Subject:	Functional Programming	**Code**	72.60
Credits:	3		
Department	Digital Systems and Data	**Version**	2023

Course: Exchange, Computer Science Engineering

Curriculum: IN23, S10 A - Rev18, S10-Rev23, S10 - Rev18

Objectives:

No.	Description
1	- To build simple programs using fundamental notions of the functional paradigm. - Demonstrate simple properties of functional programs using structural induction.

Contents:

Characteristics of the Functional Paradigm. Referential transparency. A method for creating polymorphic and recursive data types. Infinite data structures for higher-order functions. Lambda calculus. Alpha, beta, eta, delta reductions. Fixed point. Forms for evaluation. Introduction to computational semantics. Types of semantics: denotational, axiomatic, operational. Formal method for program specification and verification. Application workshop with a functional language such as Lisp, Hope, Haskel.

Required bibliography:

No.	Description
1	No bibliography has been uploaded.
2	No bibliography has been uploaded.
3	No bibliography has been uploaded.
4	No bibliography has been uploaded.
5	No bibliography has been uploaded.

Optional bibliography:

No.	Description
1	No additional bibliography has been uploaded.
2	No additional bibliography has been uploaded.
3	No additional bibliography has been uploaded.
4	No additional bibliography has been uploaded.

Curriculum:	Functional Programming	**Code**	72.60
Credits:	3		
Department	Digital Systems and Data	**Version**	2023

Course transcript:

No.	Description
1	**1. Preliminary Concepts.** • Review of the notion of programming and the concept of program. • Desirable properties of programs. Reasoning and demonstration of these properties. • Difficulties of the classical programming model for reasoning about programs. • Description of the functional programming model. • Main features of functional languages: referential transparency, high order and currying, and type systems.
2	**2. Functional Paradigm Computing Model.** • Values and expressions. Functions as values. • Mechanisms for defining expressions and values. Equations that define functions. Syntax. • Denotational and operational view of expressions. Computation by reduction models. Semantics. • Reduction orders: applicative reduction and normal reduction. • Hindley-Milner Type System. Basic types. Type constructors. Polymorphism. Syntax for values of each type (characters, tuples, lists, strings, functions). Mechanisms for defining new types and functions on them. Non-recursive algebraic types. • Partial and total functions. • High-order functions. Currying.
3	**3. Formal Techniques** • Demonstration of properties • Notion of ownership and demonstration. Different ways of guaranteeing properties: by construction, by automatic check, by manual demonstration. • Some interesting properties of programs: correctness, termination, program equivalence. • Induction/Recursion. • Inductive definition of sets. • Recursive definition of functions on these sets. • Inductive demonstrations of these functions. • Examples: programs, arithmetic expressions, lists.
4	**4. Application of Concepts: Lists** • Comprehension Lists. Definition and examples. Semantics of comprehension lists by reduction. • Lists as an inductive type. Basic functions on lists (append, head, tail, take, drop, reverse, sort, elem, etc.). • High-order functions on lists. Path pattern: map. Selection pattern: filter. Recursion pattern: foldr. • Demonstration of properties on lists and functions on lists.
5	**5. Type Systems.** • Basic notions. Strong typing systems. Advantages and limitations of programming languages with types. • Types language. Assignment of types to expressions. Interesting properties of this assignment. Inference algorithm. • Mechanisms for defining new types and functions on them. Recursive algebraic types. • Examples: enumerations, lists, binary trees, general trees.

Subject:	Functional Programming	Code	72.60
Credits:	3		
Department	Digital Systems and Data	Version	2023

No.	Description
6	**6. Functional Design Techniques - Program Transformation and Synthesis.** • Motivation. Obtaining programs from specifications. Efficiency improvement, with correction by construction. • Transformation of expressions that use lists by comprehension into expressions using map, filter and concat. • Program transformation and synthesis. Techniques and examples
7	**7. Lambda calculus** • Language definition. Syntax. Definition of substitution. • Model of computation. Notions of alpha, beta and eta reduction. Operational semantics. • Lambda calculus as a theoretical model of functional languages. Representation of booleans, pairs, numbers, lists, and other constructs.

Practical assignments:

No.	Description
1	**PA1.** Introduction to Haskell syntax and the Hugs environment.
2	**PA2.** Expressions and values. Types. Lambda notation Types. Lambda notation.
3	**PA3.** Currifying. High Order. Reduction. Evaluation orders.
4	**PA4.** Demonstrations. Properties of programs. Induction. Recursion.
5	**PA5.** Algebraic types. Pattern matching. Lists.
6	**PA6.** Synonyms of types. Recursive algebraic types. Trees.
7	**PA7.** Abstract data types and modules.
8	**PA8.** High-order functions on lists.
9	**PA9.** Generic recursion patterns. Tree functions.

Subject:	Functional Programming	**Code**	72.60
Credits:	3		
Department	Digital Systems and Data	**Version**	2023

No.	Description
10	**PA10.** Derivation and synthesis of programs.
11	**PA11.** Lazy evaluation. Infinite structures. Partial elements. Duality principles.

Laboratory assignments:

No laboratory assignments.

Professor in charge:	Martinez Lopez, Pablo Ernesto
Head of Department:	Bolo, Mario Enrique