CSC/CPE 202 – Data Structures

1. CSC/CPE 202 – Data Structures

2. **credit units** 4  **contact hours** 6

3. **Course Coordinator:** John Clements

4. **Textbook (or other required material):**

5. a. **Course Description:** Introduction to data structures and analysis of algorithms. Abstract datatypes. Specification and implementation of advanced data structures. Theoretical and empirical analysis of recursive and iterative algorithms. Software performance evaluation and testing techniques.

   b. **Prerequisite:** CPE/CSC 101 with a grade of C- or better; MATH 141 or MATH 221 with a grade of C- or better; or consent of instructor.

   c. **Required/Elective/Selective Elective for CPE, CSC, EE, SE**

<table>
<thead>
<tr>
<th>Required</th>
<th>CSC</th>
<th>CPE</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elective</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selective Elective</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. a. **Course Learning Objectives**

   The student will be able to:

   - Apply the steps of the design recipe to develop small programs in a way that scales to large programs.
   - Describe and define the Abstract Data Types (ADTs) covered, including lists, stacks, queues, priority queues, binary search trees, AVL trees, and hash tables.
   - Describe the meaning of operations on the given ADTs using complete test cases.
   - Explain, and diagram "by hand", functional and imperative implementations of methods associated with the given ADTs.
   - Analyze for performance some algorithms associated with the given ADTs.
   - Compare and contrast array and linked-list implementations of certain ADTs.
   - Compare and contrast imperative and functional implementations of given ADTs.
   - Implement and test each given ADT as a generic program.
   - Use ADTs in software applications.
   - Discuss the runtime and memory efficiency of principal algorithms for sorting.
   - Demonstrate different traversal methods for trees, including pre, post, and in-order traversal of trees.
   - Model a variety of real-world problems in computer science using appropriate forms of trees, such as representing the organization of a hierarchical file system.
   - Determine whether a recursive or iterative solution is most appropriate for a problem.
b. **Level at which Student Outcomes are addressed**

(“B” = Basic level, “I” = Intermediate level, “A” = Advanced level)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC</td>
<td>B</td>
<td>B</td>
<td></td>
<td></td>
<td>B</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>SE/ CPE</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

7. **Major Topics Covered: (number of lecture hours per)**

- The Design Recipe (1)
- Simple programs using Linked Lists (1)
- Collections, data structures, Abstract Data Types (ADT) (1)
- Introduction to algorithm analysis; working out the abstract running time of iterative, as well as recursive algorithms (2)
- Analysis of List ADT operations for alternate implementations (1)
- Basic Sorting Algorithm: Insertion sort (1)
- Advanced sorting algorithms: merge sort, quick sort; implementation and analysis (3)
- Iterators; supplying iterators for alternate implementations of List ADT (1)
- Stack ADT; alternate implementations and analysis (1)
- Queue ADT; alternate implementations and analysis (1)
- Priority Queue ADT; heap implementation and analysis (1)
- Binary Search Tree ADT; implementation and analysis (3)
- AVL Tree ADT; implementation and analysis (2)
- Binary tree traversal strategies: pre-order, in-order, post-order, and level order; supplying iterators for binary tree traversals (2)
- Hash Table ADT; alternate implementations and analysis (3)