18CS634 Programming Paradigms 3-0-14

Course Introduction

Programming languages have been conceived and designed to provide a *natural* way to express thought processes and thus algorithms. Various interpretations of this *natural* way has resulted in varied styles of languages which we call *Programming Paradigms*. This course intends to focus on following programming paradigms: functional programming, declarative programming, object-oriented programming and programming for concurrency and distributed computing. Students will develop practical competency in languages like Haskell/Clojure, Prolog and Scala representing distinct paradigms. They will also be exposed to choicest selection of other related languages.

Course Objectives

This course is aimed at

1. Introducing students to functional, logic and concurrent programming paradigms.
2. Enabling students to formulate newer abstractions (both procedure and data) in the above paradigms.
3. Familiarizing students with writing functional and concurrent programs.
4. Preparing students to solve complex real-world problems using appropriate programming paradigms.

Course Outcomes

At the end of the course, the students will be able to

CO	Statement
CO1	Understand and apply the concepts that form the basis of functional, logic and concurrent programming paradigms.
CO2	Formulate abstractions with procedures and data in different programming paradigms.
CO3	Write programs in different programming paradigms especially functional, logic and concurrent paradigms.
CO4	Formulate, implement and solve a given problem scenario using appropriate programming paradigm.

CO-PO Mapping

COs	PO1	PO2	PO3	PO4	PO5	PSO1	PSO2	PSO3
CO1	1		1			2	3	3
CO2	2	1	3			2	3	3
CO3	1		1		2	2	3	3
CO4	3	3	3	2		2	3	3

Program Specific Objectives (PSOs) (for Reference)

1. Hone the skill of computer science professionals in areas of research and innovation.
2. Develop experts with high professional competence in recent and futuristic technologies.
3. Create man power with technical competency in computer science to design and develop solutions for the societal problems.

Justification for CO-PO Mapping

PO1 Engineering Knowledge and PO3 Design/Development of Solutions: CO2 and CO4 has been given relatively higher affinity as they are related to the design and formulation of abstraction towards solving (engineering) problems. CO1 and CO3 has been given relatively lower affinity as the knowledge of programming paradigms assist in this process.

PO2 Problem Analysis: Though design and formulation of abstraction (CO2) has high relevance in development of solutions, in an iterative approach it helps in better and refined problem analysis. Formulating and solving a problem scenario (CO4) has high relevance both towards problem analysis and solution development.

PO4 Investigation: Formulating and solving a problem scenario (CO4) necessitates investigation into the problem and hence has high affinity.

PO5 Modern Tool Usage: Writing programs in different programming paradigms pivot around programming languages (which are essentially tools) for respective programming paradigms.

PSO1 Research and Innovation: With increasing complexity of real-world problems, different programming paradigms facilitates innovative way of solving those problems.

PSO2 Professional Competence and PSO3 Technical Competency: Knoweldge of different programming paradigms, formulating abstractions and solving problems have higher affinity to the said program specific objectives.

Syllabus

Overview of different programming paradigms. Functional Programming – functional style, lists and recursion, higher-order functions, infinite lists, recursion without base cases, currying, folds.

Logic Programming – programming “without algorithms”, prolog lists and recursion, constraint progamming. Object-oriented (OO) Programming – What makes a language OO (encapsulation, inheritance and polymorphism), OO languages, functional support in OO languages. Multi-paradigm languages.

Concurrency – concurrency vs parallelism, communicating sequential processes, deadlocks, starvation, Threads vs Actor model for concurrency, Issues with concurrency: safety, liveness, fairness. Concurrency in functional programming.

References:

1. Michael L. Scott, *Programming Language Pragmatics*, Morgan Kaufmann, 4th Edition, 2015.
2. Richard Bird, *Thinking Functionally with Haskell*, Cambridge Universiy Press, 2014.
3. Ivan Bratko, *Prolog Programming for Artificial Intelligence*, Pearson Education, 4th Edition, 2011.
4. Aleksandar Prokopec, *Learning Concurrent Programming in Scala*, Packt Publishing, 2nd Edition, 2017.

Evaluation Pattern: 4D

Assessment	Weight
Quizzes/Tutorial/Lab Assignments	25
Case Study/Mini Project	30
Periodical 1/2	15
End Semester	30