DATA STRUCTURES B

1. **DESCRIPTION:** Participants will answer questions about fundamental data structures and their associated algorithms.
   A TEAM OF UP TO: 2 **APPROXIMATE TIME:** 50 minutes

2. **EVENT PARAMETERS:**
   a. Each team may bring one (1) 8.5" x 11" sheet of paper that may contain information on both sides in any form and from any source. The sheet of paper may be laminated or placed in a sheet protector to increase durability. Affixed labels, as well as multiple sheets of paper (whether in a single sheet protector or not), are prohibited.
   b. Each team is expected to bring their own writing utensils.
3. **THE COMPETITION:**
   a. The competition will consist of answering questions about fundamental data structures and algorithms, focusing on behavior, use cases, and problem-solving applications.
   b. Unless otherwise requested, answers involving runtime analysis should use standard asymptotic notation (Big-O, Big-Theta, Big-Omega). Participants should be familiar with analyzing the best, average, and worst-case runtimes of common algorithms and operations.
   c. Pseudocode and diagrams may be used where appropriate. Questions may involve interpreting, tracing, completing, or modifying pseudocode, as well as analyzing diagrams of data structures.
   d. Participants will be expected to know the following data structures and algorithms, in terms of concepts, applications, performance tradeoffs, and use cases, as well as specially listed topics:
      i. DATA STRUCTURES
         1. Arrays and Lists: memory organization and indexed access, contiguous storage, traversal/insertion/deletion/resizing operations, dynamic arrays
         2. Stacks and Queues: LIFO vs FIFO, push/pop/peek/enqueue/dequeue, function calls, recursion, buffering, scheduling, circular queues, overflow/underflow
         3. Hash Tables: simple hashing concepts, collisions, lookup complexity, dictionary, map, set behavior and applications
         4. Linked Lists: singly and doubly linked lists, node structure and pointer references, traversal/insertion/deletion
         5. Binary Trees: tree terminology (root, leaf, height, depth, subtree, parent, child), binary search tree properties, AVL trees, red-black trees, 2-4 trees, traversals (preorder, inorder, postorder, level-order), insertion/searching/deletion in search trees, balanced vs unbalanced trees
         6. Heaps and Priority Queues: heap properties and array representations, min/max-heaps, insertions/deletions/heapify, priority queue behavior, heap sort
         7. Graphs: directed vs undirected graphs, weighted vs unweighted graphs, graph representations (adjacency lists, adjacency matrices), paths, cycles, connectedness, traversal

DATA STRUCTURES B (CONT.)

ii. ALGORITHMS
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
6. Quick Sort
7. Merge Sort
8. Breadth-First Search (BFS) and Depth-First Search (DFS)
9. Dijkstra's Algorithm
10. Prim's and Kruskal's Algorithms

iii. ALGORITHMIC CONCEPTS
1. Time Complexity
2. Space Complexity
3. Divide-and-Conquer
4. Greedy Algorithms
5. Recursion

e. Participants will NOT be expected to know:
i. Programming language syntax or language-specific features
ii. Complete implementations of algorithms from memory
iii. Advanced graph algorithms beyond those listed in the rules
iv. Dynamic programming
v. NP-completeness, computational complexity classes, proofs of correctness
vi. Specialized data structures not explicitly listed in the rules

4. **SAMPLE QUESTIONS/TASKS:**
a. Explain what recursion means in programming.
b. Using Prim's algorithm, construct a minimum spanning tree for the following graph.
c. Compare the stability and complexity of Merge Sort and QuickSort.
d. Given the following pseudocode algorithm, determine its runtime expressed in Big-O notation.
e. In which of the following data structures can elements be accessed directly?
f. If an AVL tree has a balance factor of -1, what kind of rotation is performed to self-balance?

5. **NOTES FOR EVENT SUPERVISORS**
a. Students should not be expected to write code in any specific programming language. Pseudocode will be used whenever possible.
b. Students are not expected to write complete implementations of algorithms. Questions involving pseudocode should focus on interpreting, tracing, completing, modifying, and applying algorithms.
c. There should not be an emphasis on Leetcode type problems. Content should focus on theoretical applications of the data structures and algorithms.

DATA STRUCTURES B (CONT.)

6. **SCORING:**
    a. Highest score wins.
    b. Points will be awarded for the quality and accuracy of responses.
    c. Selected questions may be used as tiebreakers.

Recommended Resources:

W3Schools DSA Tutorial: https://www.w3schools.com/dsa/dsa_intro.php
GFG DSA Tutorial: https://www.geeksforgeeks.org/dsa/dsa-tutorial-learn-data-structures-and-algorithms/
DSA Roadmap: https://roadmap.sh/datastructures-and-algorithms