

SYLLABUS / FIŞA DISCIPLINEI

1. Information on the study programme / Date despre programul de studii

1.1. Institution / Instituția de învățământ superior	Universitatea de Vest din Timișoara				
1.2. Faculty / Facultatea	Matematică și Informatică				
1.3. Department / Departamentul	Computer Science (Informatică)				
1.4. Study program field	Computer Science (Informatică)				
1.5. Study cycle/ Ciclul de studii	MSc / master				
1.6. Study programme / Programul de studii / calificarea*	Artificial Intelligence and Distributed Computing				

2. Information on the course / Date despre disciplină

2.1. Title of the course / Denumirea disciplinei	Advanced Logic and Functional Programming / Programare Logica si Funcțională Avansată						
2.2. Teacher in charge of the course / Titularul activităților de curs	Prof. Dr. Mircea Marin						
2.3. Teacher in charge of the seminar / Titularul activităților de seminar	Prof. Dr. Mircea Marin						
2.4. Study year / Anul de studii	1	2.5. Semester / Semestrul	1	2.6. Examination type / Tipul de evaluare: E(xam)/C(olloquim)	E	2.7. Course type / Regimul disciplinei: M(andatory)/ E(lective)/ F(acultative)	E

3. Estimated study time (number of hours per semester) / Timpul total estimat (ore pe semestru al activităților didactice)

3.1. Attendance hours per week / Număr de ore pe săptămână	3	out of which din care: 3.2 lecture/curs	2	3.3. seminar/laborator	1
3.4. Attendance hours per semester / Total ore din planul de învățământ	42	out of which: 3.5 lecture / curs	28	3.6. seminar/laborator	14

Distribution of the allocated amount of time / Distribuția fondului de timp*

	hours / ore
Individual study / Studiu după manual, suport de curs, bibliografie și notițe	34
Supplementary documentation at library or using electronic repositories / Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate	21
Preparing for laboratories, homework, reports etc. / Pregătire seminarii/laboratoare, teme, referate, portofolii și eseuri	37
Exams / Examinări	8
Tutoring / Tutorat	8
3.7. Total number of hours of individual study / Total ore studiu individual	108

3.8. Total number of hours per semester / Total ore pe semestru	150
3.9. Number of credits (ECTS) / Număr de credite	6

4. Prerequisites (if it is the case) / Precondiții (acolo unde e cazul)

4.1. curriculum / de curriculum	Functional and Logic Programming / Programare Logică și funcțională
4.2. skills / de competențe	Recursive thinking, top-down and bottom-up design of applications / Gândire recursivă, abordări top-down și bottom-up în dezvoltarea aplicațiilor

5. Requirements (if it is the case) / Condiții (acolo unde e cazul)

5.1. for the lecture / de desfășurare a cursului	room with whiteboard and video projector @online: internet connection, video camera, microphone
5.2. for the seminar, laboratory / de desfășurare a seminarului/laboratorului	computers with preinstalled languages Racket, Haskell, and SWI Prolog

6. Acquired skills / Competențe specifice acumulate

Professional skills / Competențe profesionale	New tools, skills and problem-solving methods in the discipline of software engineering, using Functional Programming and Logic Programming. Emphasis will be put on using tools and concepts that are not available in other programming styles.
Transversal skills / Competențe transversale	identify problems, create solutions, innovate and improve current practices

7. Objectives of the course / Obiectivele disciplinei (reiese din grila competențelor specifice acumulate)

7.1. General objective / Obiectivul general al disciplinei	Students learn techniques of functional programming using the Lisp (or, more precisely, Racket) and Haskell language and (constraint) logic programming in the Prolog language. All these languages are declarative in the sense that the programmer describes the problem to be solved instead of indicating the exact sequence of actions to be taken for solving the problem. These languages are used often in AI-related fields, such as machine learning and agent systems.
7.2. Specific objectives / Obiectivele specifice	<i>Knowledge Objectives (KO):</i> (1) Write programs using functional programming concepts, such as Algebraic Data Types and Type Classes; (2) Use Functional Programming to achieve high levels of abstraction and program correctness; (3) Solve appropriate problems using Logic Programming.

	<p><i>Ability Objectives (AO):</i> Compare and contrast programming programming styles, such as Functional Programming, Logic Programming and Procedural Programming, and choose the one that is more appropriate for your problem.</p> <p><i>Attitudinal Objectives (AtO):</i> identify the advantages and disadvantages of the alternative approaches to solve a problem.</p>
--	---

8. Content / Conținuturi*

8.1. Lecture / Curs	Teaching strategies / Metode de predare	Remarks, details / Observații
L1.(2h) Introduction to declarative programming languages. Comparison with classical imperative programming languages. Main features of logic programming and functional programming.	Lecture, conversation, illustration	
L2.(2h) Introduction to strict functional programming using the Racket language. Basic language idioms, atoms, lists, recursion.	Lecture, conversation, illustration	
L3.(2h) Racket: lambda abstraction, built-in functions, advanced data structures	Lecture, conversation, illustration	
L4.(2h) Efficient computation techniques. Optimizations of tail-recursive definitions in Racket; built-in higher-order functions	Lecture, conversation, illustration	
L5.(2h) Racket: state space search; applications in AI	Lecture, conversation, illustration	
L6.(2h). Lazy functional programming with Haskell: types, patterns, built-in functions, lambda abstractions	Lecture, conversation, illustration	

