

## 1. COURSE

CS1D1. Discrete Structures I (Mandatory)

## 2. GENERAL INFORMATION

2.1 Credits	: 4
2.2 Theory Hours	: 2 (Weekly)
2.3 Practice Hours	: -
2.4 Duration of the period	: 16 weeks
2.5 Type of course	: Mandatory
2.6 Modality	: ■FaceToFace■
2.7 Prerequisites	: None

## 3. PROFESSORS

Meetings after coordination with the professor

## 4. INTRODUCTION TO THE COURSE

Discrete structures provide the theoretical foundations necessary for computation. These fundamentals are not only useful to develop computation from a theoretical point of view as it happens in the course of computational theory, but also is useful for the practice of computing; In particular in applications such as verification, cryptography, formal methods, etc.

## 5. GOALS

- Apply Properly concepts of finite mathematics (sets, relations, functions) to represent data of real problems.
- Model real situations described in natural language, using propositional logic and predicate logic.
- Determine the abstract properties of binary relations.
- Choose the most appropriate demonstration method to determine the veracity of a proposal and construct correct mathematical arguments.
- Interpret mathematical solutions to a problem and determine their reliability, advantages and disadvantages.
- Express the operation of a simple electronic circuit using Boolean algebra.

## 6. COMPETENCES

- 1) Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions. (**Assessment**)
- 6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (**Assessment**)

## 7. SPECIFIC COMPETENCES

Nospecificoutcomes

## 8. TOPICS

Unit 1: Sets, Relations, and Functions (22)	
Competences Expected:	
Topics	Learning Outcomes
<ul style="list-style-type: none"> <li>• Sets <ul style="list-style-type: none"> <li>– Venn diagrams</li> <li>– Union, intersection, complement</li> <li>– Cartesian product</li> <li>– Power sets</li> <li>– Cardinality of finite sets</li> </ul> </li> <li>• Relations: <ul style="list-style-type: none"> <li>– Reflexivity, simmetry, transitivity</li> <li>– Equivalence relations</li> <li>– Partial order relations and sets</li> <li>– Extremal elements of a partially ordered sets</li> </ul> </li> <li>• Functions <ul style="list-style-type: none"> <li>– Surjections, injections, bijections</li> <li>– Inverses</li> <li>– Composition</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Explain with examples the basic terminology of functions, relations, and sets [Assessment]</li> <li>• Perform the operations associated with sets, functions, and relations [Assessment]</li> <li>• Relate practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context [Assessment]</li> </ul>
Readings : [Gri03], [Rosen2007], [howToProve]	

Unit 2: Basic Logic (14)	
Competences Expected:	
Topics	Learning Outcomes
<ul style="list-style-type: none"> <li>• Propositional logic</li> <li>• Logical connectives</li> <li>• Truth tables</li> <li>• Normal forms (conjunctive and disjunctive)</li> <li>• Validity of well-formed formula</li> <li>• Propositional inference rules (concepts of modus ponens and modus tollens)</li> <li>• Predicate logic <ul style="list-style-type: none"> <li>– Universal and existential quantification</li> </ul> </li> <li>• Limitations of propositional and predicate logic (e.g., expressiveness issues)</li> </ul>	<ul style="list-style-type: none"> <li>• Convert logical statements from informal language to propositional and predicate logic expressions [Usage]</li> <li>• Apply formal methods of symbolic propositional and predicate logic, such as calculating validity of formulae and computing normal forms [Usage]</li> <li>• Use the rules of inference to construct proofs in propositional and predicate logic [Usage]</li> <li>• Describe how symbolic logic can be used to model real-life situations or applications, including those arising in computing contexts such as software analysis (eg, program correctness), database queries, and algorithms [Familiarity]</li> <li>• Apply formal logic proofs and/or informal, but rigorous, logical reasoning to real problems, such as predicting the behavior of software or solving problems such as puzzles [Usage]</li> <li>• Describe the strengths and limitations of propositional and predicate logic [Usage]</li> </ul>
Readings : [Rosen2007], [Gri03], [howToProve]	

Unit 3: Proof Techniques (14)	
Competences Expected:	
Topics	Learning Outcomes
<ul style="list-style-type: none"> <li>• Notions of implication, equivalence, converse, inverse, contrapositive, negation, and contradiction</li> <li>• The structure of mathematical proofs</li> <li>• Direct proofs</li> <li>• Disproving by counterexample</li> <li>• Proof by contradiction</li> <li>• Induction over natural numbers</li> <li>• Structural induction</li> <li>• Weak and strong induction (i.e., First and Second Principle of Induction)</li> <li>• Recursive mathematical definitions</li> <li>• Well orderings</li> </ul>	<ul style="list-style-type: none"> <li>• Identify the proof technique used in a given proof [Assessment]</li> <li>• Outline the basic structure of each proof technique (direct proof, proof by contradiction, and induction) described in this unit [Usage]</li> <li>• Apply each of the proof techniques (direct proof, proof by contradiction, and induction) correctly in the construction of a sound argument [Usage]</li> <li>• Determine which type of proof is best for a given problem [Assessment]</li> <li>• Explain the parallels between ideas of mathematical and/or structural induction to recursion and recursively defined structures [Familiarity]</li> <li>• Explain the relationship between weak and strong induction and give examples of the appropriate use of each [Assessment]</li> <li>• State the well-ordering principle and its relationship to mathematical induction [Familiarity]</li> </ul>
Readings : [Rosen2007], [Vel06], [Sch12], [howToProve]	

Unit 4: Data Representation (10)	
Competences Expected:	
Topics	Learning Outcomes
<ul style="list-style-type: none"> <li>• Numerical representation: sign-magnitude, floating point.</li> <li>• Representation of other objects: sets, relations, functions.</li> </ul>	<ul style="list-style-type: none"> <li>• Explain numerical representations such as sign-magnitude and floating point. [Assessment].</li> <li>• Carry out arithmetic operations using different kinds of representations. [Assessment].</li> <li>• Explain the floating point standard IEEE-754 [Familiarity].</li> </ul>
Readings : [Rosen2007], [Gri03], [howToProve]	

## 9. WORKPLAN

### 9.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

### 9.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

### 9.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

## 10. EVALUATION SYSTEM

\*\*\*\*\* EVALUATION MISSING \*\*\*\*\*

## 11. BASIC BIBLIOGRAPHY

- [Gri03] R. Grimaldi. *Discrete and Combinatorial Mathematics: An Applied Introduction*. 5 ed. Pearson, 2003.
- [Sch12] Edward R. Scheinerman. *Mathematics: A Discrete Introduction*. 3 ed. 2012.