SYLLABUS

Functional and Logic Programming

University year 2025-2026

1. Information regarding the programme

1.1. Higher education institution	Babeș Bolyai University
1.2. Faculty	Faculty of Mathematics and Computer Science
1.3. Department	Department of Computer Science
1.4. Field of study	Computers and Information Technology
1.5. Study cycle	Bachelor
1.6. Study programme/Qualification	Artificial Intelligence
1.7. Form of education	Full Time

2. Information regarding the discipline

2.1. Name of the discipline	Functional and Logic Programming					Discipline code	
2.2. Course coordinator				Assist.Prof.Dr. Cristian-Paul Bara			
2.3. Seminar coordinator				Assist.Prof.Dr. Cristian-Paul Bara			
2.4. Year of study	3	2.5. Semester	5	2.6. Type of evaluation	C	2.7. Discipline regime	Compulsory DD

3. Total estimated time (hours/semester of didactic activities)

3.1. Hours per week	**4**	of which: 3.2 course	**2**	3.3 seminar/laboratory/project	**1 LP 1 S**
3.4. Total hours in the curriculum	56	of which: 3.5 course	28	3.6 seminar/laboratory/project	**28**
Time allotment for individual study (ID) and self-study activities (SA)					**hours**
Learning using manual, course support, bibliography, course notes (SA)					22
Additional documentation (in libraries, on electronic platforms, field documentation)					18
Preparation for seminars/labs, homework, papers, portfolios and essays					27
Tutorship					11
Evaluations					16
Other activities:					-
3.7. Total individual study hours	**94**				
3.8. Total hours per semester	**150**				
3.9. Number of ECTS credits	**6**				

4. Prerequisites (if necessary)

4.1. curriculum	• Fundamentals of Programming • Mathematical Foundations of Computer Science
4.2. competencies	• Average programming skills in a high level programming language

5. Conditions (if necessary)

5.1. for the course	• Students will attend the course with their mobile phones shut down • Students will attend the course with their laptops shut down; students with special needs will discuss these at the beginning of the semester
5.2. for the seminar /lab activities	• Students will attend the lab with their mobile phones shut down

	• Laboratory with computers; high level declarative programming language environment (CLisp, SWIProlog)

6.1. Specific competencies acquired [1]

Professional/essential competencies	•
Transversal competencies	•

6.2. Learning outcomes

Knowledge	The student knows: ...
Skills	The student is able to ...
Responsibility and autonomy:	The student has the ability to work independently to obtain ...

7. Objectives of the discipline (outcome of the acquired competencies)

7.1 General objective of the discipline	• Get accustomed with basic notions, concepts, theories and models of new programming paradigms (functional and logic programming)

[1] One can choose either competences or learning outcomes, or both. If only one option is chosen, the row related to the other option will be deleted, and the kept one will be numbered 6.

7.2 Specific objective of the discipline	• Get accustomed with a programming language for each of these paradigms (Common Lisp and Turbo Prolog) • Acquire the idea of using these programming paradigms based on the applications' necessities • Assure the necessary base for approaching certain advanced courses • Ability to apply declarative programming techniques to different real life problems • Ability to model phenomena using declarative techniques • Improved programming abilities using the declarative paradigm

8. Content

8.1 Course	Teaching methods	Remarks
1. Basic elements of Prolog. Facts and rules in Prolog. Goals. The control strategy in Prolog. Variables and composed propositions. Anonymous variables. Rules for matching. The flow model. Sections of a Prolog program. Examples	Exposure: description, explanation, examples, discussion of case studies	
2. The Prolog program. Predefined domains. Internal and external goals. Multiple arity predicates. The IF symbol (Prolog) and the IF instruction (other languages). Compiler directives. Arithmetic expressions and comparisons. Input/output operations. Strings	Exposure: description, explanation, examples, discussion of case studies	
3. Backtracking. The backtracking control. The "fail" and "!"(cut) predicates. Using the "!" predicate. Type of cuts. The "not" predicate. Lists in Prolog. Recursion. Examples for backtracking in Prolog. Finding all solutions in the same time. Examples of predicates in Prolog. Non-deterministic predicates	Exposure: description, explanation, examples, discussion of case studies	
4. Composed objects and functors. Unifying composed objects. Arguments of multiple types; heterogeneous lists. Comparisons for composed objects. Backtracking with cycles. Examples of recursive procedures. The stack frame. Optimization using the "tail recursion". Using the "cut" predicate in order to keep the "tail recursion".	Exposure: description, explanation, examples, discussion of case studies	
5. Recursive data structures. Trees as data structures. Creating and traversing a tree. Search trees. The internal database of Prolog. The "database" section. Declaration of the internal database. Predicates concerning operations with the internal database	Exposure: description, explanation, examples, discussion of case studies	
6. Advanced issues of Backtracking in Prolog. Files management in Prolog.	Exposure: description, explanation, examples, discussion of case studies	
7. Programming and programming languages. Imperative programming vs. declarative programming. Introduction. The importance of the functional programming as a new programming methodology. History and presentation of LISP	Exposure: description, explanation, examples, discussion of case studies	

