SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	**Babeş Bolyai University**
1.2 Faculty	**Faculty of Mathematics and Computer Science**
1.3 Department	**Department of Computer Science**
1.4 Field of study	**Computer Science**
1.5 Study cycle	**Bachelor**
1.6 Study programme / Qualification	**Computer Science**

2. Information regarding the discipline

2.1 Name of the discipline	**Functional and Logic Programming**						
2.2 Course coordinator	**Prof.Dr. Horia F. Pop**						
2.3 Seminar coordinator	**Prof.Dr. Horia F. Pop**						
2.4. Year of study	**2**	2.5 Semester	**3**	2.6. Type of evaluation	**C**	2.7 Type of discipline	**Compulsory**

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

3.1 Hours per week	3	Of which: 3.2 course	2	3.3 seminar/laboratory	1 lab
3.4 Total hours in the curriculum	42	Of which: 3.5 course	28	3.6 seminar/laboratory	14
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					28
Additional documentation (in libraries, on electronic platforms, field documentation)					25
Preparation for seminars/labs, homework, papers, portfolios and essays					28
Tutorship					7
Evaluations					20
Other activities:					-

3.7 Total individual study hours	108
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
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. Specific competencies acquired

Professional competencies	C1.1 Adequate description of programming paradigms and specific language mechanisms, as well as identification of differences between semantic and syntactic aspects. C1.3 Elaboration of adequate source codes and unitary testing of some components in a known programming language, based on given design specifications. C1.5 Development of program units and elaboration of corresponding documentations.
Transversal competencies	CT1 Application of efficient and organized work rules, of responsible attitudes towards the didactic-scientific domain, to creatively value one's own potential, with the respect towards the principles and norms of professional etic. CT3 Use of efficient methods and techniques to learn, inform, research and develop the abilities to value the knowledge, to adapt to requirements of a dynamic society and to communicate in Romanian language and in a language of international circulation.

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)
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 flux 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	Exposure: description, explanation, examples, discussion of case studies	

recursive procedures. The stack frame. Optimization using the "tail recursion". Using the "cut" predicate in order to keep the "tail recursion".		
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, proofs, debate, dialogue	
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. Macro-definitions. 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. 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		
8.2 Seminar	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	

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

8.3 Laboratory	Teaching methods	Remarks
Lab 1: Recursive algorithms in Pseudocode	Explanation, dialogue, testing data discussion, case studies	Deliver problem at lab 1 Receive problem for lab 2
Lab 2: Lists in Prolog	Explanation, dialogue, testing data discussion, case studies	Deliver problem at lab 2 Receive problem for lab 3
Lab 3: Trees in Prolog. Lists management in Prolog.	Explanation, dialogue, testing data discussion, case studies	Deliver problem at lab 3 Receive problem for lab 4
Lab 4: Backtracking in Prolog	Explanation, dialogue, testing data discussion, case studies	One hour Deliver problem at lab 4 Receive problem for lab 5
Lab 4: Practical test in Prolog	Practical test	One hour
Lab 5: Recursive programming in Lisp	Explanation, dialogue, testing data discussion, case studies	Deliver problem at lab 5 Receive problem for lab 6

Lab 6: Using MAP functions in Lisp.	Explanation, dialogue, testing data discussion, case studies	Deliver problem at lab 6 Receive problem for lab 7
Lab 7: Practical test in Lisp	Practical test	One hour

Bibliography

5. CZIBULA G., POP H.F., Elemente avansate de programare in Lisp si Prolog. Aplicatii in Inteligenta Artificiala, Editura Albastra, Cluj-Napoca, 2012
6. Product documentation: Gold Common Lisp 1.01 si 4.30, XLisp, Free Lisp.
7. Product documentation: Turbo Prolog 2.0, Logic Explorer, Sicstus Prolog.
8. 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

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
10.4 Course	- know the basic principle of the domain; - apply the course concepts - problem solving	Written test in Logic and Functional Programming	60%
10.5 Seminar activities	- seminaries activity - seminaries attendance (penalties of 15% of the seminar grade for an absence, one excused)	Evaluation of seminaries activity	10%
10.6 Lab activities	- be able to implement course concepts and algorithms - apply techniques for different classes of programming languages - no delays in submissions	Programs documentation and delivery	10%
		Practical test in Prolog (one hour at lab 4)	10%
		Practical test in Lisp (one hour at lab 7)	10%
10.7 Minimum performance standards			
➢ 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. ➢ For final passing of the course, the following minimal criteria apply collectively: at least grade 4 (from a scale of 1 to 10) at the written test; successful delivery of 4 out of 8 lab problems; at least grade 5 (from a scale of 1 to 10) computed as final grade average.			

Date	Signature of course coordinator	Signature of seminar coordinator
30.04.2015	Prof.Dr. Horia F. POP	Prof.Dr. Horia F. POP
Date of approval		Signature of the head of department
.....		Prof.Dr. Bazil Pârv