State Final Examination Outline for the Bachelor's Degree in Computer Science

A. Computer and Information Systems

Programming (FPR, UPR, OOP, SJ, C#/JAVA I a II, UTI, ZDS)

- Programming paradigms (declarative and imperative, functional, structured, and object-oriented paradigms, specific features and principles of each paradigm).
- Data types (static and dynamic allocation, dynamic typing, memory lifecycle, simple, structured, abstract and generic types, collections, files, streams).
- Programming techniques (control structures, functions, their parameters and return values, recursion, polymorphism, collections, exceptions and their structured handling, parallelism and threads).
- Application development (C#/Java, type system, virtual machine and compilation, user interface design, delegates and events, database access, working with textual data, asynchronous programming, serialization, network communication).
- Representing numbers on the computer (number systems and conversions, integers, fixed point numbers, floating point numbers expressed in binary, decimal, and hexadecimal bases, arithmetic with numbers in different representations, character encoding).

Sample question: As a programmer, you are asked to solve the problem of transferring data between two applications. What forms of data transfer/communication between applications are you familiar with, on what basis would you select/choose the most appropriate form, and how would you implement it in your chosen programming language?

Software engineering (SWI, VIS)

- Software development life cycle (typical phases, their contents, basic development models).
- UML (diagram types, static system view, dynamic system view, mapping to source code, use in development).
- Information systems (life cycle, information systems design, and architecture).
- Design patterns and practices for developing information systems (GoF patterns, domain logic, data sources, object-relational behaviors and structures, domain-specific languages).

Sample question: Explain when it is appropriate to use the composite pattern, use a UML diagram to represent it schematically and project this diagram into source code in your chosen programming language.

Database systems (DS I, DS II)

- Data models (relational data model: definition, normal forms, functional dependencies, object-relational data model: basic features)
- Query languages (SQL: data definition language, data manipulation language, SELECT - joins, subqueries, aggregation functions; procedural SQL extensions: PL/SQL, T/SQL, general features: stored procedures, cursors, triggers).

- Information system analysis (conceptual and data model, state analysis, functional analysis - mini-specifications, form design).
- Transactions (definition, ACID, transaction serialization, recovery, concurrency control, transaction isolation level).
- Database query execution (physical database design, query execution, logical and physical operations).
- Data layer design and implementation (object-relational mapping, DTO, DAO, efficient data layer implementation).

Sample question: What SQL query can return unexpected results in a transaction when we use the Read Committed isolation level? Also, describe the session you are working on and write a specific query about that session.

Computer architectures and operating systems (APPS, OSY)

- Computer architectures (basic computer architectures, their description, advantages and disadvantages, principle of computer operation, addressing methods, hierarchical memory arrangement).
- RISC processors (basic design features, methods of accelerating processors, instruction chaining, jump prediction).
- GPUs, CUDA (reasons for using GPUs, description of CUDA technology, calculation methods, basic programming rules, calculation methods, memory handling).
- Processes (process creation, process implementation in the OS, process table, process states, handling of hardware interrupts, multitasking, pseudo-parallelism, scheduling algorithms).
- Inter-process communication (concurrency, critical section, mutual exclusion, basic ways of implementing mutual exclusion, semaphore, monitor, message queue, shared memory).
- Memory management (principle of virtual memory, segmentation, and its advantages, free memory management/proving).

Sample question: Explain the principle of virtual memory and what segmentation is used for. How do these technologies improve on or add to von Neumann's basic concept of how a computer works?

Computer networks and communication technologies (POS, PB)

- Routing and switching in computer networks (LAN topology; active elements and their functions - switch and its function, router; routing protocols - basic description, classes of routing protocols, hierarchical routing).
- TCP/IP protocol family (TCP/IP model and its relation to the ISO-OSI reference model; network protocols - IPv4 vs. IPv6; transport layer protocols and the principle of operation of a reliable TCP transmission channel).
- Internet services and their protocols (electronic mail; HTTP; DNS - record types, domain tree, zones, DNS security; SSH).
- Cryptography basics (block and stream ciphers including their modes; symmetric and asymmetric cryptography and their use; hash functions; public key infrastructure - basic pillars; certification authority; public key certificate).
- Building secure applications (general principles; common vulnerabilities, their categorization and description; database and web application security and typical attacks against them).

