List of topics for the

FINAL EXAM

of the Computer Science BSc course of ELTE

Last modified: July 7, 2020

1. **Limits and continuity of functions.** Limits and continuity of functions. Properties of continuous functions defined on a compact set: theorems of Weierstrass and Bolzano. Power series, theorem of Cauchy–Hadamard. Analytical functions.

2. **Differential and integral calculus.** Differentiation of functions. Partial derivatives, gradient, Jacobi matrix. Extreme values, discussion of functions. Riemannian integral, integration by parts, integration with substitution. Newton–Leibniz formula.

3. **Numerical methods.** Iterative methods of solving nonlinear equations: fixed point iterations, Newton's method. Interpolation: Lagrange and Newton forms. Least squares methods. Numerical integration: interpolation formulas, Newton–Cotes formulas, simple and composite quadrature formulas.

4. **Number theory, graphs.** Sets, relations, functions and operations. Complex numbers. Enumeration problems. Undirected and directed graphs, trees, Eulerian and Hamiltonian graphs, representation of graphs. Basic concepts in number theory: divisibility, congruence, primes. Polynomials: definition and operations, including division with remainder.

5. **Basics of probability and statistics.** Discrete and continuous random variables, law of large numbers, central limit theorem. Statistical estimates, classical statistical tests.

6. **Artificial intelligence.** Pathfinding problems and their modelling with directed graphs (the state space model). Heuristic pathfinding algorithms: local search (hill climbing, tabu search, simulated annealing), backtracking algorithms, heuristic graph search (A, A^*, A^C, B algorithm). Two-player games.

7. **Algorithmic patterns.** Concept of enumerator. Enumerators of well-known collections (interval, array, sequence, sequential input file). Algorithmic patterns on enumerators (summation, counting, maximum search, conditional maximum search, linear search, selection). Technique of analogy. Testing programs created with algorithmic patterns.

8. **Object oriented modeling.** Aspects of object oriented modeling: static model (class diagram, object diagram, package diagram, component diagram); dynamic model (state diagram, sequence diagram, use case diagram).

9. **Object oriented design.** Development phases of software systems, development methodologies. Architectural patterns (MV, MVC etc.). Role and classification of design patterns (creational, structural, behavioral); demonstrate two design patterns for each category.
10. **Fundamentals of programming languages.** Compilation and linking of code. Lexical, syntactic and semantic rules of a programming language. Evaluation of expressions. Statements, control stuctures. Representation of basic types. Compound types. Program structure, scope and visibility. Storing variables in memory, lifetime of variables. Parameter passing. Exceptions.
11. **Object oriented programming languages.** Classes and objects. Encapsulation, members, constructors. Information hiding. Overloading. Memory management, garbage collection. Inheritance, multiple inheritance. Subtyping. Static and dynamic type, type checking. Overriding, dynamic binding. Generics. Subtype and parametric polymorphism. Comparing and copying of objects.
12. **Formal languages and automata.** Generative grammars and Chomsky hierarchy. Normal forms of grammars. Regular expressions. Finite automata. Pushdown automata. Closure properties of language classes. Algorithmic problems in the regular and the context-free language classes.
13. **Theory of computation.** Turing machines and the Church–Turing thesis. Variants of Turing machines: multitape, nondeterministic, counting, offline. Recursive and recursively enumerable languages. Undecidable problems. Time and space complexity classes: P, NP, PSPACE. NP-complete problems.
14. **Basic algorithms.** Efficiency of algorithms, growth of functions. Comparison sorts, insertion sort, merge sort, quicksort, heap sort, lower bounds for the number of comparisons. Sorting in linear time (bucket, counting, and radix sorts). Data compression (naive, Huffman, LZW). String Matching (brute-force, quicksearch, KMP).
15. **Data structures and data types.** Arrays, stacks, queues, linked lists; binary trees, traversals, representations; binary heaps, priority queues; binary search trees, AVL trees, B+ trees; hash tables, hash functions, collision resolution by chaining, open addressing, probe sequences; representations of graphs.
16. **Advanced algorithms.** Elementary graph algorithms: breadth-first search, depth-first search and its applications. Minimum spanning trees, a general algorithm, Kruskal, Prim. Single-Source Shortest Paths: Queue-based Bellman-Ford, Dijkstra, DAG shortest paths. All-Pairs Shortest Paths: Floyd-Warshall algorithm. Transitive closure of a graph.
17. **Operating systems.** Processes and implementations, scheduling algorithms. Multitasking, parallelism, critical sections, mutual exclusion and implementations. Peter-

son algorithm. Semaphores, shared memory, message passing. Input-Output devices and scheduling, deadlocks. Memory management, virtual memory. Paging and segmentation. Paging algorithms (e.g. LRU- Least Recently Used, etc). Storage systems, redundant solutions, filesystems and their basic types, properties.

18. **Computer networks.** Layer models. Physical layer: baseband, broadband, digital encoding, modulation. Datalink layer: framing, error control (detection, correction), CRC, flow-control, dynamic channel allocation. Network layer: distance vector protocol, link-state routing protocol, BGP, path-vector protocol, IPv4 vs IPv6. Transport layer: UDP, TCP (connection management, congestion); Application layer: DNS, HTTP, DHCP, ARP.

19. **Concurrent programming.** Multithreaded programs. Scheduling, context switch. Race condition. Synchronization. Blocking operations. Use of memory (stack and heap) in threads. Programming language support for threads. Data types for synchronization and communication.

20. **Databases–design and query.** Relational model, entity–relationship model, transformation from ER to relational model. Relational algebra, SQL. Procedural extension of SQL (PL/SQL, PSM). Designing relational models, normal forms, decompositions.

21. **Databases—optimization and concurrency control.** Functions and components of database management systems. Index structures, executing queries, optimization strategies. Processing transactions, journaling and restoring, concurrency control.

22. **Functional programming.** Properties of functional programming languages: eager and lazy evaluation strategies, referential transparency. Static typing, currying. Offside rule. Basic types, conversions. Function definitions and the typing of functions. Pattern matching, guards, case distinction. Recursion. Local definitions. Higher-order functions, lambda functions. Function composition. ZF-expressions. Type classes, parametric and ad-hoc polimorphism. Type synonyms. Definition of algebraic data types.