ALGORITHMS AND PROGRAMMING LANGUAGES

ACADEMIC YEAR 2024-2025

Degree	**B**achelor of **S**cience in **C**omputer **S**cience			
Qualification	**C**omputer **S**cience			
Professor	PhD Gagik Ayvazyan			
Distribution of hours	**CM 30 h.**	**TP 30 h.**	**TPS 120 h.**	**ECTS 6**

EXPECTED LEARNING OUTCOMES OF THE COURSE		
A- Knowledge		A 1 Understand and apply formal methods in specification and verification of algorithms, methods of designing and analyzing algorithms and data structures.
B-Skills	**B1 -** Skills to apply professional knowledge	B 1.1 Demonstrate the ability to apply logic-based methods in program verification. B 1.2 Demonstrate the ability to calculate weakest precondition of program segment. B 1.3 Utilize assignment, consequence and composition rules of Hoare logic to program verification. B 1.4 Demonstrate the ability to use the conditional rule for proof of program correctness. B 1.5 Utilize loop invariant and loop rule of Hoare logic to program verification. B 1.6 Demonstrate the ability to apply the termination theorem for proving termination of loop segments of programs. B 1.7 Demonstrate the ability to select computer algorithms and relevant data structures for the design of software systems and their components. B 1.8 Use computer algorithms and data structures (graph algorithms, string matching algorithms, dynamic programming technique and hashing) in professional fields such as engineering, artificial intelligence, and data science.
	B2 - General (transversal) skills	B 2.1 Demonstrate the ability to analyze complex problems and formulate algorithmical solutions using appropriate algorithms and data structures. B 2.2 Adapt algorithms and data structures to different problem domains and evolving real-world scenarios. B 2.3 Take initiative in self-directed learning,

		seeking opportunities to deepen understanding and skills in logic-based methods, algorithms and data structures beyond the scope of the course. B 2.4 Manage time effectively to meet deadlines for assignments and exams while balancing other academic and personal commitments.

KNOWLEDGE / SKILLS ASSESSMENT & EVALUATION		
Ongoing evaluation tasks *(max 1/3 of grade for the total course)*	**Midterm exam**	**Final exam**
Assessment : **O**ral ☐ **W**ritten ☐	**A**ssessment : **O**ral ☐ **W**ritten ☐	**A**ssessment : **O**ral ☐ **W**ritten ☒
Duration : XXX h. **C**riteria :	**G**roup base: **Y**es ☐ **N**o ☐	**G**roup base: **Y**es ☐ **N**o ☒
Course project : **Y**es ☐ **N**o ☐ **P**resentation : **Y**es ☐ **N**o ☐	**D**uration : 1.5 h.	**D**uration : 1.5 h.
Tasks type & **W**eight : XXXXXX	**E**xam type : Written	**E**xam type : Summative Semestral Exam (written) / Assignment /Practical Assignment

TEACHING METHODS & TOOLS

Students will be guided to develop problem-solving skills through interactive activities that are closely tailored to the lesson at hand. Students will be guided to discover activities that promote self-learning and help students develop critical thinking skills and retain knowledge that leads to self-actualization.

Lecture
Explanation
Slideshow
Presentation
Demonstration
Video-presentation

Practical Work
Modeling
Exercises
Self-study
Instruction with demonstration

Extra-mural/individual work
Study of textbooks, sources
Individual work
Problem solving learning

KNOWLEDGE & SKILLS PREREQUISITS

Set, function, recurrence, relation, algorithm, logic, formal proofs, programming skills

COURSE DESCRIPTION /SYLLABUS / RESOURCES

Course objective: The objective of this course is to provide an approach to proving the correctness of computer programs using a

logic-based method. The course introduces Hoare's approach to program verification. The course also includes methods of designing computer algorithms and data structures. Topics covered in this course include the axioms and inference rules of Hoare logic, as well as graph algorithms, string matching algorithms, hashing and dynamic programming techniques.

TOPIC	HOURS	CORE RESOURCES [1]	ADDITIONAL RESOURCES
1. Program specification and verification. Hoare logic 1.1 Hoare's notation. Hoare's triple. 1.2 Formal specification by Hoar triple 1.3 Predicate transformer. Weakest precondition 1.4 Precondition strengthening 1.5 Postcondition weakening	1.5hCM	Almeida J.B, Frade M.J, Pinto J.S, Sousa S.M, Rigorous Software Development: An Introduction to Program Verification pp. 129-139 Krzysztof R. Apt, Frank S. de Boer, Ernst-Rudiger Olderog Verification of Sequential and Concurrent Programs pp.63-65	http://www.cs.uu.nl/docs/vakken/b1li/slides/14-slides.pdf
2. Hoare calculus. Inference rules of Hoare logic 2.1 Checking validity mechanically 2.2 Calculation of weakest precondition in case of assignment, sequence and selection 2.3 The assignment rule 2.4 The consequence rules 2.5 The composition rule 2.6 The If-Then Rule 2.7 The If-Then-Else Rule 2.8 The array assignment rule	3hCM	Almeida J.B, Frade M.J, Pinto J.S, Sousa S.M, Rigorous Software Development: An Introduction to Program Verification pp. 129-139 Krzysztof R. Apt, Frank S. de Boer, Ernst-Rudiger Olderog Verification of Sequential and Concurrent Programs pp.65-69	https://www.cs.cmu.edu/~aldrich/courses/654-sp07/slides/7-hoare.pdf
3.Invariants and variants 3.1 Loop Invariants 3.2 Loop Variants 3.3 The Loop rule in Hoare logic	3hCM	Almeida J.B, Frade M.J, Pinto J.S, Sousa S.M, Rigorous Software Development: An Introduction to Program Verification pp. 140-143 David Gries, The science of	https://www.cs.cmu.edu/~aldrich/courses/654-sp07/slides/7-hoare.pdf

