CIEE Global Institute - Copenhagen

Course name:	Computer Science II Data Structures
Course number:	(GI) CMPS 2001 CPDK
Programs offering course:	Copenhagen Open Campus Block
Open Campus track:	STEM and Society
Language of instruction:	English
U.S. semester credits:	3
Contact hours:	45
Term:	Spring 2020

Course Description

The course provides a contemporary foundation in design, analysis and application of data structures and algorithms for computer science students and other disciplines using computer programming. It reviews and builds on basic programming and language tools for algorithm analysis and general design paradigms using C++. Students will use object-oriented programming, including recursion, arrays, lists, stacks, queues, trees, hashing and other operations. Graphing and algorithm design techniques are also covered. Data structures and the algorithms used to manipulate data structures will be applied to solving problems in a broad array of contexts.
.

Learning Objectives

By the end of this course, students will be able to:

- Define and apply elementary and advanced data structures, including arrays, stacks, queues, lists, trees and graphs.
- Write efficient and useful object-based programs using C++
- Evaluate solutions derived from using data structures and algorithms.
- Use data structures and algorithms to solve problems in a wide context.
- Communicate use of data structures and algorithms effectively, including oral, written and visual communication.
- Build a foundation for future computer studies.
- Appreciate the impact of computer programming in modern life, in the United States and their host country.

Course Prerequisites

Computer Science I Programming or similar introductory programming course/experience. Previous experience with C++ is not required.

Methods of Instruction

The course will be taught using lectures, class discussions, lecture activities, reading assignments, problem sets, presentations, program analysis and program writing. In addition, students will local related facilities or other points of interest, conducting interviews with local computer programmers. Students will work individually and in groups on assigned problem sets and programs. Students are expected to read portions of the textbooks and complete problems before lectures. Students will work in groups to present current applications of computer programming in their lives and in the lives of those in the local community.

Assessment and Final Grade

Weekly Exams (Five)	20 %
Problem Sets (Five)	25 %
Program 1	10%
Program 2	10%
Program 3	15%
Participation	20 %
Total	100%

Course Requirements

Weekly Exams
Each week, students will take an exam based upon the previous week's material. These exams will include standard exam formats of True/False, Multiple Choice, Short Answer and Problem Solving. Each exam will take approximately 20-30 minutes.

Problem Sets
Problems from the instructor or textbook will be assigned to individuals or groups. Student solutions to these problems will be collected and discussed in review sessions. The instructor will work through or give solutions to all problems. Similar problems will appear on weekly exams. Assessment for problem sets will be based on timely and correct completion of problems.

Program 1

Groups of 2-3 students will write a program of at least 50 lines, demonstrating basic data structure functions of C++

Program 2
Individual students will write a program of at least 75 lines that demonstrates basic data structures, object-based programming and algorithms.

Program 3
Individual students will write a program of at least 100 lines that demonstrates basic and advanced data structures, object-based programming and algorithms.

Group Project
The class will be divided into groups of 3-5 students. This project will involve using C++ programming to address a practical application of class concepts. Students will use both basic and advanced data structures and algorithm functions, will debug early versions of their programs and rewrite algorithms to increase efficiency. Project ideas must be approved by the instructor and feedback will be given by instructor throughout the project's development.

Group Presentations
Students will deliver a 15-minute presentation of their group project. The presentation will focus on program syntax, the behavior of data structures, efficiency of algorithms and utility of the program to solve a real world problem.

Participation
Each student is required to attend all sessions and to participate actively in class discussions, group presentations and during site visits. Be prepared to read approximately 25,000 words weekly and take extensive notes in lectures. Textbook chapters will be assigned as a reference but all tested material will be in lectures. Attendance is mandatory. Participation grade will reflect attendance as well as active participation in classroom and site visit activities.

Class Attendance
Regular class attendance is required throughout the program, and all unexcused absences will result in a lower participation grade for any affected CIEE course. Due to the intensive schedules for Open Campus programs, unexcused absences that constitute more than 10% of the total course will result in a written warning.

Students who transfer from one CIEE class to another during the add/drop period will not be considered absent from the first session(s) of their new class, provided they were marked present for the first session(s) of their original class. Otherwise,

the absence(s) from the original class carry over to the new class and count against the grade in that class.

For CIEE classes, excessively tardy (over 15 minutes late) students must be marked absent. Attendance policies also apply to any required co-curricular class excursion or event, as well as to Internship, Service Learning, or required field placement. Students who miss class for personal travel, including unforeseen delays that arise as a result of personal travel, will be marked as absent and unexcused. No make-up or re-sit opportunity will be provided.

Attendance policies also apply to any required class excursion, with the exception that some class excursions cannot accommodate any tardiness, and students risk being marked as absent if they fail to be present at the appointed time.

Unexcused absences will lead to the following penalties.

