هيئة تقويم التعليم والتدريب
Education & Training Evaluation Commission
اعتماد
NCAAA

Academic Standards for Computer Science Programs

2025

Version 2.0

Introduction

This document aims to contribute to the establishment of minimum requirements for bachelor's degree programs in Computer Science to ensure the academic quality of the programs. It seeks to ensure that graduates are highly qualified, possessing the knowledge, skills, and values required by the labor market, in accordance with national trends and in line with best practices required for the academic and professional pursuit of the specialty.

The Education and Training Evaluation Commission (ETEC), according to its mandate by virtue of the Council of Ministers decision No. 108, dated 14/2/1440 AH, has developed specialized academic standards for Computer Science programs. The remit of the Commission includes building systems for evaluating, approving, and accrediting programs in education and training, including institutional and programmatic, encompassing rules, standards, frameworks, and indicators, and building and developing high-quality national academic programs.

Terms

Academic Standards: Established benchmarks for the quality and level of student achievement in an academic program, outlining the expected knowledge, skills, and values upon completion and ensuring consistency, fairness, and credibility in learning and assessment as a foundation for quality assurance in higher education.

Education and Training Evaluation Commission (ETEC): Independent body aimed at evaluating, assessing, and accrediting qualifications in education and training in both public and private sectors, raising the quality and efficiency of those qualifications and ensuring they contribute to the national economy and development.

Essential Knowledge Units (EKUs): Knowledge Units necessary for future learning in a given discipline.

General Knowledge Units (GKUs): Knowledge Units that should be introduced to students majoring in a discipline.

Key Learning Outcomes (KLOs): The minimum required Learning Outcomes in the discipline that students are expected to obtain.

Knowledge Units (KUs): Mandatory multiple related topics that must be included in an institution's degree program.

Learning Outcomes (LOs): Description of what a learner is expected to know, understand, and be able to achieve, which is represented in his/her behavior and ability at the end of a specific educational program.

National Qualifications Framework (NQF): A comprehensive and uniform structure for building, organizing, and categorizing qualifications into levels based on Learning Outcomes.

Specific Knowledge Units (SKUs): Knowledge Units derived from a General Knowledge Unit.

Specific Learning Outcomes (SLOs): Learning Outcomes for a Specific Knowledge Unit.

Goals

The purpose of this work is to develop the minimum Knowledge Units (KUs) in Computer Science, specifying a set of Specific Learning Outcomes (SLOs) that students are expected to acquire after completing their studies. This document lays the foundations for the design of the academic program and the study plans and the alignment of the program with the demands of the labor market. This document also lays out the curriculum design, to facilitate selecting the appropriate teaching strategies and methods and tools for evaluation of the students.

Methodology

This document describes the minimum Essential, General, and Specific Knowledge Units (EKUs, GKUs, and SKUs) in Computer Science. The Learning Outcomes (LOs) of each KU set the threshold for what the students are expected to learn and be able to achieve after successfully completing that KU. Educational institutions should take into account the depth and breadth of these KUs so that the LOs integrate communication skills and values into the curriculum. Institutions can also offer additional KUs that are consistent with their objectives. It should also be noted that the KU is not necessarily an independent course; one or more courses can cover a single KU. Likewise, one course could also cover one or more KUs entirely or partially.

The methodology consists of the following phases:

1. Survey and benchmarking

- Benchmarking with international learned societies and professional bodies.
- Benchmarking with top-rated international and local universities.
- Identifying national labor market requirements.
- Incorporating input from specialists and experts from different sectoral groups.

2. Preparation of the contents of the specific standards document

- Identifying program Key Learning Outcomes (KLOs).
- Defining the general characteristics of the curriculum.
- Developing the GKUs and SKUs for each GKU.
- Formulating SLOs for each SKU.
- Determining the minimum topics required for each SKU.
- Describing the methodology for aligning academic content with the National Qualification Framework (NQF).

The KUs are derived from analyzing several high-ranked QS universities and international regulatory bodies and professional associations. The total credit hours outlined in this document account for about 60% of the total credit hours typically required for a Computer Science program.

The ETEC has developed this document in cooperation and coordination with different groups in the field of Computer Science, such as the such as the Ministry of Communications and Information Technology (MCIT), the Saudi Data and Artificial Intelligence Authority (SDAIA), the National Cybersecurity Authority (NCA), the Saudi Information Technology Company (SITE), and the Advanced Technology and Cybersecurity Company (SIRAR by STC), and with the involvement of various stakeholders from government bodies, the private sector, and academia.

Scope and Uses

This document is primarily intended to guide the development of curricula for a bachelor's degree in Computer Science for public and private higher education institutions in Saudi Arabia. It defines the minimum required knowledge, skills, and values that must be incorporated into program design to ensure graduates meet both academic and professional expectations. It is also used to support the creation of standardized tests and inform accreditation processes, ensuring consistency and quality across programs.

Key Learning Outcomes

Key Learning Outcomes[1] describe the essential knowledge, skills, and values that graduates of the Computer Science undergraduate program will be able to demonstrate once they complete the program.

On successful completion of a bachelor's degree in Computer Science, graduates should be able to:

KLO 1: Specialist Knowledge for Solving Computing Problems: Apply (a) knowledge of computing fundamentals, (b) knowledge of a computing specialization, (c) mathematics, science, and domain knowledge appropriate for the computing specialization to the abstraction and conceptualization of computing models for defined problems and requirements; and demonstrate the in-depth knowledge and skills for a professional role in a recognized computing specialization.

KLO 2: Problem Analysis: Identify, formulate, research literature, analyze complex computing problems and conduct investigation through: (a) experiments, (b) analysis and interpretation of data, as well as (c) synthesis of information to provide valid and substantiated conclusions with holistic considerations.

KLO 3: Design/ Development of Solutions: Design solutions with own perspectives for complex computing problems, and design systems, components, or processes that meet identified needs with appropriate consideration for public health and safety, information privacy and security, cyber security, life cycle cost analysis, net zero carbon, resource efficiency, culture, society, and environment.

[1] The key learning outcomes are defined according to the Seoul Accord Graduate Attributes (2025).

KLO 4: Modern Tool Usage: Create, select, adapt and apply appropriate techniques, resources, and modern computing tools, including but not limited to simulation, prediction and modelling, with recognition of their limitations, to complex computing activities and problems.

KLO 5: Individual and Team Work: Participate effectively as an individual and as a member or leader in diverse and inclusive teams and in multi-disciplinary, face-to-face, remote and distributed settings.

KLO 6: Project Management and Finance: Apply knowledge of IT management principles to the work as a member or leader of a team with consideration of market needs and budget, to manage projects under multidisciplinary environments.

KLO 7: Communication: Communicate effectively on complex computing activities with the computing community and with society at large, such as being able to comprehend and write effective reports, design documentation and make effective presentations, taking into account of cultural, language, and learning differences.

KLO 8: Computing Professionalism and Society: Analyze and assess the impacts on the sustainable development of professional computing practices and solutions related to the society, economy, information privacy and security, cyber security, health and safety, legal frameworks, corporate governance, entrepreneurship and the environment when solving complex computing problems.

KLO 9: Ethics: Apply ethical principles, adhere to professional ethics and norms of professional computing practice, accept responsibilities, incorporate diversity and inclusion, and comply with relevant local and international laws.

KLO 10: Life-long Learning: Recognize the need and have the ability for acquiring independent and life-long learning as a computing professional, adapt to new and emerging computing technologies and exercise critical thinking in the broadest context of technological advancement.

Curriculum General Criteria

Based on the benchmarking study of leading universities and the KUs and skills in the Computer Science programs, the KUs have been grouped into categories. The curriculum should consist of the following:

1. Essential Knowledge Units (EKUs) (minimum of 30 credit hours): Mathematics, Statistics, Basic Sciences, Project Management, and Finance appropriate to the discipline.
2. Computing Knowledge Units (minimum of 40 credit hours):
   - 2.1. General Knowledge Units (GKUs): mandatory Computer Science knowledge.
   - 2.2. Specific Knowledge Units (SKUs): mandatory Computer Science knowledge derived from GKUs.
3. Professional practice and experiential learning:

3.1. Coverage of ethical aspects of professional practice.

3.2. Minimum of three semester credit hours (or equivalent) of culminating computer science experience.

3.3. Field training experience of a minimum duration of eight weeks.

Each group consists of different subgroups essential in any typical Computer Science program. These subgroups are identified in the next section.

Knowledge Units

Essential Knowledge Units (EKU)

The Essential Knowledge Units should consist of a minimum of 30 credit hours in Mathematics, Statistics, Basic Sciences, and Project Management and Finance. This should include 23 credit hours of Mathematics, Physics, and Chemistry as shown in Table 1. Additional credit hours may be allocated to other Basic Sciences to address specific disciplinary depth requirements. Standardized tests should not include these EKUs.

Table 1: Essential Knowledge Unit for Computer Science programs

#	EKU	Description	Minimum Requirements (credit hours)
1	Mathematics and Statistics	Covers discrete mathematics, calculus, linear algebra, and probability to form the foundation of computing. This EKU equips students with the logical structures and continuous analytical tools necessary for algorithmic analysis, optimization, and statistical inference	11
2	Physics	Provides fundamental knowledge of physical laws, emphasizing mechanics and electromagnetism to understand hardware architecture and physical system constraints. This EKU covers electric circuits and magnetic fields.	6
3	General Chemistry	Provides the scientific basis for understanding material properties, energy conversion, and material constraints of computing hardware, including atomic structure, chemical bonding, and electrochemistry.	4

4	Project Management and Finance	Covers essential professional skills in project management principles, financial planning, budgeting, and risk assessment necessary for leading and justifying complex computing and technology initiatives.	2

Program Core Knowledge Units

The following table (Table 2) provides an overall view of the curriculum distribution of KUs: Essential, General, Specific and others. The table provides general recommendations on the acceptable range of topic distribution for each KU. The weighting of the credits is calculated based on a minimum of 40 GKU credits, as shown in Table 2.

Table 2: General and Specific Knowledge Units of a Computer Science program.