8. Basic elements in Lisp. Dynamic data structures. Syntactic and semantic rules. Functions' classification in Lisp. Primitive functions in Lisp. Basic predicates in Lisp.	Exposure: description, explanation, examples, discussion of case studies	
9. Predicates for lists; for numbers. Logic and arithmetic functions. Defining user functions. The conditional form. The collecting variable method. Examples	Exposure: description, explanation, examples, discussion of case studies	
10. Symbols' managing. Other functions for lists' accessing. OBLIST and ALIST. Destructive functions. Comparisons. Other interesting functions. Examples	Exposure: description, explanation, examples, discussion of case studies	
11. Definitional mechanisms. The EVAL form. Functional forms; the functions FUNCALL and APPLY. LAMBDA expressions, LABEL expressions. Generators, functional arguments. MAP functions. Iterative forms. Examples	Exposure: description, explanation, examples, discussion of case studies	
12. Other elements in Lisp. Data structures. Macrodefinitions. Optional arguments. Examples	Exposure: description, explanation, examples, discussion of case studies	
13.-14. Graded paper in Logic and Functional Programming	Written test	

Bibliography

1. CZIBULA G., POP H.F., Elemente avansate de programare in Lisp si Prolog. Aplicatii in Inteligenta Artificiala, Editura Albastra, Cluj-Napoca, 2012
2. POP H.F., SERBAN G., Programare in Inteligenta Artificiala - Lisp si Prolog, Editura Albastra, ClujNapoca, 2003
3. http://www.ifcomputer.com/PrologCourse, Lecture on Prolog
4. http://www.lpa.co.uk, Logic Programming
5. FIELD A., Functional Programming, Addison Wesley, New York, 1988.
6. WINSTON P.H., Lisp, Addison Wesley, New York, 2nd edition, 1984.

8.2 Seminar / laboratory	Teaching methods	Remarks
S1. Recursion	• Explanation • Conversation • Modelling • Case studies	
S2. Lists in Prolog	• Explanation • Conversation • Modelling • Case studies	
S3. Processing of heterogeneous lists in Prolog	• Explanation • Conversation • Modelling • Case studies	
S4. Backtracking in Prolog	• Explanation • Conversation • Modelling	

	• Case studies	
S5. Lists processing in LISP	• Explanation • Conversation • Modelling • Case studies	
S6. MAP functions in LISP	• Explanation • Conversation • Modelling • Case studies	
S7. Recap	• Explanation • Conversation • Modelling • Case studies	
Lab 1: Recursive algorithms in Pseudocode	Explanation, dialogue, testing data discussion, case studies	Problem given at lab 1 and submitted at lab 1
Lab 2: Lists in Prolog	Explanation, dialogue, testing data discussion, case studies	Problem given at lab 1 and submitted at lab 2
Lab 3: Trees in Prolog. Lists management in Prolog.	Explanation, dialogue, testing data discussion, case studies	Problem given at lab 2 and submitted at lab 3
Lab 4: Backtracking in Prolog	Explanation, dialogue, testing data discussion, case studies	Problem given at lab 3 and submitted at lab 4
Lab 4: Practical test in Prolog	Practical test	One hour
Lab 5: Recursive programming in Lisp	Explanation, dialogue, testing data discussion, case studies	Problem given at lab 4 and submitted at lab 5
Lab 6: Recursive programming in Lisp	Explanation, dialogue, testing data discussion, case studies	Problem given at lab 5 and submitted at lab 6
Lab 7: Using MAP functions in Lisp	Explanation, dialogue, testing data discussion, case studies	Problem given at lab 6 and submitted at lab 7
Lab 7: Practical test in Lisp	Practical test	One hour

Bibliography

1. CZIBULA G., POP H.F., Elemente avansate de programare in Lisp si Prolog. Aplicatii in

Inteligenta Artificiala, Editura Albastra, Cluj-Napoca, 2012
2. Product documentation: Gold Common Lisp 1.01 si 4.30, XLisp, Free Lisp.
3. Product documentation: Turbo Prolog 2.0, Logic Explorer, Sicstus Prolog.
4. http://www.swi-prolog.org

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

- The course respects the IEEE and ACM Curricula Recommendations for Computer Science studies;
- The course exists in the studying program of all major universities in Romania and abroad;
- The content of the course is concordant with partial competencies for possible occupations from the Grid 1 - RNCIS.

10. Evaluation

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.4 Course	- know the basic principle of the domain; - apply the course concepts - problem solving	Written test in Logic and Functional Programming	50%
10.5 Seminar/laboratory	- activity at seminaries	Evaluation of seminaries activity	Bonus 10%
	- be able to implement course concepts and algorithms - apply techniques for different classes of programming languages	Programs documentation and delivery	25%
		Practical test in Prolog (one hour at lab 4)	12.5%
		Practical test in Lisp (one hour at lab 7)	12.5%
10.6 Minimum standard of performance			

- Each student has to prove that (s)he acquired an acceptable level of knowledge and understanding of the subject, that (s)he is capable of stating these knowledge in a coherent form, that (s)he has the ability to establish certain connections and to use the knowledge in solving different problems.
- In order to pass the course, the following minimal criteria apply collectively: at least grade 5 (from a scale of 1 to 10) at the written test; at least grade 5 (from a scale of 1 to 10) computed as final grade average, attendance of at least 5 seminars and at least 6 labs as scheduled during the semester.

11. Labels ODD (Sustainable Development Goals)[2]

[2] Keep only the labels that, according to the *Procedure for applying ODD labels in the academic process*, suit the discipline and delete the others, including the general one for *Sustainable Development* – if not applicable. If no label describes the discipline, delete them all and write „*Not applicable*.".

Not applicable.

Date:	Signature of course coordinator	Signature of seminar coordinator
05.10.2025	Assist.Prof.Dr. Cristian-Paul Bara	Assist.Prof.Dr. Cristian-Paul Bara

Date of approval:
...

Signature of the head of department

Assoc.prof.phd. Adrian STERCA