MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية			
Module Title	**Programming Fundamentals**	**Module Delivery**	
Module Type	Core	☒ **Theory** ☒ **Lecture** ☒ **Lab** ☒ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	TUCS110		
ECTS Credits	6		
SWL (hr/sem)	**200**		
Module Level	1	**Semester of Delivery**	2nd
Administering Department	Computer Science	**College**	CCSM
Module Leader	Mohamed Taheer Ahmed	**e-mail**	Mohanad.H.Ramadhan@tu.edu.iq
Module Leader’s Acad. Title	Assistant Lecturer	**Module Leader’s Qualification**	master
Module Tutor		**e-mail**	
Peer Reviewer Name	Mahammed Aktham	**e-mail**	
Scientific Committee Approval Date	07/06/2023	**Version Number**	1.0

Relation with other Modules العلاقة مع المواد الدراسية الأخرى			
Prerequisite module	None	**Semester**	
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Aims أهداف المادة الدراسية	The aim of this module is to introduce students to the fundamental concepts of algorithms, algorithm design, and problem-solving techniques. The module will cover various algorithmic paradigms, data structures, and analysis methods to equip students with the skills necessary for designing and analyzing algorithms effectively.
Module Learning Outcomes مخرجات التعلم للمادة الدراسية	1. Understand the importance of algorithms in computer science and the significance of algorithmic problem-solving. 2. Design algorithms using flowcharts and pseudocode, and implement them using programming constructs such as flow control statements and loops. 3. Analyze the time and space complexity of algorithms using Big O notation and asymptotic analysis. 4. Implement and utilize basic data structures such as arrays, strings, stacks, and queues for algorithmic problem-solving. 5. Apply various sorting and searching algorithms, including bubble sort, selection sort, insertion sort, quicksort, mergesort, heapsort, linear search, binary search, depth-first search, and breadth-first search. 6. Utilize string algorithms for pattern matching and string manipulation tasks. 7. Demonstrate the ability to review and evaluate projects related to algorithm design and implementation.
Indicative Contents المحتويات الإرشادية	1. Introduction to algorithms: Understanding the role and significance of algorithms in computer science. 2. Algorithmic problem-solving: Exploring strategies and techniques for solving computational problems effectively. 3. Algorithm design: Drawing flowcharts and writing pseudocode to represent algorithmic solutions. 4. Flow control: Implementing flow control statements (if-else, switch-case) for decision-making in algorithms. 5. Loops: Utilizing loops for repetitive tasks, including counter and cumulative variables, and nested loops. 6. Complexity analysis: Analyzing the time and space complexity of algorithms using Big O notation and asymptotic analysis. 7. Basic data structures: Introduction to arrays, strings, stacks, and queues for storing and manipulating data. 8. Sorting algorithms: Implementing and analyzing sorting algorithms such as bubble sort, selection sort, insertion sort, quicksort, mergesort, and heapsort. 9. Searching algorithms: Implementing and analyzing searching algorithms such as linear search, binary search, depth-first search, and breadth-first search. 10. String algorithms: Exploring algorithms for pattern matching and string manipulation tasks. 11. Reviewing students' projects: Providing feedback and evaluation on projects related to algorithm design and implementation.

Learning and Teaching Strategies
استراتيجيات التعلم والتعليم

Strategies	Lectures: Traditional lectures can be used to introduce key concepts, theories, and principles related to algorithms. Lectures should be interactive, incorporating examples, demonstrations, and real-world applications to illustrate abstract concepts effectively. Group Discussions: Group discussions encourage collaborative learning and critical thinking. Students can discuss challenging topics, share insights, and work together to solve algorithmic problems. Group discussions also promote communication skills and teamwork. Problem-Solving Sessions: Dedicated problem-solving sessions allow students to practice applying algorithmic techniques to solve a variety of problems. These sessions can involve solving algorithmic puzzles, coding challenges, and algorithm design exercises individually or in groups. Practical Coding Assignments: Assigning practical coding assignments allows students to implement algorithms and data structures in programming languages of their choice. Through coding assignments, students gain hands-on experience with algorithm implementation, debugging, and optimization. Case Studies: Case studies provide real-world examples of how algorithms are used to solve practical problems in various domains, such as finance, healthcare, and engineering. Analyzing case studies helps students understand the relevance and applicability of algorithms in different contexts.

