Óbuda University John von Neumann Faculty of Informatics	Institute of Software Engineering
Name and code: Comprehensive exam (NIXSS1EBNE)	**Credits:** 0
Computer Science BSc	*Daytime 2020/21 year II. semester*

Subject lecturers: Dr. László Csink, Dr. Sándor Szénási, Dr. Zoltán Imre Vámossy, Dr. Henriette Komoróczki-Steiner, László Somlyai				
Prerequisites: (with code)	Software design and Development I (NIXSF1EBNE), Software design and Development II (NIXSF2EBNE), Web programming and advanced development techniques (NIXWH1EBNE), Advanced development techniques (NSXHF1EBNE), Digital systems (NIXDR0EBNE), Electronics (NIEEL0EBNE)			
Weekly hours:	Lecture: 0	Seminar: 0	Lab. hours: 0	Consultation: 0
Way of assessment:	Examination			

Course description

Goal: -

Course description: **Topics of Engineering**

1. Design and analysis of combination networks: universal logic functions, basics of systematic system design.
2. Description of combination networks: logic functions, truth tables, schematics and Karnaugh maps.
3. Ideal and realistic components, properties of real components: cause of non-ideal behavior, propagation delay, hazards in combination networks
4. Definition of sequential networks, classification of sequential networks.
5. Development and analysis of sequential networks: Basic latches, use and behavior of flip-flops, development of network from gates and latches.
6. Analysis of sequential networks, state-tables, state functions, state-diagrams, next-state maps. Race situation and oscillation in sequential networks.
7. Basic schematics of important logic network families and their properties, such as RTL, DTL, TTL, CMOS. Basic register schematics and basics of their operation.
8. Static and dynamic properties of digital circuits, properties of rising/falling edges and propagation delays, transfer characteristic of basic gates, static and dynamic power consumption.

Topics of Software Design and Development

You are expected to have general knowledge of the topics, to present examples, to present pseudocodes of relevant algorithms, to analyze algorithms' efficiency, or occasionally provide C# code. There might be questions that involve several topics (e.g. compare the insertion sort, quicksort and heapsort algorithms).

1. The basics of algorithms: The concept of the algorithm, flow structures, tools for describing the algorithm (block diagram, box diagram, and pseudocode), efficiency, effectiveness, big O notation
2. Simple Basic Programs: BP Sequence, BP Decision, BP Selection, BP Linear Search, BP Counting, BP Maximum Search
3. Compound Basic Programs: BP Copy, BP Picking, BP Separation, BP Intersection, BP Union, BP Merge (union of sorted arrays)
4. Combining Basic Programs: Combining BP Copy with BP Maximum Search, Combining BP Counting with BP Linear Search, Combining BP Maximum Search with BP Picking, Combining BP Picking with BP Maximum Search, Combining BP, Picking with BP Copy
5. Sorting: Sorting with Simple Changes (SSC), Minimum Selection Sort (MSS), Bubble Sort (BS), Modified Bubble Sort, Insertion Sort (IS), Modified Insertion Sort (MIS), Shell Sort (SHS)
6. Searching: Linear Search in a sorted sequence, Binary Search, Application of Binary Search, applications of Binary Search: BP Decision, BP Selection, BP Picking and BP Counting for sorted sequences
7. Sets: Set as a data structure, creation of a set out of a sorted array; check whether an array is a set, membership, inclusion, subset, Union, intersection, subtraction, complement, symmetric difference
8. Recursion 1: Idea of recursion, general model, recursive function call, advantages and disadvantages. Program transformations: R $\rightarrow$ I transformation, I $\rightarrow$ R transformation, examples.
9. Recursion 2: Recursive conversion of a number to another base Recursive and iterative factorial, Recursive and iterative. Fibonacci algorithm, recursive binomial (bin, bin1, bin2), Hanoi towers, String reversal (reverse, reverse1), palindrome (palindrome, palindrome1, and palindrome2), power and power1
10. Advanced sorting 1: Recursive Merge Sort, Recursive Quicksort. Description, performance, randomized version.
11. Advanced sorting 2: Heap, min-heap and max-heap, heap algorithms, heapsort.
12. Advanced sorting 3: Distribution sort, counting sort, radix sort, bucket sort.
13. Dynamic Programming: Greedy algorithms. Divide-and-Conquer strategy, the idea of Dynamic Programming, the knapsack-problem. Longest Common Subsequences. Matrix Chain Multiplication.
14. Backtracking: The idea of backtracking, the 8-queens problem
15. Lists: Definitions, comparison with the array, list algorithms (traversal, search, insert, delete).
16. Sorted linked lists: Definitions, algorithms (traversal, search, insert, delete), sentinel nodes, special linked lists (doubly, multiply, circular) 17. Graphs 1: Directed, undirected, weighted graphs, graph as data structure, path, connectivity, Cycles, Components. Finding an acyclic path (the labyrinth problem: Theseus, Ariadne and Minotaur).
18. Graphs 2: Minimum-weight spanning trees (Kruskal, Prim).
19. Graphs 3: Maximum flow.
20. Binary Search Tree: Concept of a tree, definitions, binary tree, binary search tree, BST operations (lookup, traversals, insert, remove
21. B-trees: Definitions, advantages, disadvantages, insert a node, remove a node.
22. Hashing: The idea, direct addressing, various hashing tables, universal hashing, collision, solutions.
23. Object oriented programming: Inheritance, overriding, hiding, problems of multiple inheritance, polymorphism, non-virtual and virtual methods.
24. Interfaces: Implicit and explicit interfaces, sorting example
25. Event handling: Function pointers, events, delegates, examples.
26. Exception handling: Advantages, system exceptions, application exceptions, your own exceptions, exception rethrow, best practices, examples

Lecture schedule	
Education week	Topic

Midterm requirements

Midterm Test Scheduling

Education week	Topic

Midterm grade calculation methods

Method of replacement

Type of exam

When the current pandemic situation ceases
The exam has two parts:
First part (35 minutes): an entry test based on engineering and software design and development.
Second part (75 minutes, short break, again 75 minutes): full answers to questions relating the above listed topics, including the solution of problems as well as theoretical issues. If you achieve less than 50 % at the first part, you fail. If you pass at the first part, you can write the second part. You must achieve an overall 50 % at the second part, but at least 40 % at engineering and 40 % at software design and development as well. There is a break between the first and the second part.
In the case of an online exam:
The exam has two parts:
First part: a test based on software design and development via moodle.
Second part: test questions via moodle and full answers to questions relating engineering topics, including the solution of problems and theoretical issues. During this second part of the exam, the students should download the questions in a .doc file; they should enter their solutions into the file and then, saved as .pdf, they should upload it to the moodle.

There is a break between the first and the second part.

Exam grade calculation methods

The student must achieve an overall 50 % at the comprehensive exam, but at least 40 % at engineering and 40% at software design and development part.

0-50%	failed (1)
51-63%	satisfactory (2)
64-75%	average (3)
76-87%	good (4)
88-100%	excellent (5)

You may be exempted from the Comprehensive exam if you have at least grade 4 of each of the following subjects
- Software design and development I (NIXSF1EBNE)
- Software design and development II (NIXSF2EBNE)
- Digital systems (NIXDR0EBNE)
- Electronics (NIEEL0EBNE)

References

Obligatory:

Recommended:

Recommended Literature to the Software design and development part:
Cormen, Leiserson, Rivest, Stein: Introduction to Algorithms. Third edition.

Others: