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

2.Data about the subject

2.1 Subject name	***Data Structures and Algorithms***			Subject code	**12**
2.2 Course responsible / lecturer	Lect. dr. eng. Joldoş Marius - Marius.Joldos@cs.utcluj.ro Lect. dr. eng. Varga Robert - Robert.VARGA@cs.utcluj.ro				
2.3 Teachers in charge of seminars / Laboratory / project	Lect. dr. eng. Pocol Ciprian - Ciprian.Pocol@cs.utcluj.ro				
2.4 Year of study	I	2.5 Semester	1	2.6 Type of assessment (E - exam, C - colloquium, V – verification)	E
2.7 Subject category	Formative category: DF – fundamental, DS – speciality, DC – complementary				DF
	Optionality: DOB – mandatory, DOP – optional (alternative), DFA – optional (free choice)				DOB

3.Estimated total time

3.1 Number of hours per week	5	of which:	Course	3	Seminars	-	Laboratory	2	Project	-
3.2 Number of hours per semester	70	of which:	Course	42	Seminars	-	Laboratory	28	Project	-
3.3 Individual study:										
(a) Manual, lecture material and notes, bibliography										30
(b) Supplementary study in the library, online and in the field										20
(c) Preparation for seminars/laboratory works, homework, reports, portfolios, essays										20
(d) Tutoring										5
(e) Exams and tests										5
(f) Other activities:										-

3.4 Total hours of individual study (suma (3.3(a)...3.3(f)))	80
3.5 Total hours per semester (3.2+3.4)	150
3.6 Number of credit points	6

4.Pre-requisites (where appropriate)

4.1 Curriculum	Computer Programming course
4.2 Competence	Programming in C

5.Requirements (where appropriate)

5.1. For the course	
5.2. For the applications	

6.Specific competence

6.1 Professional competences	**C1 –** Operating with basic Mathematical, Engineering and Computer Science concepts • **C1.1 –** Recognizing and describing concepts that are specific to the fields of calculability, complexity, programming paradigms, and modeling computational and communication systems • **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 • **C1.4 –** Formal evaluation of the functional and non-functional characteristics of computing systems • **C1.5 –** Providing a theoretical background for the characteristics of the designed systems
6.2 Cross competences	N/A

7. Expected Learning Outcomes

7.1 Knowledge	Demonstrate advanced and critically informed knowledge by being able to: • Explain the fundamental concepts, properties, and limitations of core data structures, including arrays, linked lists, stacks, queues, trees, hash tables, and graphs. • Describe and critically explain major algorithm design strategies such as greedy methods, divide-and-conquer, backtracking, dynamic programming, and branch-and-bound. • Explain the principles of iterative and recursive problem solving, including their theoretical foundations and practical implications. • Explain concepts of time and space complexity and their role in evaluating algorithm performance and scalability.
7.2 Abilities and Skills	Demonstrate advanced cognitive and practical skills by being able to: • Select and justify appropriate data structures for modeling and solving non-trivial computational problems. • Design algorithms using suitable development techniques and adapt strategies to problem constraints. • Implement data structures and algorithms in the C programming language. • Analyze and compare algorithms with respect to efficiency, and resource usage.
7.3 Responsibilities and autonomy	• Work independently to design, implement, test, and debug algorithmic solutions. • Take responsibility for selecting suitable algorithmic and data structure choices under given constraints. • Apply learned concepts autonomously to new or unfamiliar algorithmic problems. • Reflect on solution quality and identify areas for further improvement or learning.

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

8.1 General objective	To provide students with a solid foundation in fundamental data structures and algorithms, and to develop their ability to design, implement, and analyze algorithms with respect to correctness, efficiency, and computational complexity.

8.2 Specific objectives	• **Analyze and compare** static and dynamic data structure implementations in terms of performance and memory trade-offs. • **Compare and evaluate** iterative and recursive solutions, and **determine** when recursion is an appropriate design choice. • **Determine** the time and space complexity of simple iterative algorithms and recursively defined algorithms. • **Design and implement** algorithms using fundamental design strategies, including greedy methods, divide-and-conquer, backtracking, dynamic programming, and branch-and-bound. • **Implement** fundamental data structures—including arrays, linked lists, stacks, queues, trees, hash tables, and graphs—using the C programming language. **Implement and evaluate** common sorting algorithms and analyze their performance characteristics • **Select** appropriate data structures for modeling and solving given computational problems.

