SYLLABUS

1. Data about the program of study

1.1 Institution	The Technical University of Cluj-Napoca
1.2 Faculty	Faculty of Automation and Computer Science
1.3 Department	Computer Science
1.4 Field of study	Computer Science and Information Technology
1.5 Cycle of study	Bachelor of Science
1.6 Program of study/Qualification	Computer science/ Engineer
1.7 Form of education	Full time
1.8 Subject code	32.

2. Data about the subject

2.1 Subject name	***Functional programming***				
2.2 Course responsible/lecturer	Conf. dr. ing. Radu Slavescu – Radu.Razvan.Slavescu@cs.utcluj.ro				
2.3 Teachers in charge of seminars/ laboratory/ project	Ing. Istvan Csaszar Ing. Bogdan Salau				
2.4 Year of study	III	2.5 Semester	1	2.6 Type of assessment (E - exam, C - colloquium, V - verification)	E
2.7 Subject category	*DF – fundamentală, DD – în domeniu, DS – de specialitate, DC – complementară*				DD
	DI – Impusă, DOp – opțională, DFac – facultativă				DI

3. Estimated total time

3.1 Number of hours per week	4	of which:	Course	2	Seminars		Laboratory	2	Project	
3.2 Number of hours per semester	56	of which:	Course	28	Seminars		Laboratory	28	Project	
3.3 Individual study:										
(a) Manual, lecture material and notes, bibliography										18
(b) Supplementary study in the library, online and in the field										10
(c) Preparation for seminars/laboratory works, homework, reports, portfolios, essays										10
(d) Tutoring										4
(e) Exams and tests										2
(f) Other activities:										

3.4 Total hours of individual study (suma (3.3(a)…3.3(f)))	44
3.5 Total hours per semester (3.2+3.4)	100
3.6 Number of credit points	4

4. Pre-requisites (where appropriate)

4.1 Curriculum	Data Structures and Algorithms Course
4.2 Competence	This course assumes no prior knowledge of functional programming, but advises at least one year of programming experience in a regular programming language such as Java, C, C++.

5. Requirements (where appropriate)

5.1. For the course	Basic notions of programming
5.2. For the applications	Linux

6. Specific competence

6.1 Professional competences	**C2** Designing a software system in a functional manner **C2.1** Identifying and describing the software components of the system **C2.2** Explaining the role, interaction and functioning of each component **C2.3** Building software components of some computing systems using design methods, languages, technologies and tools specific to Functional Programming **C2.4** Implementing the software components

	C2.5 Evaluating the functional and non-functional characteristics of the computing systems using specific metrics
6.2 Cross competences	N/A

7. Discipline objective (as results from the *key competences gained*)

7.1 General objective	Increasing the ability to develop more correct and concise code via the functional paradigm (immutability, formal proof of code correctness, easy parellelization) and to understand its underpinning formalism (lambda calculus)
7.2 Specific objectives	Writing better code with the concepts introduced by functional programming: high order functions, lazy evaluation, lambda calculus, infinite structures, recursion as main way of performing iteration, formal proofs

8. Contents

8.1 Lectures	Hours	Teaching methods	Notes
Introduction. Programming Paradigms	2	(Onsite/onlie) Slides, Demos on the whiteboard, New examples Quick individual work (1 minute)	
Basic concepts of programming in Haskell, ML: functions, constants, primitive data types, recursion, tuples, infix operators, evaluation.	2		
Basic concepts: local declarations, polymorphism.	2		
Lists: list construction, basic operations on lists.	2		
Lists: polymorphic equality.	2		
Lists: list operators (generators, filters, list expressions).	2		
Trees: alternative data, pattern matching, exceptions, binary trees (list-tree conversions).	2		
Trees: binary trees (binary search trees, AVL balanced trees, examples (operations on sets)).	2		
Trees: binary trees (examples (Huffman codes)), propositional reasoner (example).	2		
Higher-order functions: anonymous functions, partial application, functions as data, data as functions, combinator functions, functionals for lists (list operator style, style without lists).	2		
Infinite data: lazy evaluation, unbounded objects, circular structures.	2		
Transformation and reasoning: structural induction, equivalence of functions, structural induction on trees, induction on number of nodes, general principle of induction.	2		
Lambda calculus: Lambda notation, conversions, combinators.	2		
Para-functional programming: basic language, mapped expressions, eager expressions.	2		

Bibliography

1. Haskell - A Purely Functional Language, http://www.haskell.org/
2. G. Hutton. **Programming in Haskell, 2nd edition** Cambridge University Press, 2016
3. M. Lipovaca. **Learn You a Haskell for Great Good.** No Starch Press, 2011.
4. Raul Rojas, A Tutorial Introduction to the Lambda Calculus, FU Berlin, 2015

8.2 Applications – Seminars/Laboratory/Project	Hours	Teaching methods	Notes
Introduction in Functional Programming using Elm	2	(Onsite/online) Exercises and problem solving, implementing functions on the computer, Tracing algorithms Miniprojects	
Elm Types	2		
Lists and Recursivity	2		
Higher order Functions in Elm	2		
Miniapplication in Elm	2		
Miniapplication in Elm	2		
Introduction in Haskell and ML	2		
ML Lists, Recursion,.	2		
ML type checking	2		
ML Trees	2		

Haskell – High order functions	2		
Haskell -Lazy evaluation, circular lists, infinite lists.	2		
Lambda Calculus	2		
Final evaluation (Programming in ML and Haskell).	2		

Bibliography

1. www.haskell.org
2. elm-lang.org
3. M. Lipovaca. **Learn You a Haskell for Great Good.** No Starch Press, 2011.
4. A. Cumming A gentle introduction to ML (tutorial online)

** Se vor preciza, după caz: tematica seminariilor, lucrările de laborator, tematica și etapele proiectului.*

9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

The content of the class is similar to the contents taught at other international universities. The students are encouraged to identify elements of functional programming in the current practice of IT companies running at the local level.

10. Evaluation

Activity type	Assessment criteria	Assessment methods	Weight in the final grade
Course	Understanding functional programming elements, Class participation, Homework	(Onsite/online) Written exam/Moodle test	50%
Seminar			
Laboratory	Quantity and quality of code in Elm, Haskell and ML	(Onsite/online) Individual assignments and mini-project	50%
Project			

Minimum standard of performance:
Understanding and code writing for the following concepts; Recursion, High Order Functions, Pattern Matching.
Grade calculus: 50% laboratory + 50% final exam
Conditions for participating in the final exam: Laboratory ≥ 5
Conditions for promotion: Grade ≥ 5

Date of filling in:	**Titulari**	**Titlu Prenume NUME**	**Semnătura**
	Course	Conf. dr. ing. Radu Slavescu	
	Applications	Ing. Istvan Csaszar	
		Ing. Bogdan Salau	

Date of approval in the department	Head of department Prof.dr.ing. Rodica Potolea
Date of approval in the Faculty Council	Dean Prof.dr.ing. Liviu Miclea