GKU	Weight	SKU	Weight
1. Algorithms, Computability, and Complexity	16%	1.1. Basic analysis and algorithmic methods	8%
		1.2. Data structures, automata, and computability	8%
2. Architecture and Organization	9%	2.1. Digital logic and machine-level data representation	3.5%
		2.2. Machine organization at the assembly level	3.5%
		2.3. Architecture of memory systems, interfacing, and communication	2%
3. Discrete Structures	17%	3.1. Introductory logic, sets, relations, and functions	7%
		3.2. Proof techniques and basics of counting	5%
		3.3. Graphs, trees, and discrete probability	5%
4. Information Management	8%	4.1. Database systems and data modeling	8%
5. Networking	7%	5.1. Networking fundamentals and networked applications	7%
6. Operating Systems	9%	6.1. Principles of operating systems	5%
		6.2. Concurrency, scheduling, memory management, and OS protection	4%
7. Programming Languages	18%	7.1. Object-oriented programming and functional programming	12%
		7.2. Program representation, language translation, and execution	6%

GKU	Weight	SKU	Weight
8. Software Engineering	16%	8.1. Software processes, software management, and requirements engineering	8%
		8.2. Software design, construction, verification, and validation	8%

Appendix A: Alignment of Key Learning Outcomes of a Computer Science Program within the National Qualifications Framework

Table A1 shows how each KLO aligns with specific areas within the National Qualifications Framework (NQF).

Table A1: Alignment of the Key Learning Outcomes for a Computer Science program with the NQF.

Computer Science Key Learning Outcomes	NQF Learning Areas		
	Knowledge and Understanding	Skills	Values, Autonomy, and Responsibility
1.	✓		
2.		✓	
3.		✓	
4.		✓	
5.			✓
6.			✓
7.		✓	
8.			✓
9.			✓
10.	✓		✓

Appendix B: Learning Outcomes and Topics for Knowledge Units

Essential Knowledge Unit (EKU.1): Mathematics and Statistics

Description	Mathematics and Statistics form the rigorous foundation for Computer Science. This EKU encompasses discrete mathematics for analyzing logical structures and algorithmic complexity, alongside continuous analytical tools such as calculus and linear algebra. It integrates probabilistic reasoning and statistical inference necessary for modeling complex systems, processing data, quantifying uncertainty, and deriving robust computing solutions.
Topics	**The following topics must be included in this EKU:** 1. Discrete Mathematics 2. Calculus 3. Linear Algebra 4. Probability Theory 5. Statistical Inference and Modeling
Specific Learning Outcomes	**By completing this EKU, students should be able to:** 1. Analyze discrete mathematical structures to solve computing-related problems. 2. Apply logic and proof techniques to validate the correctness of algorithms 3. Analyze the properties of infinite series expansions related to function approximation. 4. Calculate partial derivatives and use optimization methods for gradient-based learning. 5. Perform matrix operations and solve systems of linear equations. 6. Apply counting techniques to compute probabilities for discrete scenarios. 7. Formulate probabilistic models for solving problems involving uncertainty and dependence. 8. Apply hypothesis testing and confidence intervals to draw rigorous inferences about data populations. **The table below maps the Specific Learning Outcomes for the EKU to the Key Learning Outcomes.**

	SLOs	KLOs									
		1	2	3	4	5	6	7	8	9	10
	1	✓	✓								
	2	✓	✓								
	3	✓									
	4	✓		✓							
	5	✓	✓								
	6	✓									
	7	✓		✓							
	8	✓	✓								

Essential Knowledge Unit (EKU.2): Physics

Description	This EKU provides fundamental knowledge of physical laws, with a focus on Mechanics and Electricity & Magnetism. It covers kinematics, work and energy, electric circuits, and magnetic fields. This unit includes mandatory experimental experience.
Topics	**The following topics must be included in this EKU:** 1. Mechanics 2. Electricity & Magnetism 3. Optics/Thermodynamics
Specific Learning Outcomes	**By completing this EKU, students should be able to:** 1. Apply conservation laws and mechanics principles to analyze physical phenomena in computing systems. 2. Analyze the behavior of oscillatory systems and wave propagation phenomena (mechanical and electromagnetic). 3. Analyze performance limitations and failure modes of hardware based on electromagnetic constraints. 4. Utilize principles of thermodynamics to analyze energy transfer, heat management, and cooling needs for high-density computing. 5. Assess thermodynamic cycles to compare operational effectiveness. **The table below maps the Specific Learning Outcomes for the EKU to the Key Learning Outcomes.**

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1	✓	✓								
2	✓	✓								
3	✓	✓								
4	✓		✓							
5		✓	✓				✓			

Essential Knowledge Unit (EKU.3): General Chemistry

Description	Chemistry provides the scientific basis for understanding material properties, energy conversion, and device stability. This unit covers the fundamental principles of atomic structure, chemical bonding, and electrochemistry necessary for analyzing material limitations and assessing environmental impact.
Topics	**The following topics must be included in this EKU:** 1. Atomic Structure and Bonding 2. Stoichiometry and Chemical Reactions 3. Electrochemistry and Energy Systems
Specific Learning Outcomes	**By completing this EKU, students should be able to:** 1. Describe the relationship between atomic structure, bonding mechanisms, and material periodicity. 2. Relate material structure (molecular/crystal) to macroscopic physical properties. 3. Apply stoichiometry and conservation principles to quantify chemical reactions and mass/energy balances. 4. Analyze energy conversion mechanisms and quantify electrochemical processes. 5. Analyze material and energy considerations to assess environmental impact and inform sustainable design choices. **The table below maps the Specific Learning Outcomes for the EKU to the Key Learning Outcomes.**