9. Contents

9.1 Lectures	Hours	Teaching methods	Notes
Notions of Algorithmics (growth of functions, efficiency, programming model). Dynamic singly-linked lists	3	Lectures, demos and Discussions.	Video-projector Required.
Operations on dynamic lists. Doubly-linked and circular lists	3		
Trees: Definitions. Traversals. ADT Tree. Implementations. Binary Search Trees.	3		
Sets ADTs and Implementations. Dictionary ADT. Hash Tables. Mapping ADT. Priority Queue ADT.	3		
AVL trees. 2-3 Trees. B+ trees. Union-Find Set ADT.	3		
Graphs: Definitions. Representations. ADT's. Traversals for graphs and applications	3		
Graphs. Topological sort, strongly connected components, articulation points . Applications.	3		
Algorithm Design Techniques I. Backtracking.	3		
Algorithm Design Techniques II. Brute Force Algorithms. Greedy Algorithms.	3		
Algorithm Design Techniques III. Divide-and-Conquer.	3		
Algorithm Design Techniques IV. Dynamic Programming.	3		
Algorithm Design Techniques IV. Search Tree Strategies (branch and bound).	3		
Sorting Algorithms	3		
Review	3		
Bibliography: 1. Aho, Hopcroft, Ullman. Data Structures and Algorithms, Addison-Wesley, 427 pages, 1987. 2. Cormen, Leiserson, Rivest, Stein: Introduction to Algorithms, 4th edition. MIT Press / McGraw Hill, 1291 pages, 2022.			
9.2 Applications - Seminars / Laboratory / Project	Hours	Teaching methods	Notes
Review of C Programming. Laboratory requirements.	2	Tutoring, discussions, and assisted program development.	PCs equipped with MinGW C development kit and Code-blocks IDE.
Singly-linked Lists, Stacks and Queues.(Array-based and Dynamic Allocation Implementations)	2		
Arbitrary Trees. Binary Trees	2		
Binary Search Trees	2		
Hash Tables.	2		

Laboratory Test 1	2		
Graph Representations and Traversals (BFS) and applications	2		
Graph Representations and Traversals (DFS) and applications	2		
Algorithm Design I Backtracking and Branch and Bound	2		
Algorithm Design II. Divide & Conquer	2		
Algorithm Design III. Greedy	2		
Algorithm Design IV. Dynamic Programming and Heuristics.	2		
Laboratory Test 2	2		
Bibliography: 1. Moodle course Web Site available at https://moodle.cs.utcluj.ro/			

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

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

The contents of the course is in accordance with the ACM Computer Science Curricula recommendations. The contents was discussed with the Computer Science departments of the similar technical universities in Bucharest and Timișoara and was evaluated by CNEAA and Aracis

11. Evaluation

Activity type	Assessment criteria	Assessment methods	Weight in the final grade
Course	The understanding of the concepts taught and the ability to solve problems	Three in-class tests (T) + Final Written exam (W)	60% W + 10% T
Seminar	-	-	-
Laboratory	Quality of the assigned applications and in-class tests	Analysis and evaluation of the solved assignments and two in- class tests	30%
Project	-	-	-
Minimum standard of performance: : evaluation grade ≥ 5 Grade calculus: 40% laboratory + 60% exams and tests Conditions for participating in the final exam: Laboratory ≥ 5 Conditions for promotion: final written exam grade ≥ 5 and final written exam problems grade ≥ 5			

Date of filling in: 30.04.2026	**Responsible**	**Title, First name Last name**	**Signature**
	Course	Lect.dr.eng. Marius JOLDOȘ	
		Lect.dr.eng. Robert VARGA	
	Applications	Lect.dr.eng. Ciprian POCOL	

Date of approval in the department	Head of department, Prof.dr.eng. Rodica Potolea
Date of approval in the Faculty Council	Dean, Prof.dr.eng. Vlad Mureșan