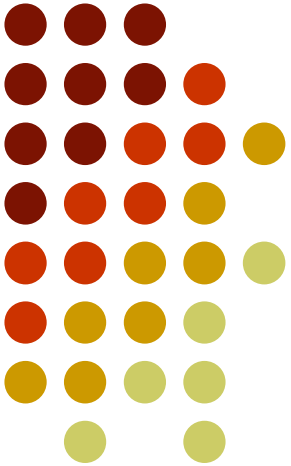
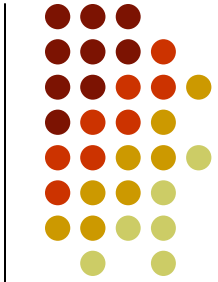
CMPT 225

Data Structures and Programming



Course information



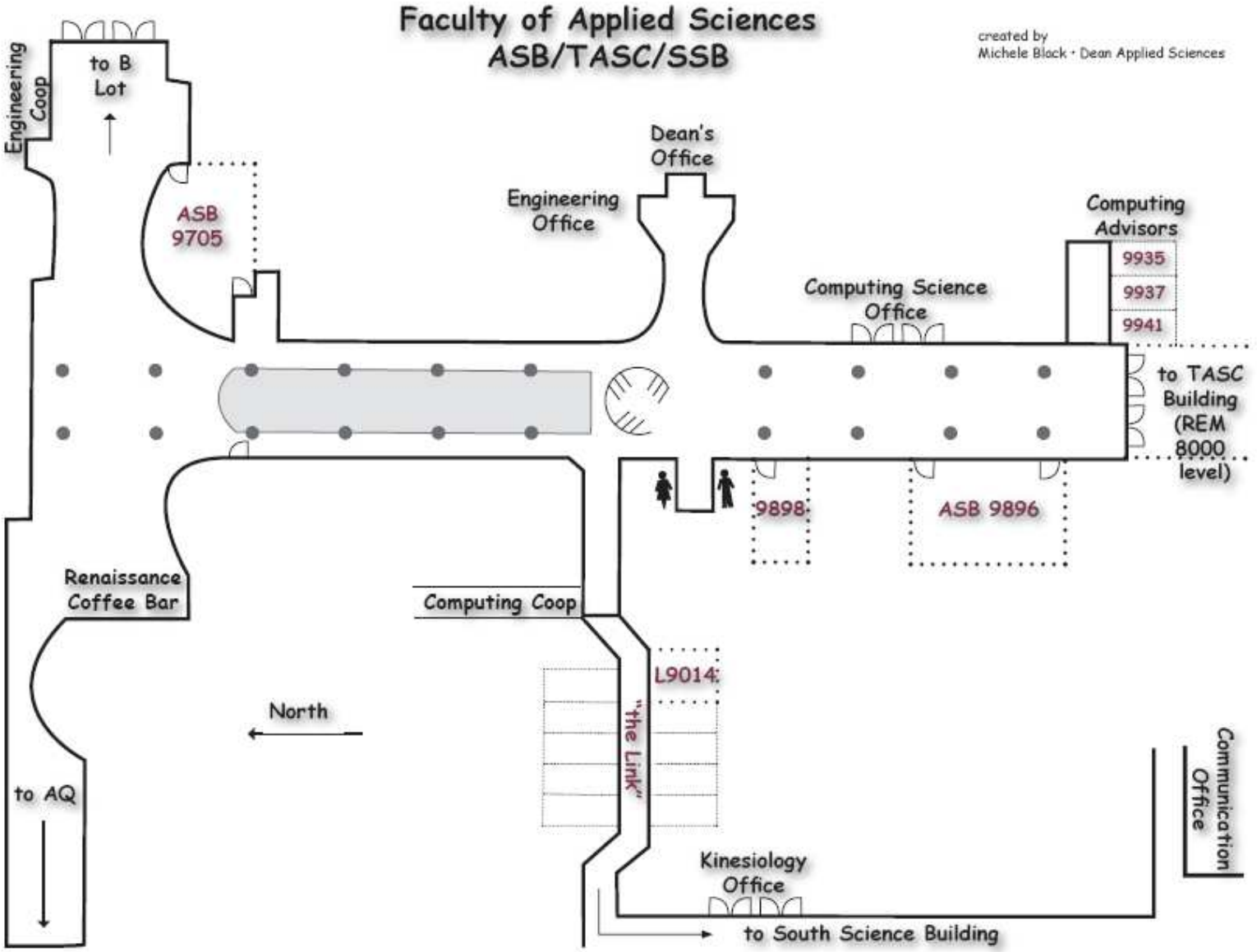
- Lecturer: Jan Manuch (Jano), TASC 9405

