|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Identification** | | | **Department** | Computer Science | | |
| **Program** | Graduate | | |
| **Subject** | **Advanced Data Structure** **(3 credits)** | | |
| **Term** | Fall 2017 | | |
| **Instructor** | **PhD, Associate Professor Leyla Muradkhanli** | | |
| **Classroom/hours** |  | | |
| **Prerequisites** | | | **Programming Languages** | | | |
| **Language** | | | English | | | |
| **Compulsory/Elective** | | | Major | | | |
| **Text books and course materials** | | | **Textbook**  **Data Structures and Algorithm Analysis in C++ by Mark Allen Weiss,**  **4th edition, Pearson, 2014.** | | | |
| **Teaching methods** | | | **Case analysis** | | |  |
| **Group discussion** | | | **x** |
| **Lab** | | | **x** |
| **Lecture** | | | **x** |
| **Course paper** | | | **x** |
| **Others** | | |  |
| **Evaluation Criteria** | | | **Methods** | | **Date/deadlines** | **Percentage (%)** |
| **Midterm Exam** | |  | **30%** |
| **Case studies** | |  |  |
| **Class Participation** | |  |  |
| **Quizzes** | |  | **10%** |
| **Project** | |  | **10%** |
| **Presentation** | |  |  |
| **Laboratory Work (Assignments)** | |  | **15%** |
| **Final Exam** | |  | **35%** |
| **Other** | |  |  |
| **Total** | |  | **100%** |
| **Course objectives** | | | This is an advanced course in data structures. The main objective of this course is to provide students with the background to fundamental design, analysis, and implementation of data structures and algorithms.  This course specifically has the following objectives:   * The analysis and evaluation of the data structure needs of particular problems; * The design, analysis, and implementation of C++ programs by using data structures and algorithms. | | | |
| **Learning outcomes** | | | Upon successfully completed this course, the student will be able to:   * Apply advance C ++ programming techniques to developing solutions for particular problems; * Design and implement abstract data types such as linked list, stack, queue and tree by using C++ as the programming language using static or dynamic implementations; * Analyse, evaluate and choose appropriate abstract data types and algorithms to solve particular problems; * Design and implement C++ programs that apply abstract data types. | | | |
| **Course outline** | | | Abstract Data Types. List. Stack. Queue. Trees. Hashing. Priority Queues (Heaps). Sorting. The Disjoint Sets Class. Graph Algorithms.  Algorithm Design Techniques. Amortized Analysis. Advanced Data Structures and Implementation. | | | |
| **Tentative Schedule** | | | | | | |
| **Week** | **Date** | **Topics** | | | | **Textbook/Assignments** |
| 1 | 16.09.17 | **Course Overview and Introduction.** | | | | Presentation  Chapter 1 |
| 2 | 23.09.17 | **Algorithm Analysis**  Abstract Data Types (ADTs) | | | | Chapter 2 |
| 3 | 30.09.17 | **The List ADT**  Simple Array Implementation of Lists  Simple Linked Lists  vector and list in the STL  Implementation of vector  Implementation of list | | | | Chapter 3 |
| 4 | 07.10.17 | **The Stack ADT**  Stack Model  Implementation of Stacks  Applications | | | | Chapter 3 |
| 5 | 14.10.17 | **The Queue ADT**  Queue Model  Array Implementation of Queues  Applications of Queues | | | | Chapter 3 |
| 6 | 21.10.17 | **Trees**  Implementation of Trees  Tree Traversals with an Application  Binary Trees  Implementation  The Search Tree ADT—Binary Search Trees | | | | Chapter 4 |
| 7 | 28.10.17 | **Hashing**  General Idea  Hash Function  Separate Chaining  Hash Tables without Linked Lists  Rehashing  Hash Tables in the Standard Library  Hash Tables with Worst-Case *O*(1) Access  Universal Hashing  Extendible Hashing | | | | Chapter 5 |
| 8 | 04.11.17 | **Midterm exam** | | | |  |
| 9 | 11.11.17 | **Priority Queues (Heaps)**  Model  Simple Implementations  Binary Heap  Applications of Priority Queues | | | | Chapter 6 |
| 10 | 18.11.17 | **Sorting**  Insertion Sort  Implementation of Insertion Sort  Shellsort  Heapsort  Mergesort  Quicksort  External Sorting | | | | Chapter 7 |
| 11 | 25.11.17 | **The Disjoint Sets Class**  Equivalence Relations  The Dynamic Equivalence Problem  Basic Data Structure  Smart Union Algorithms  Path Compression  Worst Case for Union-by-Rank and Path Compression  An Application | | | | Chapter 8 |
| 12 | 02.12.17 | **Graph Algorithms**  Definitions  Representation of Graphs  Topological Sort  Shortest-Path Algorithms  Network Flow Problems  Minimum Spanning Tree  Applications of Depth-First Search | | | | Chapter 9 |
| 13 | 09.12.17 | **Algorithm Design Techniques**  Greedy Algorithms  Divide and Conquer  Dynamic Programming  Randomized Algorithms  Backtracking Algorithms | | | | Chapter 10 |
| 14 | 16.12.17 | **Amortized Analysis**  An Unrelated Puzzle  Binomial Queues  Skew Heaps  Fibonacci Heaps  Splay Trees | | | | Chapter 11 |
| 15 | 23.12.17 | **Advanced Data Structures and Implementation**  Top-Down Splay Trees  Red-Black Trees  Treaps  Suffix Arrays and Suffix Trees  k-d Trees  Pairing Heaps | | | | Chapter 12 |
|  |  | **Final exam** | | | |  |