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1	✓									
2	✓									
3	✓									
4	✓	✓								
5			✓				✓			

Essential Knowledge Unit (EKU.4): Project Management and Finance

Description	This unit provides the essential professional knowledge in project management principles, financial planning, budgeting, and economic decision-making, which are explicitly mandated as foundational components of the CS EKU set. This knowledge is crucial for graduates to manage projects and justify solutions under economic and business constraints.
Topics	**The following topics must be included in this EKU:** 1. Project Management Fundamentals (Project Life Cycle, Planning, Scope, and Scheduling) 2. Risk, Resource, and Budget Management (Risk Identification, Resource Allocation, and Budgeting Principles) 3. Economic Decision Making (Time Value of Money, Feasibility Studies, and Cost/Benefit Analysis)
Specific Learning Outcomes	**By completing this EKU, students should be able to:** 1. Develop a basic project plan including a defined scope, WBS, and realistic scheduling. 2. Analyze common project risks and allocate resources effectively within a defined budget. 3. Apply budgeting principles to manage project funds and control expenditures throughout the project lifecycle. 4. Evaluate the economic feasibility of technology projects using cost/benefit analysis and time value of money concepts. 5. Apply financial analysis techniques to assess economic viability and return on investment of proposed computing solutions. **The table below maps the Specific Learning Outcomes for the EKU to the Key Learning Outcomes.**

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1						✓				
2						✓				
3						✓				
4						✓				
5						✓				

General Knowledge Unit (GKU) 1: Algorithms, Computability, and Complexity

Description	The Knowledge Units of Algorithm, Computability, and Complexity encompass the key concepts and skills necessary for designing, implementing, and analyzing algorithms to solve problems. Algorithm concepts are essential to both computer science and software engineering. The performance of any software system in real-world scenarios relies heavily on the chosen algorithms and the effectiveness and efficiency of the various implementation layers. Consequently, crafting an effective algorithm is vital for the performance of any software system. In addition, the algorithm knowledge provides insight into the intrinsic nature of the problem as well as possible solution techniques independent of programming language, programming paradigm, computer hardware, or any other implementation aspect. Thus, it serves as a foundational blueprint for solving problems, enabling the design and analysis of efficient solutions. By abstracting the core logic, algorithms help understand the problem's complexity and guide the selection of appropriate data structures and methods, ensuring that solutions remain adaptable and scalable regardless of specific implementation constraints.

Specific Knowledge Unit (SKU) 1.1: Basic analysis and algorithmic methods

Description	This SKU focuses on the design, implementation, and analysis of algorithms to solve problems efficiently, providing a foundation for understanding problem complexity, performance, and solution techniques independent of implementation specifics.
Topics	**The following topics must be included in this SKU:** 1. Algorithm performance and complexity analysis. 2. Algorithm design techniques.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Explain the case behaviors of an algorithm. 2. Use big O notation to determine the complexity of algorithms. 3. Contrast standard complexity classes. 4. Perform empirical studies to validate hypotheses about runtime and test algorithms with different input sizes to compare performance. 5. Use an algorithmic method (greedy, divide-and-conquer, recursive backtracking, dynamic programming, and heuristic approach) to solve an appropriate problem. 6. Determine the appropriate algorithmic method to solve a problem. 7. Implement a divide-and-conquer algorithm for solving a problem.

The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1	✓	✓								
2	✓	✓								
3	✓	✓								
4		✓								
5			✓							
6		✓								
7			✓							

Specific Knowledge Unit (SKU) 1.2: Data structures, automata, and computability

Description	This SKU explores the core concepts that link algorithms, data structures, and computability, emphasizing how foundational logic and computational theories guide the selection of efficient methods and structures to address complex problems effectively.
Topics	**The following topics must be included in this SKU:** 1. Algorithm design and efficiency. 2. Data structure implementation. 3. Automata theory and formal languages. 4. Computational complexity.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Implement basic numerical, search, and sorting algorithms. 2. Discuss the runtime and memory efficiency of principal algorithms. 3. Describe the implementation of hash tables. 4. Solve problems using fundamental graph algorithms. 5. Evaluate an algorithm, justify the selection, and apply the algorithm to a certain problem. 6. Choose the appropriate data structure for modeling a given problem. 7. Implement a string-matching algorithm. 8. Design a deterministic finite state automaton to accept a specified language. **The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.**

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1	✓		✓							
2	✓	✓								
3	✓									
4			✓							
5		✓								
6		✓	✓							
7			✓							
8			✓							

General Knowledge Unit (GKU) 2: Architecture and Organization

Description	The Computer Architecture and Organization GKU focuses on providing computer science students with a foundational understanding of the functional components of a computer system, their features, performance characteristics, and interactions. To structure efficient programs, students must first grasp the principles of computer architecture and learn to evaluate tradeoffs, such as CPU clock speed versus memory size, when optimizing system performance. This knowledge area covers topics such as digital logic, digital systems, and the representation of data at the machine level. It also addresses machine organization at the assembly level, and the architecture of memory systems, interfacing, and system communication.

