# COMP 352, Summer 2018

# Data Structures and Algorithms

T/H, 18:30-21:00, H 435

May 3rd till June 14th

Stuart Thiel

stuart.thiel@concordia.ca
`http://www.encs.concordia.ca/~sthiel/comp352`
EV 11.411
Office Hours: T/H 15:30-16:30, H 17:15-17:45
(514) 848-2424 x7211
Tutorial AAAF: T/H, 16:30-17:20, H 967
Tutor: Stephanie Kamgnia
Tutorial AAAE: T/H, 17:30-18:20, H 967
Tutor: Shahida Chauhan
Tutorial AAA?: T/H, 17:30-18:20, H H907
Tutor: Monika Sharma
POD: M/W/F, 18:30-22:30, H 821
POD: Monika Sharma

Barring Midterm, Final and Office Hour times, this will likely remain accurate.

**Course Description:** This course is about the fundamental syntax and semantics of programming, data structures and algorithms. This **does not** mean syntax or semantics of a particular language, though you will see that too. This is about the idea of what programming is.

In this course you will learn to create algorithmic solutions to problems, solutions involving data structures & their manipulation; you will learn to reliably implement these algorithmic solutions as correct object-oriented programs in Java; you will learn to work effectively to create, implement and demonstrate a program solution to a challenging problem.
**Prerequisite(s):** COMP 232 or COEN 231; COMP 249 or BCEE 231.
**Organization:** The course is a 3-credit course with 5 hours of lectures scheduled every week. There are also two ~1-hour tutorials every week. You are expected to allocate a considerable amount of time in studying the text and solving the assignments (most of which involve a lot of Java programming). Qualified tutors and a POD will provide you with help should you require it, as you are solving your assignments.
**Credits:** 3

**Text(s):** Data Structures and Algorithm Analysis in Java
**Author(s):** Clifford A. Shaffer;
**url:** `http://people.cs.vt.edu/~shaffer/Book/`

**Optional Text:** Algorithms 4th Edition
**Author(s):** Robert Sedgewick and Kevin Wayne;
**url:** `https://algs4.cs.princeton.edu/home/`

## Course Objectives:

At the completion of this course, students will be able to:

1. Create algorithmic solutions to problems, solutions involving data structures & their manipulation;
2. Faithfully implement these algorithmic solutions as correct object-oriented programs in Java;
3. Work effectively to create, implement and demonstrate a program solution to a challenging problem.

## Grading Scheme:

| | |
|---|---|
| Assignments | 40% |
| Midterm Exam | 25% |
| Final Exam | 35% |

**Letter Grade Distribution:** Grades will be curved around a B, unless the class does abysmally as a whole, then it will be curved around a B-. Two standard deviations below the mean is the threshold to FAIL. Two standard deviations above is a guaranteed A+.

## Course Policies:

- **General**
  - I hate writing code on exams. I hate marking it more. I cannot guarantee that there will be none, but I will sure try to minimize it.
  - Quizzes and exams are closed book, closed notes.
- **Assignments**
  - There are three assignments. The first two are smaller and are worth 5% each, the third and final assignment is larger and will be worth 30%. The last assignment will involve a small design component and more complex programming to tie together what you have learned in the course.
- **Attendance and Absences**
  - I do not care if you show up. It is your responsibility to know what was covered and what you should know. It is an evening course, I understand that this is inconvenient. However, I will try to keep things light, lively and informative, so please show up ready to engage with your classmates and myself.
- **The Midterm**
  - There is one midterm. The purpose of the exam is to test if you (a) have understood the material of the lecture/book; (b) done the assignments yourself and learned the right lessons from the experience. Treat the midterm as a wakeup call. As this is an intensive, there will be no time for a second midterm or a make-up exam. The midterm will consist of multiple-choice, multiple-answer questions.
- **The Final**
  - The final exam covers the whole curriculum. It will consist of multiple-choice, multiple-answer questions. The final has the same evaluation aims as the midterm (see above).

