Course Title	Data Structures				
Course Code	ACSC288				
Course Type	BSc Computer Science: Required Course BSc Computer Engineering: Required Course				
Level	BSc (Level 1)				
Year / Semester	2nd year / 3rd semester				
Teacher's Name	Dr. Achilleas Achilleos				
ECTS	5	Lectures / week	2	Laboratories/week	2
Course Purpose	The aim of this course is to provide students with an in-depth understanding of the importance of data structures in the development of programs, as well as strong familiarity with the development and usage of such concepts. The concepts include static and dynamic data structures, static and dynamic memory allocation, linear data structures (array, linked lists, stacks, queues), non-linear data structures (trees, binary trees, binary search trees, heap trees), searching and sorting algorithms and basic overview of algorithmic complexity. The course will focus on the acquisition of practical programming skills using low-level concepts as well as conceptual understanding of how such choices affect program performance.				
Learning Outcomes	Upon successful completion of the course students will be able to: • Recognize the limitations of static data structures, compare and discuss the differences in memory allocation of static and dynamic data. • Construct programs using both static and dynamic memory allocation. • Describe linear data structures (lists – single/double linked, circular, stacks and queues) and explain under which scenarios should be used. • Apply linear data structures programmatically to solve real problems. • Explain how tree data structures can be used, discriminate between the different tree types (generic, binary) and identify where they can be used. • Evaluate traversal methods and be able to design and construct core tree operations using recursive functions. • Examine implementation approaches for special tree structures such as priority queues. • Describe data structure classes available in the C++ Standard Template Library (STL) and apply them for solving relevant problems. • Develop the necessary conceptual understanding that would help				

<table>
<tr><td></td><td colspan="3">adapt to similar programmatic environments (e.g. Collections in Java).
• Recognize the importance of algorithmic complexity and understand complexity of basic algorithms and the concept of big O notation.
• Select, experiment, and develop appropriate data structures and algorithms for searching and sorting problems and judge the advantages.
The above will be applied in practice using the C++ programming language.</td></tr>
<tr><td>Prerequisites</td><td>ACSC183.</td><td>Corequisites</td><td>None.</td></tr>
<tr><td>Course Content</td><td colspan="3">1. Data Types and representation under imperative programming (1 Week)
- Abstract Data Types (ADT), data representation and data storage. Static data types. Memory requirements and implications. Data Types. Static and Dynamic Memory. ADT Structure and Memory Allocation. Unions.
2. Dynamic Data and pointer programming (2 Weeks)
- Pointer programming – Pointers, Addresses and memory management, Pointers as function arguments, Pointers and Arrays. Multi-dimensional Arrays. Multi-dimensional vs Pointer Arrays. Dynamic Data Structures.
3. Classes and the Standard Template Library (2 Weeks)
- Classes and Class Members. Constructors. Overloading Constructors. Pointers to Classes. Templates for Abstraction. The Standard Template Library (STL). STL – Standard Containers. Construction of dynamic data structures in C - new and delete operators. Exception Handling.
4. Linear Abstract Data Types (4 Weeks)
- Non-Recursive Functions. Recursion. ADT List – The Stack and Queue. Implementations and performance considerations. Applications of stacks and queues. Linked lists. List operations (append, remove, etc.). Implementation of linked lists and performance issues. Real Applications of Linked Lists. Types of linked lists: circular linked lists, double linked lists.
5. Non-Linear Data Structures (4 Weeks)
- Understanding differences of Linear and Non-Linear Data Structures. Trees: Implementation of generic trees and applications. Operation on Trees. Tree Traversal Methods. Binary trees and implementations. Special binary trees: Binary search trees, Heaps.</td></tr>
<tr><td>Teaching Methodology</td><td colspan="3">The methodology used to conduct the course is structured mainly around lectures and laboratory examples/exercises, so that students gain theoretical knowledge as well as practical skills. The taught part of course is delivered to the students with the help of computer presentations. Furthermore, the principles are demonstrated by</td></tr>
</table>

	means of specific examples that help students engage with solving problems programmatically. Lecture notes in the form of presentations and code are available through the e-learning system for students to use in combination with the recommended textbooks and references. Lectures are supplemented with computer laboratories that include rigorous demonstrations of taught concepts and experimentation with the C++ programming language. Additionally, during laboratory sessions, students apply their gained knowledge and identify the principles taught in the lectures by means of working on different tasks and problems. Students are also allocated exercises during the laboratories to improve both their individual skills and team work. Eight of these exercises are submitted at the end of the laboratory sessions. Moreover, two assignments are assigned to students to further examine their practical capabilities in applying the taught technologies. Finally, the course assessment is completed through a three-hours final exam at the end of the semester.
Bibliography	**Textbooks:** 1. Mark Allen Weiss, “Data Structures and Algorithm Analysis in C++”, Fourth Edition, 2014. 2. Thomas H. Cormen and Charles E. Leiserson, “Introduction to Algorithms”, Third Edition (MIT Press), Jul 31, 2009. **References:** 1. D. Dicheva, A. Hodge, C. Dichev, and K. Irwin, “On the Design of an Educational Game for a Data Structures Course”, December 2016, DOI: 10.1109/TALE.2016.7851763, IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE). 2. C++ reference – Data Structures – http://en.cppreference.com/w/ 3. Learn C++ – Programming – http://www.learncpp.com/
Assessment	• Eight Laboratory exercises for a total of: 20% • Two Assignments for a total of: 20% • A three-hours Final Exam: 60%
Language	English.