Specific Knowledge Unit (SKU) 2.1: Digital logic and machine-level data representation

Description	This area focuses on the principles of digital logic, the design of hardware systems, and the methods for representing and processing data at the machine level. It includes the operation of fundamental components, data structures, and the interplay between logic and architecture in computing systems.
Topics	**The following topics must be included in this SKU:** 1. Evolution and architecture of computing systems. 2. Logical and functional representations. 3. Data representation and manipulation.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Analyze the advancements across the five generations of computers and evaluate their impact on modern computing. 2. Compare multi-core and single-core architectures and justify the benefits of multi-core systems for performance and scalability. 3. Design logical circuits using expressions and gates for optimized performance. 4. Design and prototype the basic building blocks of a computer: arithmetic-logic unit, registers, central processing unit, and memory. 5. Evaluate the correctness and efficiency of a simple processor implemented at the logic circuit level. 6. Convert and verify numerical data transformations across formats in practical scenarios. 7. Develop algorithms to manipulate non-numeric data representations. 8. Evaluate records and arrays for efficient storage and retrieval.

The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1	✓							✓		
2	✓	✓								
3	✓		✓							
4	✓		✓							
5			✓							
6	✓									
7	✓		✓							
8		✓								

Specific Knowledge Unit (SKU) 2.2: Machine organization at the assembly level

Description	This area examines how computers execute programs at the assembly and machine levels, focusing on the structure of computing systems, instruction representation and execution, and input/output (I/O) operations. It also covers parallelism, subroutine management, and translating high-level programs into machine-level instructions.
Topics	**The following topics must be included in this SKU:** 1. Structure and functionality of computing systems. 2. Instruction representation and execution. 3. Interrupts, subroutines, and I/O management.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Explain the organization and functionality of the von Neumann machine. 2. Analyze the use of instruction-level parallelism and evaluate its impact on program execution efficiency. 3. Design machine and assembly-level instruction sequences to solve a given computational problem. 4. Evaluate the role of interrupts and I/O operations in ensuring efficient communication between hardware and software. 5. Write simple assembly language program segments with subroutine calls. 6. Show how high-level programs are implemented at the machine-language level. **The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.**

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1	✓									
2	✓	✓								
3	✓		✓							
4		✓						✓		
5	✓		✓							
6	✓									

Specific Knowledge Unit (SKU) 2.3: Architecture of memory systems, interfacing, and communication

Description	This area focuses on memory technologies, management principles, and performance optimization, including caching and virtual memory. It explores data transfers, input/output (I/O) mechanisms, and system buses to enhance communication efficiency. Storage devices such as magnetic disk drives are also examined for effective data access.
Topics	**The following topics must be included in this SKU:** 1. Memory technologies, management, and performance. 2. Virtual memory and system functionality. 3. Data transfers, buses, and storage devices.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Evaluate memory technologies and management principles. 2. Analyze memory delay and propose strategies to improve runtime performance. 3. Design memory configurations to optimize average access time. 4. Simulate the functionality of a system using virtual memory management. 5. Implement data transfers and I/O control using interrupts in a simulation. 6. Justify the use of different bus types based on performance requirements. 7. Evaluate the use of magnetic disk drives for efficient data access. **The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.**

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1	✓	✓								
2		✓	✓							
3			✓							
4				✓						
5				✓						
6		✓								
7		✓								

General Knowledge Unit (GKU) 3: Discrete Structures

Description	Discrete structures are foundational material for computer science. Discrete structures include essential material from a set, logic, graph, and probability theories. The material in discrete structures is prevalent in data structures and algorithms but appears in other areas of computer science. For example, creating and understanding proof is essential in every area of computer science, including formal specification, verification, databases, and cryptography. Graph theory concepts are used in networks, operating systems, and compilers. Set theory concepts are used in software engineering and databases. Probability theory is used in intelligent systems and computer applications.

Specific Knowledge Unit (SKU) 3.1: Introductory logic, sets, relations, and functions

Description	This unit covers key concepts in functions, relations, and sets, focusing on their operations and interactions. It explores propositional and predicate logic, including how informal logical statements are converted into predicate logic expressions. Students will learn symbolic methods in propositional and predicate logic and apply inference rules to prove logical statements. The unit builds a foundation in mathematical reasoning essential for advanced topics in computer science and discrete mathematics.
Topics	**The following topics must be included in this SKU:** 1. Basics of sets, functions, relations, and their operations. 2. Propositional and predicate logic, conversion of informal statements, and symbolic methods. 3. Using inference rules for proofs in propositional and predicate logic.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Illustrate functions, relations, and sets with appropriate examples. 2. Apply the operations of sets, functions, and relations to solve problems. 3. Explain propositional and predicate logic concepts. 4. Translate informal logical statements into predicate logic expressions. 5. Employ methods of symbolic propositional and predicate logic to solve logical problems. 6. Construct valid proofs in propositional and predicate logic using appropriate inference rules. 7. Model real-life situations using symbolic logic to represent and analyze scenarios.

