1. **ABET (Accreditation Board for Engineering and Technology, Inc.): Student Outcomes**

   1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.

   2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program’s discipline.

   6. Apply computer science theory and software development fundamentals to produce computing-based solutions. [CS]

2. **ACM (Association for Computing Machinery): The Body of Knowledge**

<table>
<thead>
<tr>
<th>Knowledge Area</th>
<th>Total Hours of Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Development Fundamentals (SDF)</td>
<td>2</td>
</tr>
<tr>
<td>Programming Languages (PL)</td>
<td>1</td>
</tr>
<tr>
<td>Algorithms and Complexity (AL)</td>
<td>34</td>
</tr>
</tbody>
</table>
## 3. Body of Knowledge Coverage

<table>
<thead>
<tr>
<th>Knowledge Area</th>
<th>Knowledge Unit</th>
<th>Topics Covered</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDF</td>
<td>algorithms and design</td>
<td>concept and properties of algorithms, role of algorithms, problem-solving strategies, separation of behavior and implementation</td>
<td>1</td>
</tr>
<tr>
<td>SDF</td>
<td>fundamental data structures</td>
<td>stacks, queues, linked structures</td>
<td>1</td>
</tr>
<tr>
<td>PL</td>
<td>objected-oriented programming</td>
<td>objected-oriented design, encapsulation, iterators</td>
<td>1</td>
</tr>
<tr>
<td>AL</td>
<td>basic analysis</td>
<td>all core-tier 1: best-, expected-, and worst-case behaviors of an algorithm, asymptotic notions/notations (big-O), complexity classes such as constant/logarithmic/polynomial/exponential, empirical measurements of performance, time/space trade-offs in algorithms all core-tier 2: big-O/Omega/Theta and little-O/Omega notions/notations and asymptotic analysis, recurrence relations and some version of Master Theorem, analysis of iterative and recursive algorithms</td>
<td>6</td>
</tr>
<tr>
<td>AL</td>
<td>algorithmic strategies</td>
<td>core-tier 1: brute-force, greedy approach, recursive divide-and-conquer, dynamic programming core-tier 2: heuristics</td>
<td>6</td>
</tr>
<tr>
<td>AL</td>
<td>fundamental data structures and algorithms</td>
<td>all core-tier 1: simple numerical algorithms, binary search, sorting algorithms (selection, insertion, quicksort, heapsort, mergesort) and their worst- or average-case analyses, hashing, binary search trees and their common operations, graph-representations and simple graph-traversals core-tier 2: heaps, graph-algorithms: shortest-path algorithms (Dijkstra’s and Floyd’s algorithms), minimum spanning tree (Prim’s and Kruskal’s algorithms)</td>
<td>12</td>
</tr>
<tr>
<td>AL</td>
<td>basic automata</td>
<td>core-tier 2: introduction to the ( P ), ( NP ), and ( NP )-complete classes</td>
<td>1.5</td>
</tr>
<tr>
<td>AL</td>
<td>advanced computational complexity</td>
<td>some classic ( NP )-complete problems, reductions</td>
<td>1.5</td>
</tr>
<tr>
<td>AL</td>
<td>advanced automata theory and computability</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>AL</td>
<td>advanced data structures algorithms and analysis</td>
<td>balanced trees, B-trees, disjoint-sets, graph-algorithms: topological sort and strongly connected components</td>
<td>7</td>
</tr>
</tbody>
</table>
4. Course Outline (Tentative)

1. Mathematical Preliminaries and Introductory Material
   Source: Lecture Notes, and [CLRS22] Chapters 1, 2, 3, and 4
   1.1 Asymptotic Notations
   1.2 Recurrences

2. Sorting
   Source: Lecture Notes, and [CLRS22] Chapters 6, 7, and 8
   2.1 Heapsort, Mergesort, and Quicksort
   2.2 Lower Bounds for Sorting

3. Elementary Data Structures: Lists, Stacks, and Queues
   Source: Lecture Notes, and [CLRS22] Chapter 10

4. Hashing
   Source: Lecture Notes, and [CLRS22] Chapter 11
   4.1 Hash Functions
   4.2 Chaining
   4.3 Open Addressing

5. Trees, Tree Traversals, Binary Trees, and Search Trees
   Source: Lecture Notes, and [CLRS22] Chapter 12
   5.1 Trees and Their Implementations
   5.2 Binary Trees and Their Implementations
   5.3 Tree Traversals
   5.4 Binary Search Trees

6. Priority Queues and Their Implementations
   Source: Lecture Notes, and [CLRS22] Chapter 6

7. Advanced Design Techniques and Data Structures
   Source: Lecture Notes, and [CLRS22] Chapters 14, 15, and 18
   7.1 Dynamic Programming
   7.2 Greedy Algorithms
   7.3 B-Trees

8. Graph Algorithms
   Source: Lecture Notes, and [CLRS22] Chapters 20, 21, and 22
   8.1 Undirected and Directed Graphs, and Their Representations
   8.2 Some Fundamental Undirected and Directed Graph Algorithms
   8.3 Graph Traversals
   8.4 Minimum Spanning Trees
   8.5 Shortest-Path Problems