San Pablo Catholic University (UCSP)
Undergraduate Program in
Computer Science
SILABO

Universidad Católica San Pablo
2023-II

CS1D1. Discrete Structures I (Mandatory)

1. General information		
1.1 School	:	Ciencia de la Computación
1.2 Course	:	CS1D1. Discrete Structures I
1.3 Semester	:	1^{er} Semestre.
1.4 Prerrequisites	:	None
1.5 Type of course	:	Mandatory
1.6 Learning modality	:	Face to face
1.7 Horas	:	2 HT; 4 HP;
1.8 Credits	:	4
1.9 Plan	:	Plan Curricular 2016

2. Professors

Lecturer

- Marcela Quispe Cruz <mquispec@ucsp.edu.pe>
  - PhD in Ciencia de la Computación, Pontificia Universidad Católica de Rio de Janeiro, Brasil, 2014.
  - MSc in Ciencia de la Computación, Universidad de Pernambuco, Brasil, 2009.
- Gina Lucia Muñoz Salas <glmunoz@ucsp.edu.pe>
  - MSc in Ciencia de la Computación, Universidad Católica San Pablo, Perú, 2019.
- Kelly Vizconde la Motta <kvizconde@ucsp.edu.pe>
  - MSc in Mag. Ciencia de la Computación, Universidad Católica San Pablo, Perú, 2019.

3. Course foundation

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.

4. Summary

1. Sets, Relations, and Functions 2. Basic Logic 3. Proof Techniques 4. Data Representation

5. Generales 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. Contribution to Outcomes

This discipline contributes to the achievement of the following outcomes:

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. Content

UNIT 1: Sets, Relations, and Functions (22)

Competences:

Content	**Generales Goals**
• Sets – Venn diagrams – Union, intersection, complement – Cartesian product – Power sets – Cardinality of finite sets • Relations: – Reflexivity, simmetry, transitivity – Equivalence relations – Partial order relations and sets – Extremal elements of a partially ordered sets • Functions – Surjections, injections, bijections – Inverses – Composition	• Explain with examples the basic terminology of functions, relations, and sets [Assessment] • Perform the operations associated with sets, functions, and relations [Assessment] • Relate practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context [Assessment]

Readings: Grimaldi (2003), **Rosen2007**, **howToProve**

UNIT 2: Basic Logic (14)

Competences:

Content	**Generales Goals**
• Propositional logic • Logical connectives • Truth tables • Normal forms (conjunctive and disjunctive) • Validity of well-formed formula • Propositional inference rules (concepts of modus ponens and modus tollens) • Predicate logic – Universal and existential quantification • Limitations of propositional and predicate logic (e.g., expressiveness issues)	• Convert logical statements from informal language to propositional and predicate logic expressions [Usage] • Apply formal methods of symbolic propositional and predicate logic, such as calculating validity of formulae and computing normal forms [Usage] • Use the rules of inference to construct proofs in propositional and predicate logic [Usage] • 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] • 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] • Describe the strengths and limitations of propositional and predicate logic [Usage]

Readings: Rosen2007, Grimaldi (2003), **howToProve**

UNIT 3: Proof Techniques (14)

Competences:

Content	**Generales Goals**
• Notions of implication, equivalence, converse, inverse, contrapositive, negation, and contradiction • The structure of mathematical proofs • Direct proofs • Disproving by counterexample • Proof by contradiction • Induction over natural numbers • Structural induction • Weak and strong induction (i.e., First and Second Principle of Induction) • Recursive mathematical definitions • Well orderings	• Identify the proof technique used in a given proof [Assessment] • Outline the basic structure of each proof technique (direct proof, proof by contradiction, and induction) described in this unit [Usage] • Apply each of the proof techniques (direct proof, proof by contradiction, and induction) correctly in the construction of a sound argument [Usage] • Determine which type of proof is best for a given problem [Assessment] • Explain the parallels between ideas of mathematical and/or structural induction to recursion and recursively defined structures [Familiarity] • Explain the relationship between weak and strong induction and give examples of the appropriate use of each [Assessment] • State the well-ordering principle and its relationship to mathematical induction [Familiarity]

Readings: Rosen2007, **Vel06**, Scheinerman (2012), **howToProve**

UNIT 4: Data Representation (10)	
Competences:	
Content	**Generales Goals**
• Numerical representation: sign-magnitude, floating point. • Representation of other objects: sets, relations, functions.	• Explain numerical representations such as sign-magnitude and floating point. [Assessment]. • Carry out arithmetic operations using different kinds of representations. [Assessment]. • Explain the floating point standard IEEE-754 [Familiarity].
Readings: Rosen2007, Grimaldi (2003), **howToProve**	

8. Methodology

1. El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.
2. El profesor del curso presentará demostraciones para fundamentar clases teóricas.
3. El profesor y los alumnos realizarán prácticas
4. Los alumnos deberán asistir a clase habiendo leído lo que el profesor va a presentar. De esta manera se facilitará la comprensión y los estudiantes estarán en mejores condiciones de hacer consultas en clase.

9. Assessment

Continuous Assessment 1 : 20 %

Partial Exam : 30 %

Continuous Assessment 2 : 20 %

Final exam : 30 %

References

Grimaldi, R. (2003). *Discrete and Combinatorial Mathematics: An Applied Introduction*. 5 ed. Pearson.
Scheinerman, Edward R. (2012). *Mathematics: A Discrete Introduction*. 3 ed.