The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1	✓									
2		✓	✓							
3	✓									
4		✓								
5		✓								
6		✓								
7		✓								

Specific Knowledge Unit (SKU) 3.2: Proof techniques and basics of counting

Description	This unit covers essential mathematical concepts, including proof techniques (direct proof, contradiction, and induction), the relationship between weak and strong induction, and the well-ordering principle. It also explores counting methods such as the sum and product rules, inclusion-exclusion, progressions, and the pigeonhole principle. Additionally, it includes permutations, combinations, recurrence relations, and modular arithmetic, all of which are crucial for problem-solving and analysis in discrete mathematics and computer science.
Topics	**The following topics must be included in this SKU:** 1. Proof techniques and induction. 2. Counting and combinatorics. 3. Recurrence relations and modular arithmetic.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Identify the proof technique used in a given proof (direct proof, proof by contradiction, and induction). 2. Explain the relationship between weak and robust induction, and the well-ordering principle. 3. Apply counting arguments, including sum and product rules, inclusion-exclusion principle, and arithmetic/geometric progressions. 4. Use the pigeonhole principle effectively in formal proofs in discrete mathematics. 5. Compute permutations and combinations of a set. 6. Solve different fundamental recurrence relations. 7. Perform modular arithmetic-based computations. **The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.**

	SLOs	KLOs									
		1	2	3	4	5	6	7	8	9	10
	1		✓								
	2	✓	✓								
	3		✓								
	4		✓								
	5		✓								
	6		✓								
	7		✓								

Specific Knowledge Unit (SKU) 3.3: Graphs, trees, and discrete probability

Description	This unit covers graph theory, including types of graphs, tree traversal algorithms, spanning trees, and isomorphism. It also explores probability concepts such as dependent and independent events, binomial distribution, Bayes' theorem, conditional probabilities, and variance of a probability distribution.
Topics	**The following topics must be included in this SKU:** 1. Graph theory and tree concepts. 2. Probability and probability distributions. 3. Bayesian probability.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Recognize graph theory concepts, including types of graphs and trees, and traversal methods for trees and graphs. 2. Apply graph and tree structures to solve real-world computer science problems, including constructing spanning trees and identifying graph isomorphisms. 3. Compute probabilities for basic events and apply Bayes' theorem to calculate conditional probabilities in relevant problems. 4. Distinguish between dependent and independent events, providing real-world examples to illustrate the differences. 5. Compute probabilities using binomial distributions, including calculating variance and its applications. 6. Use probability tools and techniques to solve problems in a variety of contexts. 7. Evaluate real-world problems involving event dependencies and graph structures using graph theory and probability. **The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.**

	SLOs	KLOs									
		1	2	3	4	5	6	7	8	9	10
	1	✓									
	2		✓	✓							
	3		✓								
	4		✓								
	5		✓								
	6		✓								
	7		✓								

General Knowledge Unit (GKU) 4: Information Management

Description	The capture, digitization, representation, organization, transformation, and presentation of information; algorithms for efficient and effective access and updating of stored information; data modeling and abstraction; and physical file storage techniques are all topics covered by information management. The student must be able to create conceptual and physical data models, decide which information management approaches and techniques are appropriate for a particular challenge, and choose and execute an appropriate information management solution that addresses design concerns such as scalability, accessibility, and usability.

Specific Knowledge Unit (SKU) 4.1: Database systems and data modeling

Description	This unit covers the key components of database systems, focusing on their design and functions within a database management system (DBMS). It explores declarative query languages, the relational data model, and modeling notations such as entity-relationship diagrams (ERDs) and UML. The unit compares different data models, including the object-oriented model, with an emphasis on understanding the relational model and database design principles.
Topics	**The following topics must be included in this SKU:** 1. Database system components and functions. 2. Database models and data modeling techniques. 3. Relational data models and query languages for object-oriented database modeling.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Identify the core components of a database system, including the query optimizer, query executor, storage manager, access methods, and transaction processor, and their roles within the system. 2. Discuss the functions and responsibilities of key DBMS components, such as data storage, query processing, and transaction management. 3. Explain the principles of the relational data model, including basic concepts and the use of relational databases. 4. Apply a declarative query language (e.g., SQL) to retrieve and manipulate data from a database, demonstrating an understanding of query formulation and execution. 5. Compare different data models (e.g., relational, object-oriented) and their internal structures, including their application to various data types. 6. Apply appropriate modeling notations, such as ERDs or UML, to design and describe database structures and relationships.

7. Apply the main concepts and principles of the object-oriented data model to design and implement database.

The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1	✓									
2	✓	✓								
3	✓									
4		✓	✓							
5	✓	✓								
6			✓							
7			✓							

General Knowledge Unit (GKU) 5: Networking

Description	Networking focuses on the principles and technologies that enable devices to connect and interact in a structured environment. The topics in this category include networking fundamentals, such as network architectures, protocols, and addressing schemes, as well as networked applications that leverage these infrastructures to provide services like web, email, and cloud-based solutions. Additionally, this area explores strategies for ensuring reliable data delivery, covering error detection, congestion control, and mechanisms for maintaining consistency and reliability across networked systems.

Specific Knowledge Unit (SKU) 5.1: Networking fundamentals and networked applications

Description	This area covers network organization, structure, layering principles, and switching techniques such as circuit and packet switching. It explores the roles of different network layers and the physical components that enable communication. It covers naming, addressing, and resource location in networks, along with distributed application models like client/server and peer-to-peer. It covers socket APIs, error control, and flow control mechanisms
Topics	**The following topics must be included in this SKU:** 1. Organization and structure of networks. 2. Switching techniques and layering principles. 3. Physical components of networking. 4. Naming, addressing, resource location and flow control. 5. Distributed applications and communication interfaces.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Describe the organization and structure of the internet. 2. List and define the appropriate network terminology. 3. Assess different types of network complexity (e.g., edge, core). 4. Explain the principles and roles of network layering. 5. Explain the basics of naming schemes and resource location. 6. Design and implement a client-server socket-based application to meet specific requirements. 7. Develop and optimize a reliable protocol for efficient data delivery.

