|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course Syllabus** | | | | | | | | | | | | |
| 1. **Course Information** | | | | | | | | | | | | |
| **Course Title** | | **Data Structures and Algorithms** | | | | | | | | | | |
| **Course Code** | | SWEG 3103 | | | | | | | | | | |
| **Credit Hrs.** | | 4 | | | | | | | | | | |
| **Pre-requisite(s)** | | Fundamentals of Programming I & II | | | | | | | | | | |
| **Target Group:** | | 3rd year SE | | | | | | | | | | |
| **Academic Year:** | | 2021/2022 | | | | | | | | | | |
| **Semester:** | | I | | | | | | | | | | |
| 1. **Objective of the course** | | | | | | | | | | | | |
| To enable students,   * use the concepts related to data structures and algorithms to solve real world problems * practice Recursion, Sorting, and Searching on the different data structures * implement the data structures with a chosen programming language * understand advanced sorting algorithms * work with trees and graphs | | | | | | | | | | | | |
| 1. **Course Description** | | | | | | | | | | | | |
| Basically, the course covers fundamental data structures and algorithms that are common in computer science/ software engineering. A computer program is nothing but data structures plus algorithms. Writing efficient software requires selecting efficient data structures and algorithms that are appropriate for the specific problem domain.  The course covers basic algorithmic techniques and ideas for computational problems arising frequently in practical applications: sorting and searching, divide and conquer, greedy algorithms, dynamic programming. We will learn a lot of theory: how to sort data and how it helps for searching; how to break a large problem into pieces and solve them recursively; when it makes sense to proceed greedily. You will practice solving computational problems, designing new algorithms, and implementing solutions efficiently.  A good algorithm usually comes together with a set of good data structures that allow the algorithm to manipulate the data efficiently. In this course, we consider the common data structures that are used in various computational problems. You will learn how these data structures are implemented in C++ programming language and will practice implementing them in our programming assignments. This will help you to understand what is going on inside a particular built-in implementation of a data structure and what to expect from it. You will also learn typical use cases for these data structures.  **Note**: For implementation purposes in lab practices, we will formally use C++ programming language. Therefore, it’s good to revise your C++ programming language skills ahead of lab sessions. But you are not limited only to C++, you can practice implementing with other programming languages, like Java or Python, in your own. | | | | | | | | | | | | |
| 1. **Learning Outcomes** | | | | | | | | | | | | |
| * State how data is organized in a computer, how it can be retrieved, and how it can be used * Explain the basic techniques for the design and analysis of efficient algorithms * Determine complexity, efficiency of searching and sorting algorithms using Empirical and Theoretical analysis * Determine when and how to use various data structures including linked lists, stacks, queues, binary trees, search | | | | | | | | | | | | |
| * trees and graphs | | | | | | | | | | |
| **4.1. Knowledge** | | | | | | | | | | |
| * Understand how data is organized in a computer, how it can be retrieved, and how it can be used * Explain the basic techniques for the design and analysis of efficient algorithms | | | | | | | | | | |
| **4.2. Intellectual and Practical skills** | | | | | | | | | | |
| * Determine complexity of algorithms using Empirical analysis and Theoretical analysis * Compare and contrast the efficiency of sorting algorithms in sorting a given list * Compare and contrast the efficiency of searching algorithms in searching an item from a list of items * Determine when and how to use various data structures including linked lists, stacks, queues, binary trees, search trees and graphs * Compare alternative implementations of data structures with respect to performance * Apply data structures and algorithms that are frequently used in information processing | | | | | | | | | | |
| **4.3. Attitude and behavior** | | | | | | | | | | |
| To enable students think first problem understanding in any problem solving process and, then take efficiency as key priority to any  programming solution. | | | | | | | | | | |
| 1. **Course outline** | | | | | | | | | | |
| **Wee k** |  | | **Topics and Subtopics** | | |  | | | | |
| 1 | **Lecture 1** | | Introduction: Overview of data structures and algorithms | | |  | |  | |  |
| **Lab 1** | | Revision 1 on C++ programming | | |
