List of topics for the

FINAL EXAM

of the Computer Science BSc course of ELTE

1. Limits and continuity of functions.

- Limits of Functions: Definition of the limit of a function and its special cases (finite limit at a finite point, infinite limit at a finite point, etc.). Sequential characterization of the limit. Relationship between limits and algebraic operations.
- Continuity of Functions: Definition of continuity; relationship between continuity and limits. Preservation of sign. Sequential characterization of continuity. Relationship between continuity and algebraic operations. Points of discontinuity and their classification.
- Properties of Continuous Functions on Compact Sets: Boundedness theorem, Weierstrass theorem, Bolzano–Darboux theorem. The bisection method for the numerical determination of zeros.
- Power Series: Definition of a power series and its radius of convergence. Cauchy–Hadamard theorem. Continuity of power series. The exponential, sine, and cosine functions.

2. Differential and integral calculus.

- Derivative of Real Functions: Definition of the derivative at a point. Relationship between continuity and differentiability. Concept of the tangent line. Equivalent formulation of differentiability via linear approximation. Differentiation rules.
- Applications of Differential Calculus to Function Analysis: Necessary and sufficient conditions for the existence of local extrema. Characterization of monotonicity using the derivative. Relationship between convexity and the derivative. Inflection points.
- Differential Calculus of Functions of Several Variables: Interpretation of partial derivatives and directional derivatives. Concept of the total derivative and its relationship to partial derivatives. Jacobian matrix. Gradient vector. Tangent plane to a surface.
- The Definite Integral: Definition of the Riemann integral. Integrability of continuous and piecewise continuous functions. Integrability of sums, products, and quotients of functions. Integrability of the absolute value of functions. Additivity with respect to intervals. The integral function and its properties (continuity and differentiability). Newton–Leibniz formula.
- The Indefinite Integral: Definition. Necessary and sufficient conditions for the existence of a primitive function. Rules for indefinite integration: linearity, the first and second substitution rules, and integration by parts.

3. Numerical methods

Solution of nonlinear equations: fixed-point iterations, definition of contraction, Banach fixed-point theorem on the interval [a,b]. Newton's method. Polynomial interpolation: problem formulation, Lagrange and Newton forms. General concept of the least squares method and its solution techniques for fitting polynomials. Numerical integration: concept and accuracy of interpolation-type quadrature formulas, definition of Newton–Cotes formulas, the midpoint, trapezoidal, and Simpson formulas and their accuracies, composite formulas and their accuracies.

4. Number theory, graphs.

Sets, relations, functions and operations. Complex numbers. Enumeration problems. Undirected graphs, trees, Eulerian and Hamil-tonian graphs. Basic concepts in number theory: divisibility, congruence, primes. Polynomials: definition and operations, including division with remainder.

5. Basics of probability and statistics

Standard discrete and continuous distributions: Poisson, binomial, hypergeometric, Pascal, uniform, exponential, normal. Expected value, standard deviation, covariance and correlation. Weak and strong law of large numbers for independent identically distributed random variables, central limit theorem. Statistical estimation: unbiasedness, consistency. Maximum likelihood method. Classical statistical tests for the expected value of the normal distribution: z-test, t-test. Chi-square test and its applications: goodness-of-fit test, homogeneity test, and independence test.

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 collec-tions (interval, array, sequence, sequential input file). Algorithmic patterns on enu-merators (summation, counting, maximum search, conditional maximum search, lin-ear search, selection). Technique of analogy. Testing programs created with algorith-mic patterns.

8. Object oriented modeling

Aspects of object oriented modeling: static model (class di-agram, object diagram, package diagram, component diagram); dynamic model (state diagram, sequence diagram, use case diagram).

9. Object oriented design.

- Development phases of large systems, development methods. Classical (Waterfall, Spiral, V-model) and Agile (Scrum, Kanban, XP) approaches.
- SOLID design principles. Architectural patterns (MV, MVVM, MVC, etc.)
- The role and classification of design patterns (creational, structural, behavioral). Singleton, Factory Method, Builder, Strategy, Template Method, Observer, Iterator, Command, Chain of Responsibility, Proxy, Decorator, Composite design patterns.
- Project management and DevOps. Version control systems, revision, development branches, merging and conflict resolution. Generational models of VCS systems. Feature branching models (GitFlow, etc.). Continuous integration and delivery. Clean Code principles.

10. Fundamentals of programming languages

- Rules of a programming language:
  - Lexical, syntactic and semantic rules. Pragmatics.
  - Compilation, linkage, interpretation.
- Expressions and their evaluation
  - Lexical elements: literals, identifiers, operators, parentheses
  - Syntax: arity, fixity
  - Semantics: Precedence, associativity (left-to-right, right-to-left), lazy and eager evaluation, side effect, order of evaluation, sequence point
- Types:
  - Basic types and their representation
  - Automatic and explicit type conversions
  - Arrays
  - Records, structures
  - Pointers
- Memory model, subprograms
  - Stack, heap, static (lifetime)
  - Parameter passing, execution stack, program structure, scope, visibility
  - The concept of exceptions. Raising and propagation of exceptions. Throwing and handling exceptions. Various kinds of exceptions.

