2023
TP-153

Course Specification
(Bachelor)

Course Title:	Data Structures and Algorithms
Course Code:	CSC 1204
Program:	Bachelor in Computer Science
Department:	Computer Science
College:	Computers and Information Technology
Institution:	University of Tabuk
Version:	1.0
Last Revision Date:	27 July 2022

Table of Contents

A. General information about the course:

1. Course Identification

1. Credit hours: (3)					
2. Course type					
A.	☐University	☒ College	☐Department	☐Track	☐Others
B.	☒ Required		☐Elective		
3. Level/year at which this course is offered: (Level 4/Year 2)					
4. Course general Description:					
The fundamentals of data structures and algorithms are covered in this course. This course covers the principles of a number of data structures, including array, arraylist, stack, queue, linked list, tree, and graph. This course also covers time complexity analysis of algorithms using Big O notation, among other algorithm analysis techniques. There are several examples of sorting and searching algorithms are also coved in this course.					
5. Pre-requirements for this course (if any)**:**					
Object Oriented Programming -CSC1201					
6. Co-requisites for this course (if any)**:**					
N/A					
7. Course Main Objective(s):					
● To introduce data abstraction and data representation in memory ● To describe, design and use of elementary data structures such as stack, queue, linked list, tree and graph ● To discuss decomposition of complex programming problems into manageable sub problems ● To introduce algorithms and their complexity					

2. Teaching mode (mark all that apply)

No	Mode of Instruction	Contact Hours	Percentage
1	Traditional classroom	45	100%
2	E-learning	0	0%
3	Hybrid • Traditional classroom • E-learning	0	0%
4	Distance learning	0	0%

3. Contact Hours (based on the academic semester)

No	Activity	Contact Hours
1.	**Lectures**	45
2.	**Laboratory/Studio**	0
3.	**Field**	0
4.	**Tutorial**	0
5.	**Others (specify)**	0
Total		45

B. Course Learning Outcomes (CLOs), Teaching Strategies and Assessment Methods

Code	Course Learning Outcomes	Code of PLOs aligned with program	Teaching Strategies	Assessment Methods
1.0	**Knowledge and understanding**			
1.1	Appraise the use of mathematics in the solution of computing problems	K1	Lectures	Exam, Quiz, Assignment, Project
1.2	Identify the data structures such as stack, queue, binary tree, or graph required to solve a problem.	K2, K3	Lectures, Group discussion	Exam, Quiz, Assignment, Project
1.3	Recognize the theory, algorithm and design of algorithmic solutions.	K2, K3, K4	Lectures, Group discussion	Exam, Quiz, Assignment, Project
1.4	Recognize the correctness of algorithmic solutions.	K2, K3, K4	Lectures, Group discussion	Exam, Quiz, Assignment, Project
2.0	Skills			
2.1	Analyze the runtime performance of algorithms in terms of Big O, Big Omega, and Big Theta notation.	S1	Lectures	Exam, Quiz, Assignment, Project
2.2	Understand worse case, best case, average case and amortized analysis.	S2	Lectures, Group discussion	Exam, Quiz, Assignment, Project
2.3	Understand the distinction between algorithm correctness and performance.	S2	Lectures, Group discussion	Exam, Quiz, Assignment, Project

Code	Course Learning Outcomes	Code of PLOs aligned with program	Teaching Strategies	Assessment Methods
2.4	Differentiate between problems that are computable and those that are not.	S2	Lectures, Group discussion	Exam, Quiz, Assignment, Project
3.0	Values, autonomy, and responsibility			
3.1	Communicate and work (effectively, ethically, and professionally) (individually and in groups/teamwork) to accomplish all the assigned duties and projects.	V2	Class discussion	Project
3.2				
...				

C. Course Content

No	List of Topics	Contact Hours
1	**Introduction to Data Structures:** Exploring the significance of studying data structures, understanding its role in optimizing computer resources, and examining various types of data structures, abstract data types (ADTs), and collections. Additionally, the discussion extends to the ways in which algorithms contribute to problem analysis and solving, with a particular focus on their effectiveness in breaking down complex issues into more manageable components.	**3**
2	**Generic Types and Algorithm Efficiency Part1:** How to make an algorithm efficient, what factors can help for this, the role of input size for the space and time complexity, what to compare and how to compare when analyzing an algorithm.	**3**
3	**Generic Types and Algorithm Efficiency Part2:** The role of Big-O notation, how it help to analyze the growth of functions e.g., constant, linear, quadratic, peculiar, cubic, etc, along what can be the complexity of a peace of code are the part of this chapter.	**3**
4	**Recursive Functions** Recursive functions helps to add clarity also reduces the time needed to write and debug code, more importantly reduces time complexity. Further, it helps to performs better when solving problems based on different data structures like tree structures.	**3**
5	**Sorting Algorithms Part 1:**	**3**

