

# Computer Science 331

## Introduction to CPSC 331

Mike Jacobson

Department of Computer Science  
University of Calgary

Lecture #1

## Outline

- 1 Course Information
- 2 Learning Goals
  - Overview
  - Abstract Data Types
  - Algorithm Analysis and Testing
- 3 Expected Background
- 4 How to Succeed
- 5 References
- 6 What to do Next

## Course Information

### Course Information

#### Instructor: Mike Jacobson

- Phone: 210-9410, ICT 612
- email: [jacobs@cpsc.ucalgary.ca](mailto:jacobs@cpsc.ucalgary.ca)
- URL: <http://pages.cpsc.ucalgary.ca/~jacobs/>

#### Contact Times:

- Office hours: M 14:00-16:00 or by appointment *only*
- Lectures: MWF 10:00–10:50 in ICT 121
- Tutorial #1: M/W 16:00–16:50 in MS 176
- Tutorial #2: M/W 13:00–13:50 in MS 176
- Tutorial #3: M/W 11:00–11:50 in MS 156
- Tutorial #4: TuTh 9:00 - 9:50 in MS 160

First labs: next Monday

## Course Information

### Assessment

#### Components:

- 30% — four assignments (written and programming questions)
- 30% — midterm (Wednesday, March 8, 17-18:30, EDC 179)
- 40% — final exam

**Take note of term test dates/times:** let me know of conflicts as soon as possible (no make up tests)

#### Submission procedures and guidelines:

- information available on course web site

**NOTE:** a grade of **C-** or better is required to use this course as a prerequisite for any course offered by Computer Science

## Abstract Data Types

This course is centred around *abstract data types* and the *data structures* and *algorithms* used to implement them.

You will learn to:

- describe classical ADTs (stacks, queues, dictionaries, graphs) and their operations
- identify data structures that can be used to implement ADTs (arrays, linked lists, binary search trees, hash tables, ...)
- compare resource requirements for different data structures and algorithms
- describe operations using English, pseudocode, or Java implementation
- apply ADTs to solve problems, using your own code or existing packages

## Algorithm Analysis and Testing

In order to meet these goals, you will also be introduced to certain tools from *algorithm analysis*.

You will learn to:

- develop mathematical expressions for time and storage requirements for simple algorithms
- use asymptotic notation to simplify expressions for resource requirements of algorithms
- understand and correctly identify asymptotic relations between functions that are commonly used to bound resource requirements
- explain - clearly and precisely - why algorithms, discussed in this course, are correct and efficient.
- develop and implement tests to provide evidence that software correctly implements an algorithm

## Algorithms

An **Algorithm**:

- is a *finite sequence of steps that solves some well-defined problem*
- is often given either by several paragraphs in carefully written English or using *pseudocode*. Such a description is (largely) “implementation independent”
- can be *implemented* as (part of) a program using some programming language

**Note:** This course will focus at least as much on *algorithms* as on the computer programs generated from them.

⇒ CPSC 331 is *not a programming course*.

## More About Problems and Algorithms

Many computer science applications rely on solutions to a small number of *fundamental problems*

*Resource requirements and limitations* may also be important — and may differ from application to application

*Consequence:* It is often useful to know about *several* algorithms for the same problem — because there will be situations in which each is a better choice than the others

In this course we will learn about algorithms for several fundamental problems, including *searching* and *sorting*

## Data Types and Abstract Data Types

A *data type* is defined by

- Data values and their representation
- Operations defined on the data values and the implementation of these operations as executable statements (i.e., methods)

An *abstract data type* is, essentially:

- a *specification of requirements* for a data type
- it does not include (or require) a specific implementation — but it may include conditions that data values must satisfy

## Data Structures

A **data structure** provides a representation of the data values specified by an ADT

- Together with **algorithms** for an ADT's operations, this provides an *implementation-dependent* description of a data type

We will study several fundamental ADTs, along with data structures and algorithms for their operations, in this course

## Java Implementation

Assignments will require Java programming. You will

- implement algorithms and data structures on your own
- use implementations in a standard Java library (the “Java Collections Framework”) to solve problems

Java will *not* be taught (much) during the lectures. However, sources of help with Java include

- lots of material on the course web site, textbook
- tutorials, which will include more material about Java programming (some of the time)

## Algorithm Analysis and Testing

**Correctness** and **efficiency** of algorithms are both important!

In this course you will

- see numerous *proofs of correctness* of algorithms, and you will become familiar with the structure of a proof of correctness as a result
- design and implement *tests* in order to look for errors and use the results of tests to debug your programs
- learn ways to measure (and *compare*)
  - the time an algorithm requires in the “worst case”
  - the amount of *storage space*

In this course we will generally prove the correctness and efficiency of *algorithms* but we will test, debug and profile *programs*.

## Expected Background: Programming

### An Object-Oriented Programming Language:

- Java should have been introduced in a prerequisite course
- see Java resources on the course web site or the textbook
- work through Tutorial Exercise #1 as soon as you can! It will be discussed in the first tutorial next week.

### Recursion:

- you should understand how recursive programs can be used to solve problems
- recursive *definitions* of various structures and properties will be used in this course as well

## Expected Background: Other Areas

### Discrete Mathematics and Logic:

- have numerous applications in CPSC 331 (especially proofs and analysis)
- ⇒ MATH 271 is a prerequisite of this course!

### Technical Reading and Writing:

- this course will include **reading assignments**
- your writing **will be assessed** in this course

## How to Succeed

In this course you will **learn by doing!**

- Prepare for and attend **lectures**
  - obtain/read notes and other reading material ahead of time
- Prepare for and attend **tutorials**
  - read and work through exercises ahead of time
  - the more you do *on your own* the better
- Work through the **self-study exercises**
  - will help you learn required aspects of Java for this course
- Take **assignments** seriously
  - start early (not last minute!)
  - make sure that you understand what you are — and what you are *not* — allowed to do when working on these

Make use of my **office hours** if you need more help

## Recommended Reference

### Textbook:

- Thomas H. Cormen, *et.al.*, *Introduction to Algorithms, 3rd Edition*, MIT Press, 2009, eBook via the library (2nd edition)

### Recommended Java Reference:

- Kathy Sierra and Bert Bates, *Head First Java*, O'Reilly, Second Edition, 2005, eBook via the library

### Recommended Reference for Correctness:

- Michael Soltys, *An Introduction to the Analysis of Algorithms*, World Scientific, 2009.

Helpful material on proofs of correctness in Chapter 1 (can download for free from book's website).

## Other Resources

**Course web site:** *lots* of information here!

- Available from the instructor's home page
- D2L will be used for assignment submission and access to grades

**Lectures:** students are expected to attend *all* classes

- Partial notes will be made available online ahead of time
- Additional material on course web site

**Tutorials:** participation in these is expected too!

- Self-exercises and tutorial exercises will be posted on the web site ahead of time

## What to do Next

- 1 Make sure you are eligible to be *in* this course!
- 2 Buy the textbook (will be useful as a reference for a long time!)
- 3 Request your computer science account if you don't already have one
- 4 Work through Self-Study Exercise #1! It will be assumed that you have *completed* this before the first tutorial next Monday
- 5 Then read through Tutorial Exercise #1 and try to answer the questions on it!
- 6 Check out the course web site! It includes lots of information, including about how to accomplish the above