Óbuda University John von Neumann Faculty of Informatics				
Name and code: Comprehensive exam (NIXSS1EBNE)			**Credits*: 0***	
Computer Science BSc			*2024/25 year Autumn semester*	
Subject lecturers: Dr. László Csink, Dr. Sándor Szénási, Dr. Zoltán Vámossy, Dr. Henriette Komoróczki-Steiner, László Somlyai, Dániel Kiss				
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

Topics of Engineering

- Design and analysis of combination networks: universal logic functions, basics of systematic system design.
- Description of combination networks: logic functions, truth tables, schematics and Karnaugh maps.
- Ideal and realistic components, properties of real components: cause of non-ideal behavior, propagation delay, hazards in combination networks
- Definition of sequential networks, classification of sequential networks.
- Development and analysis of sequential networks: Basic latches, use and behavior of flip-flops, development of network from gates and latches.
- Analysis of sequential networks, state-tables, state functions, state-diagrams, next-state maps. Race situation and oscillation in sequential networks.
- Basic schematics of important logic network families and their properties, such as RTL, DTL, TTL, CMOS. Basic register schematics and basics of their operation.
- 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).

- 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.
- Simple Basic Programs: BP Sequence, BP Decision, BP Selection, BP Linear Search, BP Counting, BP Maximum Search.
- Compound Basic Programs: BP Copy, BP Picking, BP Separation, BP Intersection, BP Union, BP Merge (union of sorted arrays).
- 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.
- 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).
- Searching: Linear Search in a sorted sequence, Binary Search, Application of Binary Search: BP Decision, BP Selection, BP Picking and BP Counting for sorted sequences.
- 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.
- Recursion 1: Idea of recursion, general model, recursive function call, advantages and disadvantages. Program transformations: R to I transformation, I to R transformation, examples.
- 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.
- Advanced sorting 1: Recursive Merge Sort, Recursive Quicksort. Description, performance, randomized version.
- Advanced sorting 2: Heap, min-heap and max-heap, heap algorithms, heapsort.
- Advanced sorting 3: Distribution sort, counting sort, radix sort, bucket sort.
- Dynamic Programming: Greedy algorithms. Divide-and-Conquer strategy, the idea of Dynamic Programming, the knapsack-problem. Longest Common Subsequences. Matrix Chain Multiplication.
- Backtracking: The idea of backtracking, the 8-Queens problem.
- Linked lists: Definitions, comparison with the array, list algorithms (traversal, search, insert, delete).
- Sorted linked lists: Definitions, algorithms (traversal, search, insert, delete), sentinel nodes, special linked lists (doubly, multiply, circular).
- 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).
- Graphs 2: Minimum-weight spanning trees (Kruskal, Prim). Maximum flow.

- Binary Search Tree: Concept of a tree, definitions, binary tree, binary search tree, BST operations (lookup, traversals, insert, remove).
- B-trees: Definitions, advantages, disadvantages, insert a node, remove a node.
- Hashing: The idea, direct addressing, various hashing tables, universal hashing, collision, solutions.
- Object oriented programming: classes, inheritance, overriding, hiding, problems of multiple inheritance, polymorphism, non-virtual and virtual methods.
- Interfaces: Implicit and explicit interfaces, sorting example.
- Event handling: Function pointers, events, delegates, examples.
- Exception handling: Advantages, system exceptions, application exceptions, custom exceptions, exception rethrow, examples.

Type of exam

The exam has two parts:

- First part (35 minutes): an entry test based on software design and development.
- Second part (75 minutes, short break, 75 minutes again): full answers to questions related to the topics listed above, 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 the first part, you can write the second part. You must achieve an overall 50% at the second part, and 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.

Calculation of the exam grade

Students have to pass the entry test, and achieve an overall 50% at the second part, and at least 40% at engineering and 40% at software design and development as well to pass the exam. It is not possible to take any previous results into account.

Results are converted to grades for both parts based on the followings.

0-49%: Fail (1)
50-61%: Pass (2)
62-73%: Satisfactory (3)
74-85%: Good (4)
86-100%: Excellent (5)

In case you passed the entry test, you may be exempted from writing the second part if the following criteria are met:

- You have at least a grade of 4.00 (without any rounding) in Software design and development I (NIXSF1EBNE) and Software design and development II (NIXSF2EBNE) on average, and
- You have at least a grade of 4.00 (without any rounding) in Digital systems (NIXDR0EBNE) and Electronics (NIEEL0EBNE) on average.

In this case, the final grade of the Comprehensive exam is:

- Good (4), if $4.00 \leq$ mean of the grades above < 4.50
- Excellent (5), if $4.50 \leq$ mean of the grades above ≤ 5.00

It is not possible to be exempted in case of any accreditation of subjects from previous studies.