TECHNISCHE UNIVERSITÄT DARMSTADT

Functional and Object-oriented Programming Concepts:

- Basic concepts of programming languages
- Foundations of functional programming languages
- Foundations of object-oriented programming languages
- Design and implementation of small software systems
- Basic type systems
- Fundamental data structures and algorithms and their complexity
- Recursion
- Simple I/O
- Basics of testing
- Documenting source code

Algorithms and Data Structures:

- data structures: array, list, binary search tree, b-tree, graph representation, hash table, heaps
- algorithms: sorting algorithms, string matching, graph traversal, insertion, search, and deletion on particular data structures, shortest path search, minimal spanning trees
- asymptotic complexity
- NP completeness
- algorithmic strategies: Divide-and-Conquer, dynamic programming, brute-force, greedy, backtracking, meta heuristics

Operating Systems:

- Introduction to Operating Systems (OS) - Role, purpose and design issues
- Processes and Threads - OS structures, process control, abstractions, kernel/user modes and operations, context switching, interrupts
- Inter-Process Communication - Message passing IPC, RPC, layers, interfaces, hierarchies
- Coordination: Deadlocks - Process coordination, critical sections, deadlock characterization, deadlock detection and recovery, deadlock avoidance
- Scheduling/Resource Management - Task ordering, preemptive and non-preemptive scheduling, schedulers and policies, OS implementations
- Concurrency: Races, Mutual Exclusions - Critical sections, races, spin locks, synchronization
- Programming Abstractions: Semaphores - Semaphores, Monitors
- Memory Management - Storage structures, management/replacements approaches, virtual memory, paging, caching, segmentation
- I/O - Device management, drivers, segmentation, interrupt handling, DMA
- File systems - File systems requirements, design and implementation, file structures, directories, naming, partitions, virtual file systems
- Fault Tolerance/Resilience - Fault types, fault handling approaches, reliable message delivery, OS reliability and availability, security issues
- Embedded/RT OS - Memory/disk/performance management, recovery, fault-tolerances, real-time aspects
- Distributed OS - Distributed computation and communication abstractions, synchronization, coordination, consistency
- Virtual Machines - Purpose and types of virtualization, virtual file systems, Hypervisors

Introduction to Compiler Construction:

- Structure of compilers
- Context-free grammars for the description of language syntax
- Lexing and parsing techniques
- Intermediate representations
- Semantic analysis
- Run-time organisation
- Code generation
- Software tools for compiler constructions
- Implementation techniques for compilers

Automata, Formal Languages and Decidability:

- introduction: transition systems, words, languages ● basic mathematical methods and proof patterns
- finite automata and regular languages, determinism and nondeterminism, closure properties and automata constructions, Kleene Theorem, Myhill-Nerode Theorem, pumping lemma
- grammars and the Chomsky hierarchy
- context-free languages, pumping lemma, CYK algorithm
- models of computation: PDA and Turing machines
- decidability and recursive enumerability in the Chomsky hierarchy

Computer Networks and Distributed Systems:

- Foundations: Service, protocols, connection, layer model
- protocol mechanisms for media access, routing, broad-/multicast
- Multimedia Data Handling
- Aspects of continuous data streams and their processing
- Quality of service: definition and mechanisms
- Multimedia - Synchronization: Basics
- Compression procedures

Information Management:

- Information Management Concepts:
  Information systems concepts
  Information storage/retrieval, searching, browsing, navigational vs. declarative access
- Quality issues: consistency, scalability, availability, reliability
- Data Modeling:
  Conceptual data models (ER/UML structure diagr.)
  Conceptual design
  Operational models (relational model)
  Mapping from conceptual to operational model
- Relational Model:
  Operators
  Relational algebra
  Relational calculus

Implications on query languages derived from RA and RC
Design theory, normalization
● Query Languages
SQL (in detail)
QBE, Xpath, rdf (high level)
● Storage media:
RAID, SSDs
Buffering, caching
● Implementation of relational operators:
Implementation algorithms
Cost functions
● Query optimization:
Heuristic query optimization
Cost based query optimization
● Transaction processing (concurrency control and recovery):
Flat transactions
● Concurrency control, correctness criteria: serializability, recoverability, ACA, strictness
Isolation levels
Lock-based schedulers, 2PL
Multiversion concurrency control
Optimistic schedulers
Logging
Checkpointing
Recovery/restart
● New trends in data management
Main memory databases
Column stores
NoSQL

Software Engineering:

● Software Project Management
● Software Process Models
● Requirements Engineering
● Software Development Tools
● Software Quality; in particular:
● Test Processes (automated testing, test coverage metrics, debugging)
● Software Metrics
● Object-oriented Analysis and Design
● Modeling using UML
● Software Design Patterns