UNIVERSITY of NICOSIA

# Course Syllabus

| Course Code | Course Title | ECTS Credits |
|---|---|---|
| COMP-211 | Data Structures | 6 |
| **Prerequisites** | **Department** | **Semester** |
| COMP-113, MATH-101 | Computer Science | Fall, Spring |
| **Type of Course** | **Field** | **Language of Instruction** |
| Required | Computer Science | English/Greek |
| **Level of Course** | **Lecturer(s)** | **Year of Study** |
| 1st Cycle | Andreas Savva | 2nd |
| **Mode of Delivery** | **Work Placement** | **Corequisites** |
| Face-to-face | N/A | None |

**Course Objectives:**

The main objectives of the course are to:

- Introduce students to Abstract Data Types (ADT).
- Provide practical experience to advanced programming techniques and data structures including tables, linked lists, queues and stacks.
- Introduce students to advance recursion such as the divide-and-conquer and backtracking.
- Obtain a foundation that will allow students to use storage media; methods of representing structured data; and techniques for operating on data structures.
- Introduce students to searching and sorting algorithms.
- Introduce students to Binary Trees and graphs.

**Learning Outcomes:**

After completion of the course students are expected to be able to:

- be proficient in developing high quality source code.
- Discuss the use of primitive data types and build-in data structures.
- Describe common applications for different data structures.
- Implement user-defined data structures in a high-level language.
- Compare alternative implementations of data structures with respect to performance.
- Recognize when and how to use the following data structures: Arrays, Linked lists, Stacks, Queues and Binary trees.

- Compare and contrast the costs and benefits of dynamic and static data structure implementation.
- Choose the appropriate data structure for modeling a given problem.
- Describe the concept of recursion and give examples of its use.
- Describe the divide-and-conquer and backtracking approaches.
- Compare iterative and recursive solutions and determine when a recursive solution is appropriate for a problem.
- Apply various sorting and searching algorithms.

**Course Content:**

1. Programming Principles
   - Programming style
   - Coding, testing and further refinement
   - Program maintenance
   - Abstract Data Types (ADT)
2. Stacks - Arrays
   - Stack specifications
   - Implementation of Stacks
   - Application of Stacks
   - The Standard Template Library (STL)
3. Queues – Arrays
   - Specification of Queues
   - Implementation of Queues
   - Circular Queues
   - Application of Queues
4. Linked Stacks and Queues
   - Pointers
   - Linked Stacks
   - Safeguards
   - Linked queues
5. Recursion
   - Principles of recursion
   - Tree of subprogram calls
   - Divide-and-Conquer
   - Backtracking
   - Tree structure programs: Look-Ahead in Games
6. Linked Lists
   - List Definition
   - Implementation of Lists
   - Double Linked lists

- Circular Linked lists

7. Binary trees
   - Definition of Binary Trees
   - Traversal of Binary Trees (preorder, inorder, postorder)
   - Linked implementation of Binary Trees
   - Binary Search Trees – Insertion, Removal, Treesort
   - Breadth-First Vs Depth-First Traversal
8. Sorting
   - Insertion sort
   - Bubble sort
   - Quick sort
9. Searching
   - Sequential search
   - Binary search
   - Comparison Trees
10. Introduction to Graphs

**Learning Activities and Teaching Methods:**

Lectures, in-class exercises

**Assessment Methods:**

Homework, Assignments, Mid-Term, Final Exam

**Required Textbooks / Readings:**

| Title | Author(s) | Publisher | Year | ISBN |
|---|---|---|---|---|
| C++ Plus Data Structures, 6th ed. | Nell Dale, Chip Weems, Tim Richards | Jones and Barlett Learning | 2018 | 1284089185 |

**Recommended Textbooks / Readings:**

| Title | Author(s) | Publisher | Year | ISBN |
|---|---|---|---|---|
| Open Data Structures: An Introduction | Pat Morin | au Press, Athabasca University | 2013 | 2291-2614 |