	Sorting can often reduce the complexity of a problem while these algorithms have direct applications in methods like divide and conquer, simple comparison, incremental or two pointer approach etc.	
6	**Sorting Algorithms Part 2:** Organizing data in increasing or decreasing order using algorithms also help to sort characters by frequency, sort array by parity etc. At least 5 algorithms are part of this chapter including 3 basic and 2 advanced types.	**3**
7	**Searching Algorithms** These algorithms help to retrieve information stored within particular data structure, or calculated in the search space of a problem domain, with either discrete or continuous values. Under this chapter, the working of two major techniques including linear (sequential), and binary are explained in the class.	**3**
8	**Arrays and ArrayLists Part 1:** Arrays are important when dealing with large datasets and can be helpful when sorting and identifying variables. This topic is important when understanding what arrays are and how programmers use them, further give a better understanding during the application development.	**3**
9	**Arrays and ArrayLists Part 2:** Types of arrays like 1-D, 2-D arrays, Jagged Arrays, and advanced form called ArrayLists to overcome the major issues of Arrays.	**3**
10	**Linked Lists** Linked list is a key data structure when it comes to handling dynamic data elements also it is commonly used to implement data structures like stacks, queues, and hash tables. In this chapter students learn about how to add and delete nodes in the linked list at different positions of the list e.g., start, between, at end etc., while focusing both singly linked list and doubly linked list.	**3**
11	**Stacks** Under this topic, students gain a deep understanding of not only what a Stack is but also its key characteristics. Practical examples and visualisations presented during the lecture will allow students to see and learn Stack's functionality at play in real-world scenarios. While, the implementation of stacks with Arrays, ArrayLists and Linked list also discussed in the class.	**3**
12	**Queues** Understanding the queue data structure can be a great asset when dealing with different computer science problems. It's a simple and efficient data structure model that leads to better performance in many applications. Same like Stack topic, the implementation of Queues with Arrays, ArrayLists and Linked list also discussed in the class.	**3**
13	**Trees Part 1:** Tree data structures are commonly used in decision-making algorithms especially dealing with artificial intelligence, such as game-playing algorithms, expert systems, and decision trees.	**3**

14	**Trees Part 2:** Key applications of Trees, students learn about the basics of Tree data structures, tree types, tree traversing, adding, searing, deleting nodes in trees etc.	**3**
15	**Graphs** Graphs data structures are used to address real-world problems in which it represents the problem area as a network like telephone networks, circuit networks, and social networks. Under the Graph data structure, key topics includes graph types e.g., directed, weighted, connected, self-loop etc., visiting nodes in graph and convert graph to adjacency matrices. Students also learn graph traversal methods including breadth first search and depth first search conserving the Stacks and Queues ADTs.	**3**
Total		**45**

D. Students Assessment Activities

No	Assessment Activities *	Assessment timing (in week no)	Percentage of Total Assessment Score
1.	Class Works (Class Activities, Quizzes, Home-works and Projects)	1 - 10	20%
2.	Mid-Term Exam -1	6-7	20%
3.	Mid-Term Exam-2	11-12	20%
.4	Final Exam	16	40%

*Assessment Activities (i.e., Written test, oral test, oral presentation, group project, essay, etc.).

E. Learning Resources and Facilities

1. References and Learning Resources

Essential References	1) Algorithms, 4th edition by Robert Sedgewick and Kevin Wayne. Addison-Wesley Professional, 2011, ISBN 0-321-57351-X. 2) Introduction to Java Programming and Data Structures, Comprehensive Version 11th Edition, Pearson; (March 1, 2017), ISBN-10 : 9780134670942, ISBN-13 : 978-0134670942.
Supportive References	Introduction to Algorithms, Tomas H. Cormen et al., MIT press, 3rd edition, 2009, ISBN 978-0262033848.
Electronic Materials	Saudi Digital Library (SDL) (www.sdl.edu.sa)
Other Learning Materials	Introduction to the design and analysis of algorithms, Anany Levitin, Pearson, 3rd edition, 2011, ISBN 978-0-13-231681-1

2. Required Facilities and equipment

Items	Resources
Facilities (Classrooms, laboratories, exhibition rooms, simulation rooms, etc.)	**Classroom (50 seats at least)**
Technology equipment (projector, smart board, software)	White board, Data show projector
Other equipment (depending on the nature of the specialty)	N/A

F. Assessment of Course Quality

Assessment Areas/Issues	Assessor	Assessment Methods
Effectiveness of Teaching	Faculty, Program Leaders, and Advisory Board	Both Direct and Indirect
	Students	Indirect
Effectiveness of Students Assessment	Faculty, Program Leaders, Advisory Board, and Independent Opinion	Both Direct and Indirect
Quality of Learning Resources	Faculty, Students, and Advisory Board	Indirect
The Extent to which CLOs have been Achieved	Faculty, Program Leaders, Advisory Board, and Independent Opinion	Direct (as in section B) and Indirect/Surveys
	Students	Indirect
Other	-	-

Assessors (Students, Faculty, Program Leaders, Peer Reviewer, Others (specify)

Assessment Methods (Direct, Indirect)

G. Specification Approval

COUNCIL /COMMITTEE	
REFERENCE NO.	
DATE	