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	**Computers and Information Technology**
1.5 Study cycle	**Bachelor**
1.6 Study programme / Qualification	**Information Engineering**

2. Information regarding the discipline

2.1 Name of the discipline (en) (ro)			**Formal Languages and Automata**				
2.2 Course coordinator			**Prof.PhD. Simona Motogna**				
2.3 Seminar coordinator			**Prof.PhD. Simona Motogna**				
2.4. Year of study	**4**	2.5 Semester	**7**	2.6. Type of evaluation	**E**	2.7 Type of discipline	**Compulsory DD**
2.8 Code of the discipline		MLE5181					

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

3.1 Hours per week	5	Of which: 3.2 course	2	3.3 seminar/laboratory	1S 2LP
3.4 Total hours in the curriculum	70	Of which: 3.5 course	28	3.6 seminar/laboratory	42
					hours
Learning using manual, course support, bibliography, course notes					25
Additional documentation (in libraries, on electronic platforms, field documentation)					25
Preparation for seminars/labs, homework, papers, portfolios and essays					20
Tutorship					5
Evaluations					5
Other activities:					

3.7 Total individual study hours	80
3.8 Total hours per semester	150
3.9 Number of ECTS credits	6

4. Prerequisites (if necessary)

4.1. curriculum	•
4.2. competencies	• Average programming skills in a high level programming language

5. Conditions (if necessary)

5.1. for the course	•

5.2. for the seminar /lab activities	- Laboratory with computers; high level programming language environment (.NET or any Java environement a.s.o.)

6. Specific competencies acquired

Professional competencies	C1.2 Using specific theories and tools (algorithms, schemes, models, protocols, etc.) for explaining the structure and the functioning of hardware, software and communication systems C1.3 Building models for various components of computing systems C1.5 Providing theoretical background for the characteristics of the designed systems
Transversal competencies	CT1 Honorable, responsible, ethical behavior, in the spirit of the law, to ensure the professional reputation CT3 Demonstrating initiative and pro-active behavior for updating professional, economical and organizational culture knowledge

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

7.1 General objective of the discipline	- Be able to understand compiler design and the main theoretical concepts in compiler theory - Improved programming skills
7.2 Specific objective of the discipline	- Understand and work with formal languages concepts: Chomsky hierarchy; regular grammars, finite automata and the equivalence between them; context-free grammars, push-down automata and their equivalence; Turing machines - Understand and work with compilers concepts: scanning, parsing

8. Content

8.1 Course	Teaching methods	Remarks
1. Introductory notions of formal languages. Grammars and Finite Automata [1,2]	Exposure: description, explanation, examples, debate, dialogue	
2. Minimization of FA. Elimination of ε- moves [1,2]	Exposure: description, explanation, examples, proofs	
3. Regular languages, regular expressions, equivalence between finite automata, regular grammars and regular expressions. Pumping lemma [1,2]	Exposure: description, explanation, examples, proofs	
4. Equivalence between finite automata, regular grammars and regular expressions (cont.). Pumping lemma [1,2]	Exposure: description, explanation, examples, proofs	
5. Context-free grammars (CFG), syntax tree. Equivalent transformations of CFG [1,2]	Exposure: description, explanation, examples, discussion of case studies	
6. Push-down automata (PDA). Equivalence between CFG and PDA [1,2]	Exposure: description, explanation, examples, discussion of case studies	
7. Scanning (Lexical Analysis) [3,4,7,8]	Exposure: description, explanation, examples, discussion of case	

	studies	
8. Parsing: general notions, classification. Recursive-descendant parser [3,4,7,8]	Exposure: description, explanation, examples, discussion of case studies	
9. LL(1) parser [3,4,7,8]	Exposure: description, explanation, examples, discussion of case studies	
10. LR(k) Parsing method. LR(0) parser [3,4,7,8]	Exposure: description, explanation, examples, discussion of case studies	
11. SLR, LR(1), LALR parser [3,4,7,8]	Exposure: description, explanation, examples, discussion of case studies	
12. Scanner generator (lex); Parser generators (yacc) [4]	Exposure: description, examples, discussion of case studies, live demo	
13. Turing machines [1,2]	Exposure: description, explanation, examples, discussion of case studies	
14. General Structure of a compiler. Compiler phases [3,4,7,8]	Exposure: description, explanation, examples, discussion of case studies	

Bibliography

1. A.V. AHO, D.J. ULLMAN - Principles of computer design, Addison-Wesley, 1978.
2. A.V. AHO, D.J. ULLMAN - The theory of parsing, translation and compiling, Prentice-Hall, Engl. Cliffs., N.J., 1972, 1973.
3. D. GRIES - Compiler construction for digital computers,, John Wiley, New York, 1971.
4. MOTOGNA, S. – Metode de proiectare a compilatoarelor, Ed. Albastra, 2006
5. SIPSER, M., Introduction to the theory of computation, PWS Pulb. Co., 1997
6. CSÖRNYEI ZOLTÁN, Bevezetés a fordítóprogramok elméletébe, I, II., ELTE, Budapest, 1996
7. L.D. SERBANATI - Limbaje de programare si compilatoare, Ed. Academiei RSR, 1987.
8. CSÖRNYEI ZOLTÁN, Fordítási algoritmusok, Erdélyi Tankönyvtanács, Kolozsvár, 2000.
10. GRUNE, DICK - BAL, H. - JACOBS, C. - LANGENDOEN, K.: Modern Compiler Design, John Wiley, 2000