  Email: jmanuch@sfu.ca
- TAs:
  - Osama Saleh, osaleh@sfu.ca
  - Maryam Moslemi Naeini, mmoslemi@cs.sfu.ca

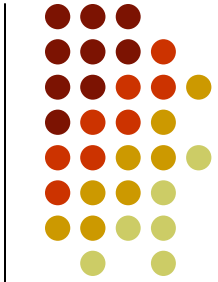
- Course website:

  http://www.cs.sfu.ca/CC/225/jmanuch/
- Grades:

  https://gradebook.cs.sfu.ca/

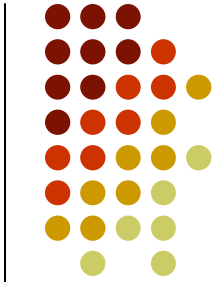

Faculty of Applied Sciences
ASB/TASC/SSB
created by
Michele Black • Dean Applied Sciences
Engineering Coop
to B Lot
ASB 9705
Dean's Office
Engineering Office
Computing Advisors
9935
9937
9941
Computing Science Office
to TASC Building (REM 8000 level)
9898
ASB 9896
Renaissance Coffee Bar
Computing Coop
L9014
"the Link"
North
to AQ
Kinesiology Office
Communication Office
to South Science Building

Course timetable



- Lectures:
  Mon, Wed, Fri: 1.30pm-2.30pm, West Mall 3210
- Midterm: TBA
- Final Exam: TBA
- Office hours:
  - Instructor: (2 hours) Mon, Wed, 2.30pm-3.30pm, T9405
  - TA(labs): (3 hours) TBA

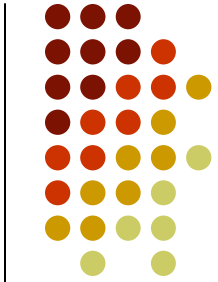
Grading scheme



- Midterm: 25%
- Final: 45%
- Assignments: 30%
  (6 assignments each worth 5%)
  Late assignments: -20% each day

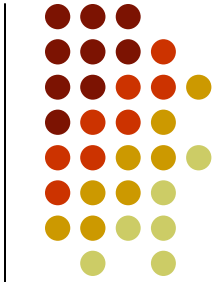
- Shifting weights:
  - It is possible to shift 50% of the midterm weight (that is 12.5% of total weight) to final weight.
  - **No other shifts are possible!**

CMPT 225 Topics



- Software Development Process
  - Software Life Cycle
  - Specification, Design and Testing
  - Decomposition, Abstraction and Encapsulation
- Data Structures and Abstract Data Types
  - Arrays and Linked Lists
  - Stacks, Queues and Priority Queues
  - Trees (Binary, Red-Black, Heaps) and Graphs
  - Hash Tables
- Algorithms
  - O() Notation
  - Recursion
  - Sorting Algorithms

Course Objectives

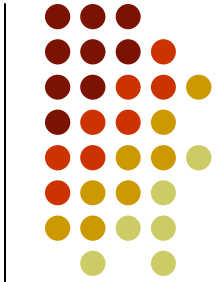


- Develop problem solving techniques
  - Use abstraction to design solutions
  - Design modular programs
  - Use recursion as a problem-solving strategy
- Provide tools for the management of data
  - Identify abstract data types (ADTs)
  - Construct implementations of the ADTs

Problem Solving and Software Engineering

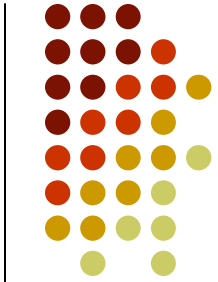
- Coding without a solution design increases debugging time
- A team of programmers is needed for a large software development project
- Teamwork requires:
  - An overall plan
  - Organization
  - Communication
- Software engineering
  - Provides techniques to facilitate the development of computer programs

What is Problem Solving?