3.4 Techniques for obtaining invariants		programming pp.138-144, 193-198	
4. Loop Termination. Derecursivation. 4.1 Partial correctness 4.2 Loop termination and total correctness 4.3 The state of a computation 4.4 The bound function for termination 4.5 The termination theorem 4.6 Recursion as a form of iteration 4.7 Tail recursion 4.8 Using iteration instead of recursion. Iterative implementation of tail recursion 4.9 Formal treatment of recursive function call	4.5hCM	Almeida J.B, Frade M.J, Pinto J.S, Sousa S.M, Rigorous Software Development: An Introduction to Program Verification pp. 151-155, 196-204 David Gries, The science of programming pp.109-112, 222-235	https://www.cs.cmu.edu/~aldrich/courses/654-sp07/slides/7-hoare.pdf https://www.pixelstech.net/article/1474689232-Traditional-recursion-vs-Tail-recursion
5. Hash tables.Hash functions 5.1 Direct-address table. 5.2 Hash tables. 5.3 Hash functions. 5.4 Collision resolution. 5.5 Hashing with chaining. 5.6 Open addressing.	3hCM	Cormen T, Leiserson Ch, Rivest R, Stein C, Introduction to Algorithms pp. 273-278, 282-288,293-297	https://www.cprogramming.com/algorithms-and-data-structures.html
6. Dynamic Programming 6.1 Dynamic programming method 6.2 Optimal substructure 6.3 Overlapping subproblems 6.4 Rod-cutting problem 6.5 Matrix-chain multiplication	6hCM	Cormen T, Leiserson Ch, Rivest R, Stein C, Introduction to Algorithms pp. 363-370, 373-380,382-390	https://www.cprogramming.com/algorithms-and-data-structures.html
7. Graph algorithms 7.1 Breadth first search algorithm 7.2 Depth first search	6hCM	Cormen T, Leiserson Ch, Rivest R, Stein C, Introduction to Algorithms pp. 363-370, 554-557, 563-566, 573-579, 612-616, 655-	https://www.cprogramming.com/algorithms-and-data-structures.html

algorithm 7.3 Floyd-Warshall algorithm 7.4 Bellman-Ford algorithm 7.5 Ford-Fulkerson algorithm 7.6 Topological sort		659, 676-687	
8. String matching 8.1 Naive string-matching algorithm 8.2 Rabin-Karp algorithm 8.3 Knuth-Morris-Pratt algorithm 8.4 Boyer-Moore algorithm	3hCM	Cormen T, Leiserson Ch, Rivest R, Stein C, Introduction to Algorithms pp. 960-966, 975-980, 985-993	https://www.cprogramming.com/algorithms-and-data-structures.html

STRUCTURE OF THE COURSE & ADDITIONAL INFORMATION REGARDING THE COURSE PREPARATION

CORE REFERENCES

1. Cormen T, Leiserson Ch, Rivest R, Stein C, "Introduction to Algorithms" 4th ed, the MIT Press, 2022.
2. Almeida J.B, Frade M.J, Pinto J.S, Sousa S.M, Rigorous Software Development: An Introduction to Program Verification, Springer-Verlag London Limited, 2011
3. Krzysztof R. Apt, Frank S. de Boer, Ernst-Rudiger Olderog Verification of Sequential and Concurrent Programs, 3rd ed, Springer, 2009
4. David Gries, The science of programming, Springer, New York Inc.

ADDITIONAL REFERENCES

1. Aaron R. Bradley, Zohar Manna, The Calculus of Computation: Decision Procedures with Applications to Verification, Springer, 2007
2. Wolfgang Ahrendt, Bernhard Beckert, Deductive Software Verification, Springer, 2016
3. Backhouse R, Program Construction—Calculating Implementations from Specifications, Wiley, New York, 2003
4. Louridas P. Algorithms, the MIT Press, 2020
5. Wengrow J., A common-sense guide to data structures and algorithms, The Pragmatic Programmers, 2020.

6. Morin P., Open data structures, AU Press. 2013

7. Steven S. Skiena "The Algorithm Design Manual", 2nd ed, Springer-Verlag London Limited, 2008

WEB RESOURCES

1. https://en.wikipedia.org/wiki/Hoare_logic
2. https://formal.kastel.kit.edu/beckert/teaching/Formale-Verifikation-SS09/01Intro.pdf
3. https://www.cs.cmu.edu/~aldrich/courses/654-sp07/slides/7-hoare.pdf
4. http://www.cs.uu.nl/docs/vakken/b1li/slides/14-slides.pdf
5. https://en.wikipedia.org/wiki/Loop_invariant
6. https://www.pixelstech.net/article/1474689232-Traditional-recursion-vs-Tail-recursion
7. https://www.cprogramming.com/algorithms-and-data-structures.html
8. www.youtube.com/watch?v=t-Mj4ji3tCw&list=PLA72M-qSGPm2WxSxXthNiYx2u4KBZlXCC
9. https://towardsdatascience.com/top-algorithms-and-data-structures-you-really-need-to-know-ab9a2a91c7b5