8.2 Seminar	Teaching methods	Remarks
1. Specification of a programming language; BNF notation	Explanation, dialogue, case studies	
2. Grammars; language generated by a grammar; grammar corresponding to a language	Dialogue, debate, case studies, examples, proof	
3. Finite automata: language accepted by a FA; FA corresponding to a language	Dialogue, debate, case studies, examples, proof	
4. Transformations: finite automata – regular grammars	Dialogue, debate, case studies, examples, proof	
5. Context free grammars; LL(1) parser	Dialogue, debate, case	

	studies, examples, proof	
6. LR(0) parser	Dialogue, debate, case studies, examples, proof	
7. Push Down automata	Dialogue, debate, case studies, examples, proof	

Bibliography

1. A.V. AHO, D.J. ULLMAN - Principles of computer design, Addison-Wesley, 1978.
2. A.V. AHO, D.J. ULLMAN - The theory of parsing, translation and compiling, Prentice-Hall, Engl. Cliffs., N.J., 1972, 1973.
3. MOTOGNA, S. – Metode de proiectare a compilatoarelor, Ed. Albastra, 2006
4. G. MOLDOVAN, V. CIOBAN, M. LUPEA - Limbaje formale si automate. Culegere de probleme, Univ. Babes-Bolyai, Cluj-Napoca, 1996.

8.3 Laboratory	Teaching methods	Remarks
1. Task 1: Specify a mini-language and implement scanner using lex 1.1: Mini language specification (BNF notation)	Explanation, dialogue, case studies	
2. Task 1: Specify a mini-language and implement scanner 1.2: implement scanner using lex	Explanation, dialogue, case studies	
3. Task 2: regular grammars + finite automata + transformations 2.1: Define data structures for RG and FA; implement transformations	Explanation, dialogue, case studies	
4. Task 2: regular grammars + finite automata + transformations 2.2: Main program, testing + delivery	Testing data discussion, evaluation	
5. Task 3: context free grammars + equivalent transformations of cfg 3.1: extend task 2 for cfg; implement transformations	Explanation, dialogue, case studies	
6. Task 3: context free grammars + equivalent transformations of cfg 3.2: main program and testing	Testing data discussion, evaluation	
7. Task 4: Parser implementations 4.1: define data structures and architecture of application	Explanation, dialogue, case studies	One of: descendant recursive, LL(1), LR(0), SLR
8. Task 4: Parser implementations 4.2: implement main functions in parsing	Explanation, dialogue, case studies	Task 4 is developed in teams of 2 students
9. Task 4: Parser implementations 4.3: main program and module integration	Explanation, dialogue, case studies	
10. Task 4: Parser implementations 4.4: testing on small formal grammars	Testing data discussion, evaluation	
11. Task 4: Parser implementations 4.5: testing on mini-language; delivery	Testing data discussion, evaluation	
12. Task 5: use tools for lexer and parser generator: lex, yacc 5.1: implementation	Explanation, dialogue, case studies	
12. Task 5: use tools for lexer and parser generator: lex, yacc 5.2: integration + delivery	Testing data discussion, evaluation	
14. Final presentation of lab work	Explanation, dialogue, case studies	

Bibliography

1. A.V. AHO, D.J. ULLMAN - Principles of computer design, Addison-Wesley, 1978.
2. A.V. AHO, D.J. ULLMAN - The theory of parsing, translation and compiling, Prentice-Hall, Engl. Cliffs., N.J., 1972, 1973.
3. MOTOGNA, S. – Metode de proiectare a compilatoarelor, Ed. Albastra, 2006
4. G. MOLDOVAN, V. CIOBAN, M. LUPEA - Limbaje formale si automate. Culegere de probleme, Univ. Babes-Bolyai, Cluj-Napoca, 1996.

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 Curriculla Recommendations for Information Engineering studies;
- The course exists in the studying program of all major universities in Romania and abroad;
- The content of the course is considered the software companies as important for average programming skills

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 exam	60%
10.5 Seminar/lab activities	- be able to apply algorithms, understand examples - problem solving	problems solved - homeworks delivered - continuous observations during semester	10%
	- be able to implement course concepts and algorithms - apply techniques for different classes of programming languages	-Practical examination during all semester -documentation - portofolio -continous observations	30%

10.6 Minimum performance standards

- Attend 75% of seminar activities during semester AND attend 90% of lab activities during semester
- At least grade 5 (from a scale of 1 to 10) at both written exam and laboratory work.
- Basic understanding of formal languages concepts: grammar, finite automata, push down automata, regular expression; understand compiler principles, scanning and parsing algorithms

Date	Signature of course coordinator	Signature of seminar coordinator
12.05.2022	SMotogna.	SMotogna.

Date of approval

Signature of the head of department

Prof. dr. Laura Dioşan

24.05.2022

Diosan