L7.(2h). Haskell: advanced programming capabilities in comparison with Racket	Lecture, conversation, illustration	
L8.(2h). Logic programming and Prolog. Facts, rules and queries. Recursion. Query answering.	Lecture, conversation, illustration	
L9.(2h). Functions. Unification. List operations.	Lecture, conversation, illustration	
L10.(2h). Prolog and logic: clauses, Herbrand base, interpretaton, model, the closed-world assumption,	Lecture, conversation, illustration	
L11.(2h). Cut and negation. Extralogicl operators, arithmetics.	Lecture, conversation, illustration	
L12.(2h). Combinatorial search in Prolog..	Lecture, conversation, illustration	
L13.(2h) Constraint logic programming	Lecture, conversation, illustration	
L14.(2h) Constraint logic programming	Lecture, conversation, illustration	
Recommended bibliography / Bibliografie		
[1] Paul Hudak: The Haskell School of Expression: Learning Functional Programming through Multimedia. Cambridge University Press. 2000.		
[2] Simon Thompson. <i>Haskell: The Craft of Functional Programming</i> . Pearson Education. 2011.		
[3] Daniel P. Friedman, Mitchell Wand, Christopher T. Haynes: <i>Essentials of Programming Languages. Second Edition</i> . The MIT Press. 2001.		
[4] Isabela Drămnesc – slides http://staff.fmi.uvt.ro/~isabela.dramnesc		
[5] Ulf Nilsson, Jan Maluszynski, Logic, Programming and Prolog, 2nd Edition, copyright Ulf Nilsson and Jan Maluszynski, 2000.		
[6] Bratko - PROLOG. Programming for Artificial Intelligence, Addison Wesley, third edition, 2001.		
[7] http://www.swi-prolog.org/		

8.2. Seminar, lab / Seminar, laborator	Teaching/learning strategies / Metode de predare/invățare	Remarks, details / Observații
1. Introduction to Racket and its programming environment. Examples of recursive programming techniques.	Dialogue, programming assignments for students.	
2. Haskell: the programming environment; characteristic features of lazy functional programming.	Dialogue, programming assignments for students.	
3. State space search; applications in AI.	Project assignment, dialogue, collaborative learning.	
4. Prolog programs as databases of knowledge. Facts, rules, and queries. Unification,	Dialogue, collaborative learning, programming assignments	
5. . Search algorithms	Dialogue, program-ming assignments.	
6. Constraint logic programming.	Dialogue, collaborative learning, problem solving	
7. Individual and collective projects.	Dialogue, collaborative learning, problem solving	
<p>Recommended bibliography / Bibliografie</p> <p>[1] Daniel P. Friedman, Mitchell Wand, Christopher T. Haynes: <i>Essentials of Programming Languages. Second Edition</i>. The MIT Press. 2001.</p> <p>[2] Paul Hudak: The Haskell School of Expression: Learning Functional Programming through Multimedia. Cambridge University Press. 2000.</p> <p>[3] Simon Thompson. <i>Haskell: The Craft of Functional Programming</i>. Pearson Education. 2011.</p> <p>[4] Isabela Drămnesc – slides http://staff.fmi.uvt.ro/~isabela.dramnesc</p> <p>[5] Ulf Nilsson, Jan Maluszynski, Logic, Programming and Prolog, 2nd Edition, copyright Ulf Nilsson and Jan Maluszynski, 2000.</p> <p>[6] http://www.swi-prolog.org/</p> <p>[7] https://www.haskell.org</p>		

**9. Correlations between the content of the course and the requirements of the IT field /
 Coroborarea conținuturilor disciplinei cu așteptările reprezentanților comunității epistemice,
 asociațiilor profesionale și angajatorilor reprezentativi din domeniul aferent programului**

The content of the course corresponds with curricula of other universities in Romania or the European Union.
 The contents of practical work (labs) meet the requirements of the local labor market. / Conținutul disciplinei corespunde curriculei din alte centre universitare, din țară sau Uniunea Europeană. Conținuturile practice (lucrări de laborator) corespund cerințelor de pe piața muncii locală.

10. Evaluation / Evaluare*

Activity / Tip de activitate	10.1. Evaluation criteria / Criterii de evaluare**	10.2. Evaluation methods / Metode de evaluare***	10.3. Weight in the averaged mark / Pondere din nota finală
10.4. Lecture / Curs	Evaluation of the theoretical knowledge and practical skills / Evaluarea cunoștințelor teoretice și a abilităților practice	Exam	30%
	Periodic assessment of the knowledge acquired from attending the lectures / Evaluare periodică a cunoștințelor	Tests, homework / Teste, teme de casa	20%
10.5. Seminar/ lab	Homework and project activities cover specific parts, as they were exposed during the semester, and their solution is based on laboratory activity / Temele de casă și proiectul acoperă părți specifice	Individual or group project / Proiect individual sau proiect de grup.	50%
10.6. Minimal knowledge for passing / Standard minim de performanță			
Lecture: the capacity to understand the basic principles and concepts of Logical and Functional Programming. An average level of understanding the principles of Constraint Programming. / Capacitatea de a înțelege principiile și concepțele de bază ale programării logice și funcționale			
Lab: the ability to solve problems of average complexity / abilitatea de a rezolva probleme de complexitate medie.			

Date/ Data completării
 23.09.2022

Signature (lecture) /
 Semnătura titularului de curs

Signature (seminar)
 Semnătura titularului de seminar

Prof.dr. Mircea Marin

Prof.dr. Mircea Marin

Signature (director of the department)
 Semnătura directorului de departament
 Conf.dr. Flavia Micota