Data Structures (CUE31023)

1. Information on the Programme

1.1. Higher Education Institution	Cihan University Sulaimaniya
1.2. College	Science
1.3. Department	Computer Science
1.4. Field of Study	Data Structures
1.5. Cycle of Study[1]	1
1.6. Specialization/ Study Programme	Computer Science
1.7. Form of Education	Full Time

2. Information on the Discipline

2.1. Discipline Name	Data Structures							
2.2. Code	CUE31023							
2.3. Language:	English							
2.4. (Theory) Lecturer **E-mail:** **Tel:** **Webpage, Google Classroom**	Dr. Asan Baker Kanbar **asan.baker@sulicihan.edu.krd** 07702396919							
2.5. Practical/Seminar/ Laboratory/ Project Lecturer **e-mail:** **Tel:** **Webpage, Google Classroom**	Dr. Asan Baker Kanbar **asan.baker@sulicihan.edu.krd** 07702396919							
2.6. Year of Study	-2022 2023	2.7 .Semester	1st	2.8. Assessment Type[2]	Written exam, & CE	2.9. Discipline Status	Content[3]	CD
							Mandotary[4]	MD

3. Total estimated time (Teaching Hours per Semester)

College of Science	
Department:	**Computer Science Department**
Decipline:	**Data structures**
Stage:	**2nd**

Total Contact Hours:	**52**
Total Self Study Hours:	**110**
Total No. Hours:	**162**
ECTS:	**6.00**

No. of Weeks	Contact Hours					Self Study					
	Theoretical	Practical	Lab.	Tutorial	Visit	Quiz	Reading	Assignment	Report	Midterm Exam.	Final Exam.
1st Week (Registration)	-	-	-	-	-	-	-	-	-	-	-
2nd Week	2		2				2			10	20
3rd Week	2		2				2	3			
4th Week	2		2			3	2				
5th Week	2		2				2	3			
6th Week	2		2			3	5		5		
7th Week	2		2				2				
8th Week	2		2				2			10	
9th Week	2		2				2	3			
10th Week	2		2			3	4		5		
11th Week	2		2				2				
12th Week	2		2				2	3			
13th Week	2		2			3	4				
14th Week	2		2				5				
15th Week (Final Exam.)		-	-	-	-	-	-	-	-	-	-
16th Week (Final Exam.)	-	-	-	-	-	-	-	-	-	-	-
TOTAL	26	0	26	0	0	12	36	12	10	20	20

4. Prerequisites (if applicable)

4.1 Curriculum-Related	
4.2 Skills-Related	Microsoft visual C ++ and visual code

5. Conditions (if applicable)

5.1. For the Theoretical	1. Read and comprehend the textbook material. 2. Attend all the classes and take notes on class discussions. 3. Actively participate in class discussions and activities. 4. Submit all the assignments and the project on time. 5. Pass tests and quizzes.
5.2. For the Practical	All students are normally required to attend the Lab; take part in lectures through applying the exercises on the computer or as quizzes, and to implement projects.

6. Cumulated Specific Competences

Professional Competencies	▪ Data Structure Knowledge: A deep understanding of various data structures such as arrays, linked lists, stacks, queues, trees, graphs, and hash tables. This includes knowledge of their properties, operations, time and space complexity, and trade-offs. ▪ Algorithmic Problem Solving: The ability to analyze problems and design efficient algorithms using appropriate data structures. This involves selecting the most suitable data structure based on problem requirements and implementing algorithms to manipulate and process data efficiently. ▪ Implementation Skills: Proficiency in implementing data structures in programming languages like C++, Java, or Python. This includes writing code for data structure operations, handling memory management, and considering error handling and edge cases. ▪ Efficiency and Performance Optimization: The skill to analyze and optimize the performance of data structures and algorithms. This involves evaluating time and space complexity, identifying bottlenecks, and employing optimization techniques to enhance efficiency. ▪ Algorithm Analysis and Complexity: The capability to analyze the time and space complexity of algorithms and understand their impact on program performance. This includes knowledge of Big O notation, worst-case, average-case, and best-case analysis, and the ability to compare and select appropriate algorithms based on efficiency requirements.
Transversal competences	• Problem-Solving: The ability to identify problems, analyze them, and devise effective solutions using data structures. This includes breaking down complex problems, applying critical thinking, and considering different approaches to arrive at optimal solutions. • Analytical Thinking: The skills to examine data structures and algorithms critically, understands their underlying principles, and evaluate their strengths and weaknesses. Analytical thinking helps in optimizing data structures, identifying patterns, and making informed decisions. • Logical Reasoning: The capacity to think logically and make logical connections between different elements of data structures. This involves understanding the flow of data and control within a program and being able to reason through the steps and outcomes of algorithms. • Attention to Detail: The ability to pay close attention to details when implementing and working with data structures. This includes ensuring accuracy, considering edge cases, and being meticulous in code writing, debugging, and documentation. • Collaboration and Teamwork: The skill to work effectively in a team environment when designing, implementing, and optimizing data structures. This involves communicating and collaborating with team members, sharing ideas, resolving conflicts, and collectively solving problems. •