**Tentative Course Outline**:
The weekly coverage might change as it depends on the progress of the class. However, you must keep up with the reading assignments.

| **Day** | **Content** |
|---|---|
| Thu. 3rd May | Preliminaries (1.1, 1.2, 1.4)<br>Algorithm Analysis (Binary Search)<br>**Read:** Shaffer: (Ch. 01 )<br>**Read:** Shaffer: Ch. 03 |
| Mon. 7th May | **Assignment 1 Given** |
| Tue. 8th May | Recursion<br>Arrays, Lists, Stacks, Queues and Dictionaries<br>**Read:** Shaffer: Ch. 02, Section 2.5<br>**Read:** Shaffer: Ch. 04 |
| Thu. 10th May | Internal Sorting: Simple Sorts and Quicksort<br>**Read:** Shaffer: Ch. 07, Sections 7.1-7.3, 7.5 |
| Tue. 15th May | Internal Sorting: Mergesort, Heapsort and Radix Sort<br>**Read:** Shaffer: Ch. 07, Sections 7.4, 7.6, 7.7<br>**Assignment 1 Due** |
| Wed. 16th May | **Assignment 2 Given** |
| Thu. 17th May | Binary Trees, Binary Search Trees<br>**Read:** Shaffer: Ch. 05, Sections 5.1,5.2,5.3,5.4 |
| Tue. 22nd May | Priority Queues, Heaps and Huffman Coding<br>**Read:** Shaffer: Ch. 05, Sections 5.5, 5.6 |
| Thu. 24th May | **Midterm** (before lecture)<br>General Trees<br>**Read:** Shaffer: Ch. 06 |
| Mon. 28th May | **Assignment 2 Due**<br>**Assignment 3 Given** |
| Tue. 29th May | General Trees<br>**Read:** Shaffer: Ch. 06 |
| Thu. 31st Jun | Advanced Trees: AVL Tree<br>Advanced Trees: Splay Tree<br>**Read:** Shaffer: Ch. 13, Sections 13.1, 13.2.1 |
| Tue. 5th Jun | Indexing & Hashing<br>Dictionaries as Maps and Associative Arrays<br>**Read:** Shaffer: Ch. 09<br>**Read:** Shaffer: Ch. 04 |
| Thu. 7th Jun | Graphs<br>**Read:** Shaffer: Ch. 11, Sections 11.1, 11.2, 11.3 |
| Tue. 12th Jun | Shortes-Paths Problems, Minimum-Cost Spanning Trees<br>**Read:** Shaffer: Ch. 11, Sections 11.4, 11.5 |
| Thu. 14th Jun | Final Exam Review |
| Fri. 16th Jun | **Assignment 3 Due** |
| Thu. 21st Jun | Final Exam 19:00-22:00 |

**Plagiarism:**
The most common offense under the Academic Code of Conduct is plagiarism, which the Code defines as "the presentation of the work of another person, in whatever form, as one's own or without proper acknowledgement" (Article 19a). [1]
It's not hard, don't cheat. Don't copy other people's code. Don't copy code off the Internet and pass it off as your own (or for this course, just don't copy code off the Internet).

## Graduate Attributes [2]

As part of either the Computer Science or Software Engineering program curriculum, the content of this course includes material and exercises related to the teaching and evaluation of graduate attributes. Graduate attributes are skills that have been identified by the Canadian Engineering Accreditation Board (CEAB) and the Canadian Information Processing Society (CIPS) as being central to the formation of engineers, computer scientists and information technology professionals. As such, the accreditation criteria for the Software Engineering and Computer Science programs dictate that graduate attributes are taught and evaluated as part of the courses. The following is the list of graduate attributes covered in this course, along with a description of how these attributes are incorporated in the course.

- **A knowledge base for engineering**: Demonstrated competence in university level mathematics, natural sciences, engineering fundamentals, and specialized engineering knowledge appropriate to the program. Knowledge of abstract data types: stacks and queues, trees, priority queues, dictionaries. Data structures: arrays, linked lists, heaps, hash tables, search trees. Design and analysis of algorithms: asymptotic notation, recursive algorithms, searching and sorting, tree traversal, graph algorithms.
- **Problem analysis**: Ability to use appropriate knowledge and skills to identify, analyze, and solve complex engineering problems in order to reach substantiated conclusions. Analyze problems and determine their constraints in order to make a choice as to what data structures and algorithms to use for their implementation.
- **Design**: Ability to design solutions for complex, open-ended engineering problems and to design systems, components or processes that meet specified needs with appropriate attention to health and safety risks, applicable standards, and economic, environmental, cultural and societal considerations. Use and compose appropriate data structures and algorithms to solve a variety of problems.
- **Use of engineering tools**: Ability to create, select, apply, adapt, and extend appropriate techniques, resources, and modern engineering tools to a range of engineering activities, from simple to complex, with an understanding of the associated limitations. Make educated choices as to what data structures and algorithms to use to solve problems following their respective strengths and constraints.