| 2 | **Lecture 2** | | Analysis of Algorithms | | |  | |  | |  |
| **Lab 2** | | Revision 2 on C++ programming | | |
| 3 | **Lecture 3** | | Simple sorting algorithms   * Classifications of algorithms * Sorting algorithms | | |  | |  | |  |
| **Lab 3** | | Implementing simple  sorting algorithms in C++ | | |
| 4 | **Lecture 4** | | Simple searching algorithms | | |  | |  | |  |
| **Lab 4** | | Implementing simple searching algorithms in C++ | | |
| 5 | **Lecture 5** | | Linked Lists   * Singly linked lists * Doubly linked lists | | |  | |  | |  |
| **Lab 5** | | Implementing linked lists in C++ | | |
| 6 | **Lecture 6** | | **S**tacks | | |  | |  | |  |
| **Lab 6** | | Implementing stacks in C++ | | |
| 7 | **Lecture 7** | | Queues | | |  | |  | |  |
| **Lab 7** | | Implementing queues in C++ | | |
| 8 | **Lecture 8** | | Trees (Part I) | | |  | |  | |  |
| **Lab 8** | | Implementing trees in C++ (Part I) | | |
| 9 | **Lecture 9** | | Trees (Part II) | | |  | |  | |  |
| **Lab 9** | | Implementing trees in C++ (Part II) | | |
| 10 | **Lecture 10** | | Graphs (Part I) | | |  | |  | |  |
| **Lab 10** | | Implementing graphs in C++ (Part I) | | |
| 11 | **Lecture 11** | | Graphs (Part II) | | |  | |  | |  |
| **Lab 11** | | Implementing graphs in C++ (Part II) | | |
| 12 | **Lecture 12** | | Advanced sorting algorithms | | |  | |  | |  |
| **Lab 12** | | Implementing advanced sorting in C++ | | |
| 13 | **Lecture 13** | | Advanced searching algorithms | | |  | |  | |  |
| **Lab 13** | | Implementing advanced searching in C++ | | |
| 14 | **Lecture 14** | | Hashing | | |  | |  | |  |
| **Lab 14** | | Implementing hashes in C++ | | |
| 15 &16 | **Final Exam weeks** | | | | | | |  | | |
| 1. **Textbook** | | | | | | | | | | | |
| CLIFFORD, SHAFFER AA. "Introduction to Data Structures and Algorithm Analysis." (2013). | | | | | | | | | | | |
| 1. **Reference** | | | | | | | | | | | |
| * Weiss, Mark. Data Structures and Algorithms Analysis in C++. (2006) * E. Horowitz, S.Sahni and Dinesh Mehta. Fundamentals of data structures in C++ (2004) | | | | | | | | | | | |
| 1. **Method of Instruction** | | | | | | | | | | | |
| Class lectures | | | | **3 Contact hour per week**   * Active learning (involves the full participation of students) * Teach inductively and to be followed by deductive assertions | | | | | | | |
| Study of text book | | | | * This is fully the responsibility of the learner | | | | | | | |
| Group Assignment | | | | * Work in groups in not more than 4 students per group * Recognize & evaluate individual contribution | | | | | | | |
| Individual Assignment | | | | * Each student is given to separate question by instructor. * Student will prepare report or submit present it and evaluated by the instructor. | | | | | | | |
| 1. **Grading** | | | | | | | | | | | |
| **Type** | | | | | **Weight** | |  | |  | | |
| Individual assignment (theoretical problem) | | | | | 10 | |  | |  | | |
| Test 1 | | | | | 15 | |  | |  | | |
| Group assignment (Real world problem,  Programming) | | | | | 10 | |  | |  | | |
| Group Project work (Real world problem, Programming) | | | | | 15 | |  | |  | | |
| Final Exam | | | | | 50 | |  | |  | | |
| **Total: 100** | |
| 1. Course policies | | | | | | | | | | | |
| * You ***must*** read the textbook (ahead of and/after) the class. * **Academic dishonesty**: *Plagiarism* is serious offense and might result in course failure. * **Collaboration**: On working assignments, you can collaborate with others to understand concepts but the actual problem should be solved by you organized in your own way. * **Attendance**: Students who fail to attend more than 15% of the classes will get F. * **Dressing code**: You should respect social norms and values. | | | | | | | | | | | |

|  |
| --- |
| * **Cheating**: zero tolerance on cheating exams, serious measures will follow. * **Mobile**: Make silent, no chatting. * **Time**: Don’t be late; try to arrive 3 min before class. If you arrive after class has started, don’t knock just go back. * **Classroom**: Don’t talk, raise your hand if you have questions * **Participation**: Ask questions and respond when asked, even if you don’t know. Say, I don’t know! * **Lab**: Practice by your own, don’t copy, one person/PC, but you can share experiences. * **Grading System:** As determined by the universities legislation. |
| 1. **Due date:** |
| All assignments must be submitted in the class on the due date for full credit. No assignment will be accepted  after class on the due date. |
| 1. **Class room Behavior:** |
| Anything that disturbs your instructor or your colleagues during the class period is considered a troublesome behavior. Examples include: Using mobiles, PDA, making offensive remarks, sleeping, working on  assignments related to other courses, etc. troublesome behaviors are completely prohibited. |