- Computer network security (attacks on infrastructure, servers, and application protocols and their detection; packet filtering; stateful firewall; virtual private networks).

Sample question: Why is HTTPS the preferred way to communicate with web servers today? What type of attack is it intended to prevent, what security function is it primarily intended to provide, and what is the importance of a Certificate Authority when communicating with HTTPS servers?

Computer graphics (ZPG, URO)

- Methods and tools for implementing graphical user interfaces (human cognitive abilities, mental models, basic design rules, color spaces, color selection, and text rendering).
- The standard display chain (implementation of chain steps, modeling, and display transformations, Phong's lighting model, visibility solutions, body identification, brief characteristics of the OpenGL standard and GLSL).
- Geometric modeling (affine and projective spaces, description of solids and ways of representing them, basic curves used in computer graphics, their expression, properties, and use, Ferguson cube, Bézier curve).

Sample question: Describe the options for representing solids and how to render them using a standard display string in the context of the OpenGL graphical interface.

B. Theoretical Foundations of Computer Science

Algoritmy (ALG I, ALG II, ZSU, UTI)

- Algorithmic problem-solving strategies (brute force, complete search, reduce and solve, divide and conquer, transform and solve, confounding memory and time complexity, dynamic programming, hungry algorithms).
- Complexity of algorithms (asymptotic notation, correctness of algorithms and their analysis).
- Basic data structures (array, list, stack, queue, graph, search tree, hash table, heap).
- Exploratory data analysis (data and their properties - numerical, categorical and other attributes, statistical properties of data, relationships between attributes).
- Machine learning methods (clustering, classification, evaluation of algorithms).

Sample question: Basic idea of the divide and conquer strategy. A practical application example is QuickSort and its time complexity. Comparison with the brute force strategy, BubbleSort.

Theoretical Computer Science (ULM, ZDS, UTI, PJP)

- Logic (propositional and predicate logic, syntax and semantics of propositional and predicate formulae, semantic and syntactic proofs, equivalent modifications, resolution methods).
- Boolean algebra (Boolean functions, minimization, connection with combinatorial circuits).
- Sets, relations and functions (operations on sets, types and properties of relations, types of relations, equivalence and ordering, properties of functions, inductive definitions and proofs).
- Theory of formal languages and automata (formal languages, operations on languages, formal means of describing languages - automata, grammars, regular expressions, finite, deterministic and non-deterministic automata, context-free grammars, stack automata, Chomsky hierarchies).
- Computability and complexity (computational models - RAM machines, Turing machines, computational complexity of algorithms, algorithmically undecidable problems, complexity classes, PTIME, NPTIME and PSPACE classes, conversions between problems, NP-complete and PSPACE-complete problems).
- Compilers (basic functions and types of compilers, stages of translation, formal resources used in compiler design, special subclasses of context-free grammars such as LL1 and regular expressions, techniques used in implementation such as recursive descent, compiler generators).

Sample question: Context-free grammars and their relation to stack automata The use of context-free grammars and stack automata in compiler design.

Mathematics for Computer Science (MA I, MA II, LA, DIM)

- Discrete mathematics (sequences, recurrent equations, combinatorial selections, discrete probability, congruence solutions).
- Graph theory (graphs, distance in a graph, measures of connectedness, trees and skeletons, graph coloring and applications, flows in networks).

- Linear algebra (matrices and matrix operations, inverse matrices, modification and solution of systems of linear equations, vector spaces and bases of vector spaces, linear mappings, spectral theory).
- Differential and integral calculus of functions of one variable (sequences, limits and continuity, derivatives, extremes, indefinite and definite integrals).
- Differential and integral calculus of multi-variable functions (partial derivatives, gradients, extremes, double and triple integrals).

Sample question: As an analyst, you are asked to propose the optimal allocation of tasks to each team. Which graph will you use to model the problem? What aspects of the problem are characterized by edge, edge coloring, and vertex degree? How will you use flows in networks in your solution?

Vvalid for the academic year 2024/2025

19.11.2024