**SENECA COLLEGE OF APPLIED ARTS AND TECHNOLOGY**

## FACULTY OF CONTINUING EDUCATION

**FINAL EXAMINATION**

# **Winter 2025**

**SUBJECT NAME: Data Structures and Algorithms**

### **SUBJECT CODE: DSA456V1A**

**EXAMINATION DATE: April 14th , 2025**

### **INSTRUCTOR NAME: AZER KARADAG**

**MARKS ALLOTTED: 100**

### **WEIGHTING: 30%**

**SPECIAL INSTRUCTIONS:**

Exam Books: Required: Not Required: X

Exam Aids: Permitted: X Not Permitted:

Exam Question Paper: Returned: Not Returned:X

**Exam approved by,**

**Sheri Ladoucier**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Sheri Ladoucier, Academic Program Manager**

**Academic Policy Section 9:**  
**Engaging in any form of academic dishonesty to obtain any type of advantage or credit is an offence and will not be tolerated by the College. Such offences under this policy include, but are not limited to, cheating, plagiarism, falsification, impersonation, misrepresentation and procurement.**

**SENECA COLLEGE OF APPLIED ARTS AND TECHNOLOGY**

**FACULTY OF CONTINUING EDUCATION**

**FINAL EXAMINATION**

**Student: Luca Novello - 038515003**

### **Exam: Data Structures and Algorithms** **TOTAL MARKS: 100**

# **PART A – Fundamental Concepts [2 marks each, 10 Marks Total]**

1. Define a data structure. Give two examples and state why data structures are important in programming.

**Answer:**

A data structure is a way of organizing and storing data so it can be accessed and modified efficiently. Examples include arrays and linked lists. They are essential for optimizing performance and managing complexity in algorithms.

1. Explain the difference between linear and non-linear data structures. Provide an example of each.

**Answer:**

Linear structures store data sequentially, while non-linear structures have hierarchical or networked relationships. Linear data allows single-path traversal, whereas non-linear supports multi-path.

1. What is recursion? Write a recursive function in pseudocode to compute the factorial of a number.

**Answer:**

Recursion is when a function calls itself to solve subproblems.

Pseudocode:

function factorial(n):

if n <= 1: return 1

else: return n \* factorial(n - 1)

1. Describe stack and queue. How are they different in terms of operations?

5. Explain what a priority queue is. How is it different from a regular queue?

# **PART B – Applied Data Structures [5 marks each, 50 Marks Total]**

6. Binary Search Tree Deletion  
You are given a BST with nodes: [45, 30, 60, 25, 35, 55, 70].  
Delete the node with value 30. Show the updated tree.

7. AVL Tree Balancing  
Insert the following into an AVL tree: [15, 20, 10, 25, 8].  
Show the AVL tree after each insertion, and explain any rotations needed.

8. Heap Sort  
Given the array: [4, 10, 3, 5, 1],  
Use heap sort to sort the array in ascending order. Show the intermediate steps.

9. Hash Table with Open Addressing  
Implement a hash table using linear probing with a table size of 7. Insert keys: [10, 20, 15, 7, 5].  
Show the final table.

10. Circular Linked List Insertion  
Given a circular linked list: [12 -> 23 -> 34 -> 45 -> 12],  
Write pseudocode to insert a new node with value 28 after the node with value 23.

11. Graph Representation (Adjacency List)  
Create an adjacency list for the graph with edges:  
(1, 2), (1, 3), (2, 4), (3, 4), (4, 5).

12. Depth-First Search (DFS)  
Perform DFS on the graph above starting from node 1. Show the order of traversal.

13. Implementing a Stack with an Array  
Write pseudocode to implement push and pop operations for a stack using a fixed-size array.

14. Advanced Trie Operations – Autocomplete Feature  
Given a trie storing the words: [“tree”, “trie”, “trip”, “track”, “trap”, “trick”],  
a) Show the trie structure.  
b) Write pseudocode for an autocomplete function that, given the prefix "tri", returns all matching words from the trie.  
c) What is the time complexity of this operation in terms of number of nodes?  
  
Hint:  
- Use DFS or BFS to explore all valid continuations of the prefix.  
- Keep track of word-end nodes as you build the result list.

15. Advanced Cycle Detection with Union-Find  
Given the undirected graph with edges:  
(1, 2), (2, 3), (3, 4), (4, 1), (3, 5)  
a) Use the Union-Find algorithm with path compression to detect whether the graph contains a cycle.  
b) Show the parent array and rank array after each union.  
c) What is the amortized time complexity?

# **PART C – Algorithms and Problem Solving [5 marks each, 40 Marks Total]**

16. Binary Search Tree – kth Smallest Element  
Given a Binary Search Tree (BST):  
 20  
 / \   
 10 30  
 / \ \   
 5 15 35  
a) Write an iterative algorithm to find the kth smallest element in the BST (assume k = 3).  
b) Explain how this algorithm leverages in-order traversal.  
c) Analyze its time and space complexity.

17. Dynamic Programming – Longest Palindromic Subsequence  
Given a string s = "agbdba",  
a) Write a dynamic programming algorithm to compute the length of the longest palindromic subsequence.  
b) Show the DP table used in your solution.  
c) What is the time and space complexity?  
  
Hint:  
- Use a 2D array dp[i][j], where dp[i][j] stores the length of the longest palindromic subsequence in s[i...j].  
- If s[i] == s[j], then add 2 and move inward; else, take the max between excluding s[i] or s[j].

18. Binary Search – Recursive Implementation  
Write a recursive implementation of binary search.  
Use it to search for 25 in the array [5, 10, 15, 20, 25, 30].

19. Fibonacci with Dynamic Programming  
Write a bottom-up DP solution to compute the nth Fibonacci number.

20. Topological Sort  
Given a directed acyclic graph (DAG) with vertices and edges:  
Vertices: A, B, C, D  
Edges: (A, B), (B, C), (A, C), (C, D)  
Perform a topological sort.

21. Prim’s Algorithm – MST  
Given the weighted graph:  
(A, B, 1), (A, C, 4), (B, C, 2), (B, D, 5), (C, D, 1)  
Use Prim’s algorithm to find the Minimum Spanning Tree starting at node A.

22. Knapsack – Recursive vs Dynamic Programming  
Explain the difference between recursive and DP solutions for the 0/1 knapsack problem.  
Which is more efficient and why?