7. Discipline Objectives (Based on the cumulated specific Competences)

7.1. General Objective	After going through this lesson, you would be able to: • Problem-Solving: The ability to identify problems, analyze them, and

	devise effective solutions using data structures. This includes breaking down complex problems, applying critical thinking, and considering different approaches to arrive at optimal solutions. • Analytical Thinking: The skill to examine data structures and algorithms critically, understand their underlying principles, and evaluate their strengths and weaknesses. Analytical thinking helps in optimizing data structures, identifying patterns, and making informed decisions. • Logical Reasoning: The capacity to think logically and make logical connections between different elements of data structures. This involves understanding the flow of data and control within a program and being able to reason through the steps and outcomes of algorithms. • Attention to Detail: The ability to pay close attention to details when implementing and working with data structures. This includes ensuring accuracy, considering edge cases, and being meticulous in code writing, debugging, and documentation. • Collaboration and Teamwork: The skill to work effectively in a team environment when designing, implementing, and optimizing data structures. This involves communicating and collaborating with team members, sharing ideas, resolving conflicts, and collectively solving problems. •
7.2. Specific Objectives	Understand the characteristics and properties of common data structures such as arrays, linked lists, stacks, queues, trees, graphs, and hash tables. Explain the basic operations and functionalities associated with each data structure, including insertion, deletion, searching, and traversal. Analyze the time and space complexity of data structure operations to evaluate their efficiency and performance. Select the most appropriate data structure for a given problem based on its requirements, constraints, and expected operations. Implement data structures using programming languages like C++, Java, or Python, including the necessary data manipulation and memory management operations. Demonstrate proficiency in applying algorithms and data structures to solve real-world problems, such as searching, sorting, graph traversal, and path finding. Design and implement advanced data structures like AVL trees, B-trees, or skip lists to handle more complex scenarios and optimize performance. Apply data structures in the design and implementation of efficient algorithms for various computational problems, including sorting, searching, and graph algorithms. Develop skills in analyzing and comparing different data structures to make informed decisions about their suitability for specific scenarios.

8. Content

week	8.1. Theoretical-Number of Hours	Teaching methods	Observation
1	registration		
2	Introduction to Data Structures and Algorithms • Overview of data structures and their importance • Introduction to algorithm analysis • Basic concepts of C++ programming	lecture	**1 lecture = 2 hours**
3	Arrays and Strings • Introduction to arrays and strings in C++ • Array operations and manipulation • String operations and manipulation	lecture, assignment	**1 lecture = 2 hours**
4	Linked Lists • Introduction to linked lists and their types • Linked list implementation in C++ • Linked list operations and manipulation	lecture, Quiz	**1 lecture = 2 hours**
5	Stacks and Queues • Introduction to stacks and queues • Stack and queue implementation using arrays and linked lists • Stack and queue operations and applications	lecture, assignment	**1 lecture = 2 hours**
6	Recursion • Understanding recursion and recursive algorithms • Recursive programming techniques • Recursive algorithms for problems like factorial, Fibonacci series, etc.	Lecture, Quiz (report1)	**1 lecture = 2 hours**
7	**MIDTERM EXAM 1**		
8	Introduction to sorting algorithms: bubble sort, insertion sort, selection sort • Divide and Conquer algorithms: merge sort, quicksort • Analysis of sorting algorithms	Lecture	**1 lecture = 2 hours**

