CMPT 225 Course Overview

Data Structures

- A course on Data Structures: common ways of organizing computer memory, with algorithms that manipulate this memory.
- We use the Abstract Data Type approach, which goes hand-in-hand with object-oriented programming.
- The computer language we will be using is C++, and there will be a lot of programming, but this is not a programming course in the same way as first-year courses are.

Professor and TAs

Dr. Thomas C Shermer, a.k.a. Tom

TASC-I 8021

shermer@sfu.ca

Office hours: MW15:30-16:30 (or by appointment)

TAs: (find them in the lab)

Ehsan Haghshenas	ehaghshe@sfu.ca
Jeffrey Ens	jeffe@sfu.ca
Junwei Sun	junweis@sfu.ca

Text

Data Structures and Algorithms in C++

by Goodrich, Tamassia, and Mount, 2nd edition.

The text is required.

It's an all-around good text. Quite clear and has good types of examples. As a theoretician and pragmatist, I'm impressed.

(As a software engineer, there are a few things I'd change with the examples, but that's not a big concern at this point.)

Marking

Midterm Exam	15%
Final Exam	35%
Lab Exam	10%
Homework (3)	30%
Lab Exercises	10%

The final covers the entire course (it is cumulative).

Marking Policies

- Laboratories are marked 0 or 1.
- Partial marks are given on exams:
  - If you get the wrong answer but show work that shows some understanding, you will get some marks.
  - If you get the right answer but show work that shows some misunderstanding, you will lose some marks.

- In the event of a marking dispute (you think your mark isn't fair) first contact the marking TA to try to resolve it. If that doesn't resolve it, then bring it to the professor.

Important Dates

May 6	Classes Start
May 16	Labs Start
May 20	No class (Victoria Day)
June 5	Homework 1 due
June 19	In-class midterm
July 1	No class (Canada Day)
July 3	Homework 2 due
July 31	Homework 3 due
Aug 2	Last day of class
Aug 6	Final exam

Assignment Submission

- Assignments must be submitted by 11:59 pm on the due date.
- Assignments are to be submitted on CourSys (coursys.sfu.ca).
- Late penalties are -10% per day, up to 5 days. Days are calendar days—weekends count.
- Assignments submitted after 5 days late will be given a 0.

Course Syllabus

- We will follow the text. You will gain the most benefit by reading ahead of lecture.
- The approximate pace is one chapter per week.
- We will not finish the book, but I do recommend finishing it on your own.
- Chapter 1 is a C++ Primer and I assume you have this knowledge from your prerequisites. Please read Chapter 1 and ensure that you know the material, including the part on pseudo-code.
- Lecture will start with Chapter 2, Object-Oriented Design.

Course Syllabus

- Chapter 2: Object-Oriented Design
- Chapter 3: Arrays, Linked Lists, and Recursion
- Chapter 4: Analysis Tools
- Chapter 5: Stacks, Queues, and Deques
- Chapter 6: List and Iterator ADTs
- Chapter 7: Trees
- Chapter 8: Heaps and Priority Queues
- Chapter 9: Hash Tables, Maps, and Skip Lists

Course Syllabus

- Chapter 10: Search Trees
- Chapter 11: Sorting, Sets, and Selection
- Chapter 12: Strings and Dynamic Programming
- Chapter 13: Graph Algorithms
- Chapter 14: Memory Management and B-Trees

C++ and Java and …

- We use C++ exclusively in this course.
- Each computer language is a tool with its own characteristics, strengths, and weaknesses.
- Don’t argue over whether a hammer or a screwdriver is a better tool. Or C++ or Java.
- C++ is a language designed so that correct programs compile quickly.
- Java is a language designed so that incorrect programs are easy to diagnose.
- Use whichever tool is appropriate for the problem at hand.

Code Style - Comments

- Comment your code. Most student code is undercommented.
- Remove as many comments as possible from your code by making the code say what the comment says.

```
// add today's sales to yearly sales
ytd += sales;

yearToDateSales += dailySales;
```

Code Style - Comments

```
void foo(int* A, int n) {
    …
    // initialize the array A
    for( int i = 0; i < n; i++) {
        …
    }
    …
}
```

```
void foo(int* A, int n) {
    …
    initializeArray(A, n);
    …
}


void initializeArray(int* A, int n) {
    for( int i = 0; i < n; i++) {
        …
    }
}
```

Code Style - Optimization

- Premature Optimization is the root of all evil.
  - Clarity and correctness are often more desirable than speed.
  - When speed is an issue, first write the program clearly and correctly, then determine what code is slowing the program down, and only then optimize that code.

Code Style - Formatting

- Always format your programs consistently.
  - Indentation
  - Blank lines
- In finished work, never leave in commented-out or debugging code.
- Always include braces around a subordinate block:

NO:

```
for(int i=0; i<n; i++)
    sum += A[i];
```

YES:

```
for(int i=0; i<n; i++) {
    sum += A[i];
}
```