11. Object oriented programming languages

Classes and objects, instantiation. Encapsulation, members, constructors. Instance and static members. Information hiding, visibility modifiers. Overloading. Memory management, garbage collection. Inheritance, multiple inheritance. Subtyping. Static and dynamic type, type checking. Overriding, dynamic binding. Interfaces. Generics. Subtype and parametric polymorphism. Program structure, packages, compilation units. Comparing and copying of objects. Testing.

12. Formal languages and automata

Generative grammars and Chomsky hierarchy. Nor-mal 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 re-cursively 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 com-parisons. 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 ad-dressing, 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 short-est paths. All-Pairs Shortest Paths: Floyd-Warshall algorithm. Transitive closure of a graph.

17. Operating systems.

- Classification of operating systems, the role of the CPU in their evolution.
- Implementation of processes, scheduling algorithms, avoidance of starvation.
- Concurrency, threads, processes, similarities and differences between threads and processes, inter-process communication.
- Critical sections, implementation of mutual exclusion. Peterson's algorithm. Semaphores, shared memory, message passing, the producer–consumer problem.
- Scheduling options for input and output devices, deadlocks. Difference between starvation and deadlock.
- Memory management, the concept of virtual memory management. Paging and segmentation. Multi-level page tables, the role of the TLB. Page replacement algorithms (e.g., LRU).
- Storages, redundancy, file systems, their basic types, services, and characteristics.

18. Computer networks.

- Layer models: ISO/OSI and TCP/IP.
- Physical layer: baseband, broadband, digital encoding schemes (Manchester, NRZI, 4/5 encoding), modulation (AM, FM, PM, QPSK, QAM-16).
- Data link layer: framing techniques, error control (detection and correction), CRC, flow control, dynamic channel allocation (ALOHA, CSMA, CSMA/CD).
- Network layer: distance-vector protocol, link-state protocol, BGP, path-vector protocol, IP fragmentation, IPv4 vs. IPv6.
- Transport layer: UDP, TCP, three-way handshake, connection management, TCP states, flow control, congestion control, slow start, congestion avoidance, Reno (fast retransmit and fast recovery), ECN marking, DCTCP (Data Center TCP)
- Application layer: DNS (name hierarchy, what existed before DNS, zone files, resolution, records, local name server), HTTP (non-persistent, persistent modes, cookies).

19. Concurrent programming.

Multithreaded programs. Scheduling, context switch. Race condition. Synchronization, explicit locks, reader-writer synchronization. Blocking operations. Use of memory (stack and heap) in threads. Programming language support for threads, monitor. Data types for synchronization and communication. Executors. Producer-consumer problem and its implementation. Deadlock.

20. Databases–design and query.

- Relational model, Relational algebra: set and multiset semantics, relational algebra operations, joins, grouping, aggregation, outer join
- Entity–relationship model: entity set, relationship, multiplicity, multiway relationships, roles, subclasses, isa relationships, weak entity sets, constraints, key, referential integrity transformation from E/R to relational model
- SQL: Data types, NULL values, 3-valued logic, WHERE, HAVING, set operators, subqueries, DML statements, CREATE TABLE, constraints, Triggers, Views
- Procedural extension of SQL (PL/SQL, PSM): structure of a PL/SQL program, datatypes, stored procedures and functions, cursors, exception handling,
- Designing relational models: functional dependencies (FD), key, superkey, computing the closure of attributes, inference rules, Armstrong's axioms, projecting FD-s, redundancy, anomalies
- Normal forms, decompositions: Boyce-Codd Normal Form, decompositions, decomposition into BCNF, Lossless-join property, Chase-test, Preservation of dependencies, 3-rd Normal Form, basis, minimal basis

21. Databases—optimization and concurrency control.

- Data storage: secondary storage devices, operation of the buffer manager, page replacement algorithms, structure of blocks, files, and records, sorted and unsorted file organization.
- Indexes: static and dynamic hash tables, B+ trees, bitmap indexes.
- Physical operators: cost of operations, output size estimation, types and costs of joins.
- Optimization: relational algebra equivalence rules, rule-based optimization, optimization of multi-table queries.
- Logging: UNDO, REDO, and UNDO/REDO logging, recovery, types of failures.
- Concurrency control: ACID properties, schedules, precedence graph.
- Locks: different lock modes, wait-for graph, deadlock, operation of the lock scheduler, hierarchical locks, intention-locking protocol, tree protocol.

22. Functional programming.

- Characteristics of functional programming languages: characterization of lazy evaluation strategy and comparison with eager evaluation strategy, evaluation (rewriting steps, reductions), referential transparency and pure functions (side-effect free functions), static typing (Haskell, Clean), type inference, polymorphism (parametric and ad-hoc), function as parameter and value,
- Defining functions: type signature, pattern matching, branching (guards), recursion (simple and tail recursion), local definitions (where), structuring by indentation (layout/offside rule), concept of partial functions, list comprehensions,
- Types and type classes: basic types (numeric types, characters, boolean type), complex types (list, tuple), conversions, type synonyms (type), defining algebraic data types (data), type classes and their instantiation (ad-hoc polymorphism), more complex predefined algebraic data types (Maybe, Either), function type (->),
- Definition and significance of higher-order functions: function as parameter and value, function type (->), anonymous functions (lambda expressions), operator sections, common higher-order functions: map, filter, composition (.).