The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1	✓									
2	✓									
3		✓								
4	✓									
5	✓									
6			✓	✓						
7			✓	✓						

General Knowledge Unit (GKU) 6: Operating Systems

Description	An operating system is an essential software system that manages hardware resources and provides services to applications. The topics in this category cover key operating system principles, such as distinguishing between kernel and user modes, and understanding important design and implementation approaches. It includes the management of concurrency, scheduling, and dispatching tasks, as well as memory management techniques such as paging and segmentation. Additionally, the operating system implements mechanisms for ensuring security and protection, which are crucial for maintaining efficient, secure, and reliable system operations.

Specific Knowledge Unit (SKU) 6.1: Principles of operating systems

Description	This area explores the objectives and functions of modern operating systems, including resource management, interrupt processing, and kernel-user mode operations. It covers client-server, distributed, and single-user systems, logical layers, APIs, middleware, and device management through drivers and input/output (I/O) queues.
Topics	**The following topics must be included in this SKU:** 1. Objectives, functions, and types of operating systems. 2. Logical layers, APIs, and middleware. 3. Resource and process management. 4. Device and interrupt handling.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Explain the objectives and functions of modern operating systems. 2. Analyze client-server, distributed, and single-user operating systems and compare their advantages and limitations. 3. Explain the concept of a logical layer in system-level design. 4. Describe the role of APIs and middleware in system development. 5. Assess resource utilization by application software and propose strategies to optimize system software. 6. Compare and evaluate kernel and user modes in operating systems for secure and efficient execution. 7. Analyze interrupt processing mechanisms for improved system performance. 8. Evaluate device management strategies, including device lists and driver I/O queues.

The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1	✓									
2		✓								
3	✓									
4	✓									
5		✓								
6		✓						✓		
7		✓								
8		✓								

Specific Knowledge Unit (SKU) 6.2: Concurrency, scheduling, memory management, and OS protection

Description	This area explores concurrency mechanisms, task synchronization, and scheduling algorithms within operating systems (OS). It covers memory hierarchy, virtual memory, and resource allocation, along with OS protection, security features, and access control mechanisms to ensure system integrity.
Topics	**The following topics must be included in this SKU:** 1. Concurrency and synchronization. 2. Task and process management. 3. Memory management and optimization. 4. Operating system security.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Analyze the concurrent operation of multiple tasks and evaluate the associated runtime problems. 2. Assess the states of a task during the management of multiple tasks. 3. Evaluate techniques for achieving synchronization in an operating system. 4. Design state and transition diagrams for specific problem domains, such as task scheduling. 5. Compare and assess processes, threads, and scheduling algorithms in operating systems for different application domains. 6. Analyze memory hierarchy, virtual memory, cache memory, thrashing, and cost-performance tradeoffs. 7. Evaluate various methods for allocating memory to tasks. 8. Analyze mechanisms in an OS to control access to resources and propose resource optimization enhancements. 9. Assess the importance, features, and limitations of an OS for protection and security. **The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.**

	SLOs	KLOs									
		1	2	3	4	5	6	7	8	9	10
	1		✓								
	2		✓								
	3		✓	✓							
	4			✓							
	5		✓								
	6		✓								
	7		✓								
	8		✓	✓							
	9		✓						✓		

General Knowledge Unit (GKU) 7: Programming Languages

Description	Programming languages are the means by which programmers express themselves explicitly, create algorithms, and reason about solutions. A computer science student will work with various languages throughout his or her career, alone or simultaneously. Software developers must be familiar with the programming models underpinning various languages to make educated design decisions in languages that enable numerous complementary approaches. A computer science student will frequently be required to master new languages, programming techniques, and the ideas underpinning the definition, composition, and implementation of programming language features.

Specific Knowledge Unit (SKU) 7.1: Object-oriented programming and functional programming

Description	This unit covers object-oriented concepts like class hierarchy, reusability, dynamic dispatch, inheritance, and encapsulation. It also explores functional programming techniques such as immutability, higher-order functions, and functional encapsulation. The usage of iterators and operations on aggregates across multiple languages is emphasized, along with a comparison of functional and object-oriented approaches. It also explores type systems, and how type systems detect program errors, along with advanced features like: generics, subtyping, and overloading to improve code flexibility and reuse.
Topics	**The following topics must be included in this SKU:** 1. Object-oriented concepts. 2. Functional programming concepts. 3. Comparing paradigms. 4. Operations on aggregates. 5. Type systems.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Design a class hierarchy that enables reusability through inheritance, subtyping, and encapsulation. 2. Compare the relationship between object-oriented inheritance, subtyping, and functional programming approaches. 3. Apply object-oriented encapsulation mechanisms and functional encapsulation techniques in writing simple programs. 4. Utilize iterators and aggregation operations in multiple programming languages to handle collections.