## Learning Objectives

- **Knowledge base**: Demonstrate competence in fundamentals of data structures and algorithms.

[1] `www.concordia.ca\students\academic-integrity\plagiarism.html`, May 4th, 2017
[2] As per the last official COMP 352 outline

- **Problem analysis**: Analyze and state model limitations and elements of uncertainty. Formulate and calculate qualitative and quantitative qualities of the problems inputs and outputs. Estimate computational complexity. Evaluate and pick the most appropriate approach based on relevant criteria.
- **Design**:
  - Critique/evaluate many possible diverse solutions and use techniques to evaluate different solutions with sound arguments related to the problems requirements and constraints. Demonstrate thinking outside the box to create innovative solutions.
  - Develop a system architecture adapted to the systems application context and its requirements and constraints. Development and specification of internal and external software interfaces at different modularity levels. Describe a solution that presents enough details for implementation.
  - Write code according to design. Validate implemented systems against system requirements, specifications and constraints, as well as interface specifications.
- **Use of Engineering tools**: Demonstrate appropriate operational use of tools (e.g. algorithms, abstract data types, data structures, asymptotic complexity analysis) for specific tasks in a laboratory environment.

## Important Notes [3]

1. **One credit** represents, for an average student, a minimum of 45 hours of workload spread across the various academic activities (Source: Article 16.1.2 of the Undergraduate Calendar). For an average student, this suggests **a minimum of 135 hours of workload** for a 3-credit course, including the time spent in lectures, tutorials, laboratories, examinations, and personal and team work.
2. Assignments will consist of a theoretical and a programming part. Each student must **independently** and **separately** prepare and submit her/his assignment.
3. Criteria used in evaluation of assignments:
   - Correctness and Testing: the program should conform to the specification given in the assignment. This includes the proper handling of special cases and error conditions and the providing of correct results. The submitted test cases take into consideration special cases and error conditions.
   - Design: the program should be constructed from coherent, independent, and decoupled functions. A function should usually access only its own parameters and local variables.
   - Style: the program should be general-purpose and well-organized.
   - Documentation and Layout: The documentation should consist of a well-annotated program and clearly formatted output. Helpful identifiers and a clear layout are part of documentation. The documentation should include the description of your design and the algorithm implemented.
   - Efficiency: The program must implement the most appropriate method.

[3]As per the last official COMP 352 outline, modified slightly as we have no marker

- Program-User Interface: The program should be easy to use.

4. Programming assignments: For all programming components of your assignments, you need to use Java version 8. You will be using the same computing facilities and the same computer account you used in previous courses (e.g., COMP 249). If you do not have a computer account, you can obtain it from the help desk at H-960 or EV 07.182. This account will give you access to the laboratories. For more information on CSE Computer accounts please visit the website: http://www.encs.concordia.ca/helpdesk/access.html. If you have your own computer and prefer to use it, you may do so, but be aware that your programs must compile and run with Java 8 at the Concordia laboratories.

5. Submission format: All assignment-related submissions must be adequately archived in a ZIP file using your last name as file name. The submission must contain your name and student ID. Use your "official" name only  no abbreviations or nick names; capitalize the usual last name. Inappropriate submissions will be heavily penalized. Only electronic submissions will be accepted. Students will have to submit their assignments using the EAS system. Assignments must be submitted to the correct folder for assignments. Assignments uploaded to an incorrect folder will not be marked and result in a zero mark. No resubmissions will be allowed. For the Java programming assignments you have to submit the complete source code and the compiled files, which must be executable without changes. If this is violated you will get a zero mark for these parts of the assignments.