Percentage of Total Course Hours Missed	*Equivalent Number of Open Campus Semester classes*	*Minimum Penalty*
Up to 10%	1 content classes, or up to 2 language classes	Participation graded as per class requirements
10 – 20%	2 content classes, or 3-4 language classes	Participation graded as per class requirements; **written warning**
More than 20%	3 content classes, or 5 language classes	Automatic **course failure**, and possible expulsion

Weekly Schedule

NOTE: this schedule is subject to change at the discretion of the instructor to take advantage of current experiential learning opportunities.

Week 1 General Overview

Session 1.1 Programming

This session will form the introduction to the class; we will outline the course requirements and formal aspects of participation and engagement in class. This will be followed by the introduction to programming, providing basic facts and concepts. Students review related math concepts, including exponents, logarithms, series, modular arithmetic, proof by induction and proof by contradiction. They will briefly review recursion. Finally, students will familiarize themselves with some important features of C++.

Readings: Chapter 1 Programming: A General Overview

Watch: ForestKnight. 2018. 5 Things I Wish I Knew When I Started Programming https://www.youtube.com/watch?v=jfxcJH_OWgE

Week 2 Algorithm Analysis and Simple Data Structure

Session 2.1 Algorithm Analysis

Students define algorithms as a set of instructions to solve a problem. They review basic components of algorithms. Students then consider how to estimate time required for a program or algorithm to run. They explore how to reduce the running time of algorithms without impacting output. They define recursion and discuss its use and misuse. They will work in groups to create very efficient algorithms for simple mathematical computations.

Exam 1 (covers material from Week 1)

Readings: Chapter 2 Algorithm Analysis

Watch: Dojo, C.S. 2018. Data Structures and Algorithms #1 – What are Data Structures? https://www.youtube.com/watch?v=bum_19loj9A

Due: Problem Set 1

Session 2.2 Simple Data Structures

Students will define Abstract Data Types (ADTs) and the three most common ADTS. They then define, explore and use three basic data structures: lists, stacks and queues. Students examine programs to spot when and how these data structures are used. They will analyze and debug programs, and show how to efficiently perform operations on lists. Students introduce themselves to stack and queue ADTs and their uses in implementing recursion, operating systems and algorithm design. Students will work on simple programs in C++ that use these data structures.

Chapter 3 Lists, Stacks and Queues

Watch: Laackmann McDowell, G. 2016. Data Structures: Stacks and Queues. HackerRank https://www.youtube.com/watch?v=wjI1WNcIntg

Due: Problem Set 2

Week 3 Trees and Hash Tables

Session 3.1 Trees

Students explore how trees are used to implement the file system of several popular operating systems and how trees can be used to evaluate arithmetic expressions. They consider the use of trees in operating systems, compiler design and search engines. Students investigate how to use binary search trees to support searching operations in $O(\log N)$ average time, as well as $O(\log N)$ worst-case bounds. They go on to describe other tree types, their attributes and applications.

Quiz 2 (covers material from Week 2)

Readings: Chapter 4 Trees

Watch: Laackmann McDowell, G. 2016. Data Structures: Trees. HackerRank https://www.youtube.com/watch?v=oSWTXtMglKE

Session 3.2 Hash Tables

Students describe the hash table ADT and explore potential applications, including insertions, deletions and constant average time. They investigate several methods for implementing hash tables using hash functions. Students then compare these methods and analyze which are appropriate for different uses. They compare hash tables to binary trees and distinguish the pros and cons of each in different contexts.

Readings: Chapter 5 Hashing

Watch: Laackmann McDowell, G. 2016. Data Structures: Hash Tables. HackerRank https://www.youtube.com/watch?v=shs0KM3wKv8

Session 3.3 Programming Workshop

Students will work individually and in groups to apply concepts covered so far. They will examine existing programs and write their own, with a focus on when different data structures are used and how they impact algorithm efficiency. Students will begin to explore ideas for their group programming project.

Read: Hayes, B. 2015. Cultures of Code. American Scientist 103(1): 10 https://www.americanscientist.org/article/cultures-of-code

Due: Due: Program 1

Week 4 **Priority Queues, Sorting**

Session 4.1 Priority Queues (Heaps)

During this session, students consider the importance of rankings in a queue. They investigate the efficient implementation of the priority queue ADT. They explore different uses of the priority queue in a variety of programming applications. They begin with simple implementations, go on to binary heaps and applications of priority queues. Students use the merge operation to explore d-Heaps, Leftist Heaps and Skew Heaps. They describe binary queue structure, operations and implementation.