5. Construct algorithms that avoid mutable state assignment, focusing on immutability and functional approaches.

6. Develop higher-order functions that accept and return other functions to enhance program flexibility.

7. Apply knowledge of types, error handling, and generics to write well-structured, type-safe programs.

8. Apply various strategies to test and debug simple programs.

9. Apply the documentation and program style standards for software readability and maintainability.

The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1			✓							
2	✓	✓								
3			✓							
4				✓						
5		✓	✓							
6			✓							
7	✓		✓							
8		✓		✓						
9							✓			

Specific Knowledge Unit (SKU) 7.2: Program representation, language translation, and execution

Description	This topic covers the use of programs as input data, abstract syntax trees, and the roles of interpreters, optimizers, and documentation generators. It distinguishes between syntax and semantics, and discusses low-level memory organization into global, text, heap, and stack sections. It also addresses memory leaks, dangling pointers, and garbage collection techniques for efficient memory management.
Topics	**The following topics must be included in this SKU:** 1. Program input and representation. 2. Language processing concepts. 3. Memory organization and management. 4. Garbage collection.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Examine programs that use other programs as input data and interpret their representations. 2. Evaluate the structure and function of an abstract syntax tree for a given language. 3. Design a program to process and optimize code, such as an interpreter, expression optimizer, or documentation generator. 4. Differentiate between syntax, parsing, semantics, and evaluation in programming languages. 5. Construct the low-level runtime representation of core language constructs. 6. Assess programming language implementations that organize memory into global data, text, heap, and stack sections. 7. Resolve issues related to memory leaks and dangling pointer dereferences. **The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.**

	SLOs	KLOs									
		1	2	3	4	5	6	7	8	9	10
	1		✓								
	2		✓								
	3			✓							
	4		✓								
	5			✓							
	6		✓								
	7		✓								

General Knowledge Unit (GKU) 8: Software Engineering

Description	The Software Engineering knowledge area focuses on applying theoretical principles, knowledge, and practical techniques to design and develop dependable software systems that meet user and customer needs. Software engineering knowledge is relevant for projects of any size, from small applications to large-scale systems. The discipline covers every stage of the software lifecycle, including gathering and analyzing requirements, designing, building, testing, deploying, and maintaining software. It aims to find optimal methods for developing quality software, whether through traditional plan-driven approaches, agile methodologies, or other strategies. Software engineering relies on engineering practices, methodologies, tools, and metrics to manage development, assess quality, and ensure a systematic approach to software evolution and reuse.

Specific Knowledge Unit (SKU) 8.1: Software processes, software management, and requirements engineering

Description	This SKU emphasizes systematic methodologies for managing software projects, including requirement analysis, planning, and adopting suitable processes to deliver quality software that meets user and customer needs.
Topics	**The following topics must be included in this SKU:** 1. Software development processes and lifecycle. 2. Software system interaction and data modeling. 3. Software engineering foundations and requirements.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Describe how software can interact with various systems, including information management, embedded, process control, and communications systems. 2. Illustrate the different process models (e.g., waterfall, iterative, and agile). 3. Illustrate the phases of a software lifecycle and its deliverables through an example. 4. Analyze several examples of software risks. 5. Define the key components of a use case. 6. Analyze a given requirements model for a simple software system. 7. Describe the common techniques and fundamental challenges for requirements elicitation. 8. Explain the key components of a data model (e.g., class diagrams or ER diagrams). 9. Write functional and non-functional requirements for a software system.

The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.

SLOs	KLOs									
	1	2	3	4	5	6	7	8	9	10
1	✓									
2	✓		✓							
3	✓		✓							
4		✓								
5	✓	✓	✓							
6		✓								
7		✓								
8	✓		✓							
9		✓				✓				

Specific Knowledge Unit (SKU) 8.2: Software design, construction, verification, and validation

Description	This SKU encompasses the software lifecycle, focusing on design, construction, testing, and validation while emphasizing engineering practices, tools, and metrics to ensure quality, dependability, and adherence to user requirements.
Topics	**The following topics must be included in this SKU:** 1. Software design principles and architecture. 2. Software integration strategies and testing methods. 3. Software analysis and code quality assurance.
Specific Learning Outcomes	**By completing this SKU, students should be able to:** 1. Explain design principles such as coupling, cohesion, and encapsulation. 2. Use a design paradigm to design a simple software system. 3. Construct models of the design of a simple software system. 4. Describe the software architecture design of a simple software system. 5. Describe mechanisms for implementing designs to achieve desired properties such as reliability, efficiency, and robustness. 6. Describe integration strategies, including top-down, bottom-up, and sandwich integration. 7. Distinguish between program validation and verification. 8. Perform an inspection of a medium-sized code segment. 9. Create and document a set of tests for a medium-sized code segment. 10. Use a defect-tracking tool in a small software project. **The table below maps the Specific Learning Outcomes for the Specific Knowledge Unit to the Key Learning Outcomes.**

	SLOs	KLOs									
		1	2	3	4	5	6	7	8	9	10
	1	✓									
	2			✓							
	3			✓							
	4			✓							
	5			✓							
	6		✓	✓							
	7		✓								
	8		✓								
	9		✓	✓							
	10				✓						

هيئة تقويم التعليم والتدريب

Education & Training Evaluation Commission