Student Workload (SWL)
الحمل الدراسي للطالب محسوب لـ ١٥ أسبوعا

Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	92	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	6.13
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	108	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	7.2
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	200		

Module Evaluation

تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	**Quizzes**	2	10% (10)	5, 11	#LO 1-3, #LO 5-8
	Assignments	2	10% (10)	7, 12	#LO 3-5, #LO 5-8
	Projects	1	10% (10)	continuous	
	Report	1	10% (10)	14	#LO 1-8
Summative assessment	**Midterm Exam**	2 hr	10% (10)	11	#LO 1-7
	Final Exam	2 hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)

المنهاج الاسبوعي النظري

Week No.	Material Covered
Week 1	Importance of algorithms in computer science
Week 2	Importance of algorithmic problem-solving
Week 3	Algorithms Design Drawing Flowchart and Writing pseudocode
Week 4	Flow Control (if-else), (switch – case)
Week 5	Loops (counter and cumulative variables), Nested Loops
Week 6	Time complexity analysis (Big O notation), Space complexity analysis and Asymptotic analysis
Week 7	Midterm exam
Week 8	Basic Data Structures: Arrays, Strings, Stacks, Queues.
Week 9	Sorting Algorithms: Bubble sort, selection sort, insertion sort
Week 10	Sorting Algorithms: Quicksort, mergesort, heapsort
Week 11	Searching Algorithms: Linear search, binary search
Week 12	Searching Algorithms: Depth-first search, breadth-first search
Week 13	String Algorithms: Pattern matching algorithms
Week 14	String Algorithms: String manipulation techniques
Week 15	Reviewing Students' Projects

Delivery Plan (Weekly Lab. Syllabus): المنهاج الاسبوعي للمختبر:	
Week No.	Material Covered
Week 1	Introduction to Algorithm Design Overview of the course objectives and expectations Introduction to algorithm design methodologies Hands-on activity: Drawing flowcharts for simple algorithms Assignment: Practice drawing flowcharts for algorithmic problems
Week 2	Review of pseudocode and its importance in algorithm design Introduction to flow control statements (if-else, switch-case) Hands-on activity: Writing pseudocode for algorithmic problems Assignment: Implementing algorithms using flow control in a programming language
Week 3	Understanding loop structures and their importance in algorithms Hands-on activity: Implementing loops for counter and cumulative variables Introduction to nested loops Assignment: Solving algorithmic problems using nested loops
Week 4	Time Complexity Analysis Introduction to time complexity analysis using Big O notation Understanding the concept of asymptotic analysis Hands-on activity: Analyzing the time complexity of algorithms Assignment: Analyzing the time complexity of sorting algorithms
Week 5	Space Complexity Analysis Introduction to space complexity analysis Hands-on activity: Analyzing the space complexity of algorithms Assignment: Analyzing the space complexity of searching algorithms
Week 6	Basic Data Structures Introduction to arrays, strings, stacks, and queues Hands-on activity: Implementing basic data structures in a programming language Assignment: Implementing algorithms using basic data structures
Week 7	Sorting Algorithms Introduction to sorting algorithms: bubble sort, selection sort, insertion sort Hands-on activity: Implementing sorting algorithms Assignment: Comparing the performance of different sorting algorithms
Week 8	Sorting Algorithms (continued) Introduction to more advanced sorting algorithms: quicksort, mergesort, heapsort Hands-on activity: Implementing advanced sorting algorithms Assignment: Optimizing sorting algorithms for different datasets
Week 9	Searching Algorithms Introduction to searching algorithms: linear search, binary search Hands-on activity: Implementing searching algorithms Assignment: Analyzing the performance of searching algorithms
Week 10	Graph Algorithms Introduction to graph algorithms: depth-first search, breadth-first search Hands-on activity: Implementing graph traversal algorithms Assignment: Solving graph-related problems using depth-first search and breadth-first search
Week 11	String Algorithms Introduction to string matching algorithms Hands-on activity: Implementing pattern matching algorithms Assignment: Applying string manipulation techniques to solve algorithmic problems

Week 12	Review and Project Work
Week 13	Project Work and Consultation Project work: Students continue working on their projects Individual consultations with the instructor for project guidance and feedback
Week 14	Project Presentation Preparation Preparation for project presentations Practice sessions for project presentations Final touches on project implementations and documentation
Week 15	Project Presentations

Learning and Teaching Resources

مصادر التعلم والتدريس

	Text	**Available in the Library?**
Required Texts	Introduction to Algorithms, Third Edition By Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein	No
Recommended Texts	Introduction to Algorithmic Design and Analysis	No
Websites		

Grading Scheme

مخطط الدرجات

Group	**Grade**	التقدير	**Marks (%)**	**Definition**
Success Group (50 - 100)	**A -** Excellent	امتياز	90 - 100	Outstanding Performance
	B - Very Good	جيد جدا	80 - 89	Above average with some errors
	C - Good	جيد	70 - 79	Sound work with notable errors
	D - Satisfactory	متوسط	60 - 69	Fair but with major shortcomings
	E - Sufficient	مقبول	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX –** Fail	راسب (قيد المعالجة)	(45-49)	More work required but credit awarded
	F – Fail	راسب	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.