Exam 3 (covers material from Week 3)

Readings: Chapter 6 Priority Queues

Watch: Taylor, D.S. 2015. Introduction to Binary Heaps.Algorithms with Attitude. https://www.youtube.com/watch?v=WCm3TqScBM8

Session 4.2 Computer Programming: Local Applications and Communication

Students see that communicating highly technical and complicated computer concepts and related program functions can be a challenge. For this session, students visit a local industry or research facility that uses computer programming extensively. Computer programmers share aspects of their work, its business and social implications, and the importance of communicating their work and deliverables to their clients and society.

Readings: Buna, S. 2017. Hard Coding Concepts Explained with Simple Real-Life Analogies. Free Code Camp.

https://medium.freecodecamp.org/hard-coding-concepts-explained-with-simple-real-life-analogies-280635e98e37

Due:

Session 4.3 Sorting

Students discuss the problem of assorting an array of elements and then begin to apply internal sorting to integers. They find there are several easy algorithms to sort, including insertion sort. Students explore the algorithms Shellsort, mergesort and quicksort. They move on to more complex sorting algorithms. They find that any general purpose algorithm requires $\Omega(N \log N)$ comparisons. Students will emphasize analysis for code optimization.

Readings: Chapter 7 Sorting

Watch: Sorting Algorithms. 2014. CParsons2005. https://www.youtube.com/watch?v=jHPexHsDxwQ

Due: Program 2

Week 5 Disjoint Sets Class, Graphs

Session 5.1 The Disjoint Sets Class

During this session, students define the Disjoint Sets Class data structure. They show how it can maintain disjoint sets and be implemented with minimal coding. They see how to greatly increase its speed with only two simple observations. They describe the equivalence problem and explain how Disjoint Sets Class addresses it. They will analyze its impact on running time and apply it in a demonstration. Students will work in groups on their group programming project and will incorporate Disjoint Sets Class into their program.

Exam 4 (covers material from Week 4)

Readings: Chapter 8 The Disjoint Sets Class

Session 5.2 Graph Algorithms

Students discuss several common problems in graph theory. They investigate how these algorithms are useful in practice but can be too slow without specific attention to data structures. Students will take several real-life problems and convert them to problems on graphs. They use algorithms to solve common graph problems. They demonstrate how data structure choice and use impact running time. Students use depth-first search to solve several complex problems in linear time.

Readings: Chapter 9 Graph Algorithms

Session 5.3 Programming Workshop

Students will work individually and in groups to apply all concepts covered so far. They examine, dug and rewrite existing programs, as well as write their own, with a focus on when different data structures are used and how they impact algorithm efficiency. Students will work on programming their group project.

Read: Robertson, D. and Giunchiglia, F., 2013. Programming the social computer. *Phil. Trans. R. Soc. A*, *371*(1987), p.20120379. https://royalsocietypublishing.org/doi/full/10.1098/rsta.2012.0379

Due: Program 3

Week 6 Advanced Data Structures

Session 6.1 Algorithm Design

Students switch their focus from algorithm implementation to design. They consider five common algorithm types used to solve problems. For each, they investigate the algorithm type's general approach, its use in several example applications and discuss its complexity and efficiency. The algorithm types are: Greedy, Divide and Conquer, Dynamic, Randomized and Backtracking. Students find, when confronted with a problem, they should consider these five algorithm types and use data structures to provide efficient solutions.

Quiz 5 (covers material from Week 5)

Readings: Chapter 10 Algorithm Design Techniques

Session 6.2 Amortized Analysis and Advanced Data Structures

Here, students will analyze running time for several data structures considered earlier. They will consider the worst case running time for sequences of operations. Students define an amortized time bound and their limitations. In doing so, students explore and analyze binomial queue operations, skew heaps, the Fibonacci heap and splay trees. Students finalize their group programs and group program presentations.

Readings: Chapter 11 Amortized Analysis

Session 6.3 Next Steps and Project Presentations

Students compare six data structures for practicality. Special attention is given to non-recursive, top down structure and the use of sentinel nodes. The data structures are the: Top-Down Splay Tree, Red-Black Trees, Treaps, Suffix Arrays and Trees, k-d Trees and Pairing Heaps. Students analyze and compare these, weighing trade-offs between code complexity, ease of deletion, searching and insertion costs.

Readings: Chapter 12 Advanced Data Structures and Implementation

Course Materials

Course Textbook

Weiss, M.A., 2013. *Data structures & algorithm analysis in C++*, 4th edition. Pearson Education.

Readings

Buna, S. 2017. Hard Coding Concepts Explained with Simple Real-Life Analogies. Free Code Camp.

Hayes, B. 2015. Cultures of Code. American Scientist 103(1): 10.

Robertson, D. and Giunchiglia, F., 2013. Programming the social computer. *Phil. Trans. R. Soc. A*, *371*(1987), p.20120379.

Online Resources

Medlock, B. Top 12 Online Resources for Total Coding Beginners. Lifehack https://www.lifehack.org/articles/technology/top-12-online-resources-for-beginners-learning-code.html .