<tr><td>ECTS</td><td>5</td><td>Lectures / week</td><td>3</td><td>Laboratories/week</td><td>0</td></tr>
<tr><td>Course Purpose</td><td colspan="5">The aim of this course is to provide students with the fundamental ideas of computation and demonstrate how these can contribute to a greater understanding of computing practice. Different models of computation will be examined and the inherent limits of computability will be discussed.</td></tr>
<tr><td>Learning Outcomes</td><td colspan="5">1. Describe in detail what is meant by a finite state automaton, a push-down automaton, a grammar and a Turing Machine.
2. Work out and illustrate the behaviour of a given example of these machines.
3. Design a machine of this type corresponding to a particular language.
4. Reason about the capabilities of such machines and demonstrate that they have limitations.
5. Describe a decision procedure in terms of a language and define a language by using a grammar.</td></tr>
<tr><td>Prerequisites</td><td colspan="2">ACSC191</td><td colspan="2">Corequisites</td><td>None</td></tr>
<tr><td>Course Content</td><td colspan="5">• Introduction to Automata Theory: Introduction to Finite Automata; Structural representations; Automata and complexity; Central concepts of Automata Theory: alphabets, strings, languages, problems.
• Finite Automata: Informal view of Finite Automata; Deterministic Finite Automata; DFA notation; DFA transition function; The language of a DFA; Nondeterministic Finite Automata; NFA transition function; DFA and NFA equivalence; Finite Automata with epsilon transitions; Epsilon-closures; DFA and Epsilon NFA equivalence.
• Regular Expressions and Languages: Operators of Regular Expressions; Building Regular Expressions; Precedence of operators; Converting DFA's to Regular Expressions; Converting Regular Expressions to Automata; Applications of Regular Expressions; Algebraic laws of Regular Expressions.
• Context-Free Grammars and Languages: Definition of Context-Free Grammars; Derivations using a grammar; Language of a grammar; Sentential forms; Constructing parse trees; Inference, derivations and</td></tr>
</table>

	parse trees; From inferences to trees; From trees to derivations; from derivations to recursive inferences; Applications of CFG's; Ambiguity in Grammars and Languages. • **Pushdown Automata:** Definition of Pushdown Automata; Graphical notation for PDA's; Instantaneous descriptions of a PDA; Languages of a PDA; From CFG's to PDA's; From PDA's to CFG's; Deterministic PDA's. • **Turing Machines:** Turing Machine notation; Instantaneous descriptions of TM's; Transition diagrams for TM's; The language of a TM; TM's and Halting; Programming techniques for TM's. • **Undecidability:** The halting problem; Reducing one problem to another; A language that is not recursively enumerable; Recursive languages; Complements of recursive and RE languages; The universal language; Undecidability of the universal language; Undecidable problems about TM's; Post's correspondence problem.
Teaching Methodology	The course is delivered through three hours of lectures per week, which include presentation of new material and demonstration of concepts. Lectures also include in-class exercises to enhance the material learning process and to assess the student level of understanding and provide feedback accordingly. All lecture notes and other material is available to students through the course homepage.
Bibliography	**(a) <u>Textbook:</u>** • John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, ***Introduction to Automata Theory, Languages, and Computation***, 3rd Edition, Addison-Wesley, 2006. **(b) <u>References:</u>** • Michael Sipser, ***Introduction to the theory of computation***, 2nd Edition, Thomson Course Technology, 2006. • Daniel I. A. Cohen, ***Introduction to computer theory***, 2nd Edition, Wiley, 1997.
Assessment	The Students are assessed via continuous assessment throughout the duration of the Semester, which forms the Coursework grade and the final written exam. The coursework and the final exam grades are weighted 40% and 60%, respectively, and compose the final grade of the course. Various approaches are used for the continuous assessment of the students, such as class participation and in class exercises, assignments and tests. The assessment weight, date and time of each type of continuous assessment is being set at the beginning of the semester via the course outline. An indicative weighted continuous assessment of the course is shown below: • Participation Activities (4% of total marks for module) • Three assignments (20% of total marks for module)

	• One closed-book test (16% of total marks for module) • One closed-book, 2-hour exam (60% of total marks for module) Students are prepared for final exam, by revision on the matter taught, problem solving and concept testing and are also trained to be able to deal with time constrains and revision timetable. The criteria considered for the assessment of each type of the continuous assessment and the final exam of the course are: (i) the comprehension of the fundamental concepts and theory of each topic, (ii) the application of the theory in solving related problems and (iii) the ability to apply the above knowledge in complex real-life problems. The final assessment of the students is formative and summative and is assured to comply with the subject's expected learning outcomes and the quality of the course.
Language	**English**