**College of Engineering & Information Technology  
Department of Computer science  
Data Structures – Course Outline**

|  |
| --- |
|  |

**Text Books / References:**

Data Structures and algorithms in C++ by Michael T. Goodrich, RobertoTamassia and David Mount.

**Course Description:**

This course provides the students an opportunity to further develop and refine their programming and analytical skills. In particular, the emphasis of this course is on the organization of information and on the implementation of common data structures such as lists, stacks, queues, trees and hashing.The course also emphasizes on teaching the students to choose the most suitable and time-efficient data structure in solving problems.

Hands-on programming is a central component of this course. There will programming assignments and some laboratory sessions.

**Intended Learning Outcomes:**

|  |  |  |
| --- | --- | --- |
| **Intended Learning Outcomes (ILO's)** | **Student Outcomes  (SO's)** | **Contribution** |
| 1. Describe the [ADT](https://en.wikipedia.org/wiki/Abstract_data_type) of various data structures | **I** | **35%** |
| 1. Develop the analytical skills through designing/choosing the most suitable data structure in solving a problem | **A**  **B** | **15%**  **25%** |
| 1. Develop the programming skills through implementing the appropriate data structures for solving computing problems | **C** | **30%** |
| 1. Develop the “Object Oriented” programming skills through applying the various data structures using the object-oriented approach | **I** | **30%** |
| 1. Explore real-world examples that use a certain data structure. | **B** | **10%** |

**Computing Resources and Software:**

* **Language**: C/ C++

**Tentative Course Schedule:**

|  |  |
| --- | --- |
| Week 1 | Introduction, primitive vs non-primitive data structure  Revision of object-oriented concepts  --------------------------------------------------------------------------------------------  Function template  Class Template  Using the string Class in C++ |
| Week 2 | - Generic Vector Class implementation  - STL vector example |
| Week 3 | - Linear Data structure definition  - Array vs Linked list  - Introduction to singly linked lists ADT (motivation + description)  - Implementing Generic Singly linked list and its operations (Add node in Front/back/inorder, delete, display, sort, search) |
| Week 4 | - Doubly linked list ADT (Motivation + description)  - Implementing doubly linked list and its operations  - Circular Linked Lists |
| Week 5 | - Implementing stack using arrays  - Implementing stack using linked lists |
| Week 6 | - Practical example on stacks: Converting from infix to postfix, evaluation of postfix expressions using stacks |
| Week 7 | First exam  - Introduction to Queue ADT and applications  - Circular queue  - Implementing circular queue using array  - Implementing queue using linked list  - Priority-Queue ADT and applications  - Deque ADT and applications |
| Week 8 | - More Queue exercises  ---------------------------------------------------------------------------------------------  - Recursive Definition and Recursive Functions  ---------------------------------------------------------------------------------------------  - Tracing a Recursive Function  - Types of Recursion |
| Week9 | - Recursion Trees  - Recursion vs. Iteration  - Efficiency of Recursive Algorithms  -------------------------------------------------------------------------------------------- |
| Week 10 | - Introduction to Computational Complexity  - Big O Notations  - Introduction to tree ADT and application |
| Week 11 | - Complete tree vs Full tree  - Tree Traversal Algorithms  - Binary Search Trees  - Searching a Binary Search Tree |
| Week 12 | * deletion and insertion into a binary search tree + other BST operations |
| Weeks13 | - AVL tree |
| Week 14 | * Hash ADT and application * Hash ADT implementation * Hash ADT implementation continue |
| Week 15 | * Hash : Collision handling techniques * Graph ADT as data storage * Types of graphs   Cyclic  Acyclic  Directed  In-directed   * Graph Traversal |
| Exam Week: | * **Final Exam** |

**Grading:**

|  |  |
| --- | --- |
| **Assessment Criteria** | **Percentage (%)** |
| First Exam | 20 % – 25 % |
| Second Exam | 20 % – 25 % |
| Final Exam | 40% |
| Labs + Assignments | 15% |
| Quizzes | 1% - 5% |

**Attendance and Participation:**

Regular attendance and active participation is expected from all students in this course. Active participation in the class means being on time, being prepared, listening to others, contributing ideas of your own and asking questions as they come up.

**Programming Assignments & class Notes:**

Assignments, announcements, class notes, and other material will be available on Moodle.

Students are responsible for checking Moodle frequently.

You may discuss the requirements and strategies of a programming assignment with others, but you *should not* look at code belonging to anyone else or make your code available to others. If you have code-specific questions you should address them to the course TA.