9	Trees • Introduction to trees and their properties • Binary tree implementation in C++ • Tree traversal algorithms: preorder, inorder, postorder	lecture, Assignment	**1 lecture = 2 hours**
10	Binary Search Trees • Introduction to binary search trees (BST) • BST operations: insert, delete, search • Balanced BSTs: AVL trees and Red-Black trees	Lecture (Report2)	**1 lecture = 2 hours**
11	**MIDTERM EXAM 2**		
12	Introduction to heaps and priority queues • Heap implementation using arrays • Priority queue implementation using heaps	lecture, Assignment	**1 lecture = 2 hours**
13	Introduction to hashing and hash tables • Hash table implementation using arrays • Collision resolution techniques: chaining, open addressing	lecture, Quiz	**1 lecture = 2 hours**
14	Graphs • Introduction to graphs and their representations • Graph traversal algorithms: BFS and DFS • Shortest path algorithms: Dijkstra's algorithm	lecture	**1 lecture = 2 hours**

week	8.2. Practical Works–Number of Hours	Teaching methods	Observation
1	registration		
2	Introduction to the C++ programming language Basic C++ syntax and data types	Lecture	**1lecture = 2 hours**
3	Implementation and manipulation of arrays String operations and algorithms in C++	Lecture, Assignment	**1 lecture = 2 hours**
4	Implementation of singly linked lists Linked list operations and algorithms	Lecture, Quiz	**1 lecture = 2 hours**
5	Implementation of stacks using arrays and linked lists Implementation of queues using arrays and linked lists Applications of stacks and queues	Lecture, Assignment	**1 lecture = 2 hours**
6	Understanding recursion and recursive algorithms Implementing recursive algorithms for common problems	Lecture, Quiz	**1 lecture = 2 hours**
7	**MIDTERM EXAM 1**		**2 hours**
8	Implementation of various sorting algorithms: bubble sort, insertion sort, selection sort, merge sort, quicksort Analysis and comparison of sorting algorithms	Lecture, Quiz	**1 lecture = 2 hours**
9	Implementation of binary trees Tree traversal algorithms: preorder, inorder, postorder	Lecture, Assignment	**1 lecture = 2 hours**
10	Implementation of binary search trees (BST) BST operations: insert, delete, search Balancing BSTs: AVL trees or Red-Black trees	Lecture, Quiz	**1 lecture = 2 hours**
11	**MIDTERM EXAM 2**		**2 hours**
12	Implementation of binary search trees (BST) BST operations: insert, delete, search Balancing BSTs: AVL trees or Red-Black trees	Lecture, Quiz	**1 lecture = 2 hours**
13	Implementation of hash tables using arrays Collision resolution techniques: chaining, open addressing	Lecture	**1 lecture = 2 hours**
14	Implementation of graphs using adjacency matrix and adjacency list Graph traversal algorithms: BFS and DFS	Lecture, Assignment	**1 lecture = 2 hours**

- ***Compulsory Bibliography:***

Key references:

- Data structures and algorithm analysis in C++ by Mark Allen wiess
- Data Structure and program design in C++ by Robert L. Kruse
- Data Structure via C++ by Michael Berman
- Data Structures and Algorithms by Catherine Leung ,2017

ional Bibliography:

9. Assessment

Type of Activity	9.1. Assessment Criteria[2]	9.2. Assessment Type	9.3. Percentage of the final Grade
9.4. Theoretical	Mid-term (20%) Final –exam (35%)	Exam	**%55**
9.5. Practical/ Seminar/Laboratory	Mid-term (10%) Final-Exam (15%)	Exam	**%25**
9.6. Activity during Semester	Quizzes (10%) + Assignment (10%)+	Exam	**%20**
Minimum performance Standards:			

Theoretical Lecturer	**Dr. Asan Baker Kanbar**
Practice Lecturer	**Dr. Asan Baker Kanbar**

Approved by the Curriculum development Committee:	
1	
2	
3	
Head of the Department	Dr. Asan Baker Kanbar