Register No : _____

THE KAVERY ENGINEERING COLLEGE

(An Autonomous Institution)

B.E/B.TECH DEGREE END SEMESTER THEORY EXAMINATIONS, APRIL/MAY 2026

Third Semester

Information Technology

UITT24301 / CD3291 - Data Structures and Algorithms

Regulations – 2024 & 2021

Duration : 3 Hrs *Maximum : 100 Marks*

*PART – A (ANSWER ALL QUESTIONS) (Marks 10*2=20)*

Q.No	*Questions*	*BTL*	*CO*	*Marks*
1.	Define Abstract Data Type. How is it different from a Data Structure?	L1	1	2
2.	Consider the code snippet: new_list = old_list.copy(). Is this a shallow or deep copy? Justify your answer.	L2	1	2
3.	State the primary advantage of a doubly linked list over a singly linked list.	L1	2	2
4.	For a stack implemented using a linked list, which end of the list (head or tail) would you choose to perform the push and pop operations? Justify your choice in one sentence.	L3	2	2
5.	Differentiate between the best-case and worst-case time complexity of the Quick sort algorithm.	L2	3	2
6.	A hash table of size 10 currently has 4 elements stored. Calculate the current load factor.	L3	3	2
7.	State the four types of rotations performed to balance an AVL Tree after an insertion or deletion.	L1	4	2
8.	A node in a max-heap has a value of 25. What are the possible values of its parent node? Justify your answer.	L3	4	2
9.	Define a Directed Acyclic Graph (DAG) and state one key property that distinguishes it from a general directed graph.	L1	5	2
10.	The Fibonacci number sequence can be computed by the recurrence F(n) = F(n-1) + F(n-2). Identify the problem with the naive recursive approach and name the technique used to overcome it.	L2	5	2

*PART – B (ANSWER ALL QUESTIONS) (Marks 5*16 = 80)*

Q.No			*Questions*	*BLT*	*CO*	*Marks*
11.	a	i	Demonstrate the implementation of Stack ADT using Python.	L2	1	8
		ii	Write a recursive Python function to implement the Binary Search algorithm. Analyze its time complexity using asymptotic (Big-O) notation and derive the recurrence relation for the same.	L3	1	8
			Or			
	b	i	Compare and contrast the principles of Inheritance and Composition in OOP.	L2	1	8
		ii	Write a recursive Python function to calculate the nth Fibonacci number. Analyze the time complexity of this naive recursive approach and explain why it is inefficient. Suggest a better approach to solve the same problem.	L3	1	8

12.	a	i	Compare and contrast the Array-based and Linked List-based implementations of the List ADT. Discuss the time complexities of common operations and the scenarios where one implementation would be preferred over the other.	L2	2	8
		ii	Describe how a stack would be used to manage a sequence of actions such as typing a character, deleting a word and formatting text. Explain what happens when the user performs multiple 'undo' operations.	L3	2	8
			Or			
	b	i	Explain the concept of a Circularly Linked List. Differentiate it from a standard Singly Linked List. Describe one practical application where a circularly linked list is particularly advantageous.	L2	2	8
		ii	Design an algorithm to check if a given string of parentheses (containing only '(' and ')') is balanced using a stack ADT. Provide a dry run of your algorithm with a valid string "(()())" and an invalid string "())(()".	L3	2	8
13.	a	i	Describe the working of the Merge Sort algorithm using a divide-and-conquer approach with an example.	L2	3	8
		ii	Explain the step-by-step process of performing a Binary Search to find the element '23' in the sorted array [4, 11, 16, 23, 45, 67, 89]. Analyze its time complexity and state the fundamental prerequisite for using this algorithm.	L3	3	8
			Or			
	b	i	Compare and contrast the Bubble sort, Selection sort, and Insertion sort algorithms based on their working methodology, best/worst-case time complexities and ideal use cases.	L2	3	8
		ii	Illustrate the process of Separate Chaining for collision resolution. Insert the keys [25, 11, 36, 5, 44, 67] into a hash table of size 10 using the hash function h(key) = key % 10. Draw the final structure of the hash table.	L3	3	8
14.	a	i	Construct an AVL Tree by inserting the following keys in the sequence given: 10, 20, 30, 40, 50, 25. Show the state of the tree after each insertion.	L3	4	8
		ii	Given the following sequence of elements: 15, 20, 5, 8, 3, 30, 25. Construct a min-heap from this sequence by repeatedly inserting the elements. Show the array representation of the heap after each insertion.	L3	4	8
			Or			
	b	i	Illustrate by inserting the following keys into an *empty B-Tree of order 3*: 10, 20, 5, 30, 40, 15, 25, 35, 50, 45. Show the tree after each insertion that causes a node split.	L3	4	8
		ii	Insert the following keys into an initially empty Binary Search Tree in the given order: 50, 30, 70, 20, 40, 60, 80, 55, 75. Delete the node containing the key 50 from the constructed BST. Show the state of the tree after each insertion and the final state after the deletion.	L3	4	8

15.	a	i	Explain the adjacency matrix and adjacency list representations of a graph. Compare their time complexities for checking an edge (u, v) and finding all neighbors of a vertex u. For a social network graph (sparse, with millions of users), justify which representation would be chosen.	L2	5	8
		ii	Prove that the greedy choice property holds for Prim's algorithm for finding a Minimum Spanning Tree.	L2	5	8
			Or			
	b	i	Illustrate Dijkstra's algorithm to find the shortest path with a sample graph.	L2	5	8
		ii	Illustrate topological sort on a Directed Acyclic Graph using Kahn's algorithm.	L2	5	8