- Problem solving
  - The process of taking the statement of a problem and developing a computer program that solves that problem
- A solution consists of:
  - Algorithms
    - Algorithm: a step-by-step specification of a method to solve a problem within a finite amount of time
  - Data structures to store the data and support the algorithms

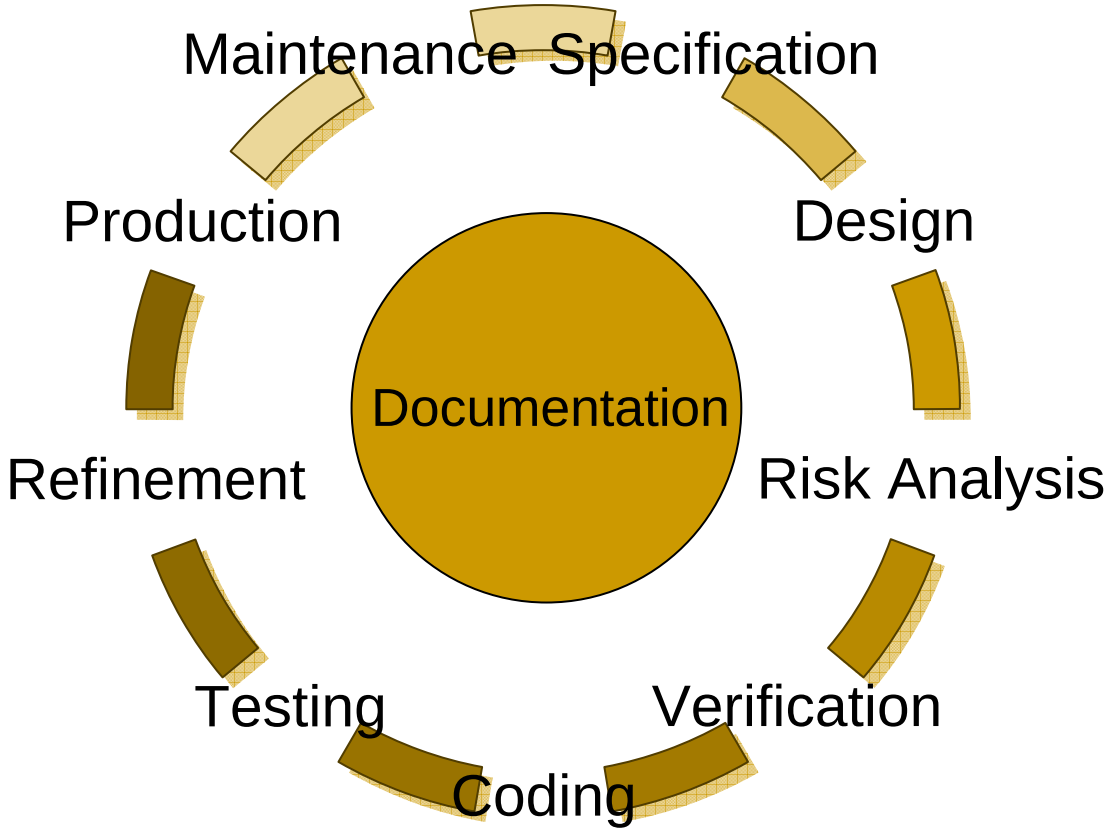
What is a Good Solution?



- A good solution is cost effective
  - To develop and maintain:
    - The total cost it incurs over all phases of the development process is minimal
  - To run (i.e. to perform its task)
    - Difficulties encountered by those who use the program (Interaction cost)
    - Computer resources (time and memory) that the program consumes
    - Consequences of a program that does not behave correctly
- Programs must be well structured and documented

Remark: Efficiency is only one aspect of a solution's cost

Software Life Cycle



Software Life Cycle Phases - 1

- Specification
  - Understand the client's problem and requirements
  - Ensure that the requirements are clear and complete and understood by all parties
- Design
  - Plan the implementation of the application's data and operations
  - Plan the testing
- Risk Analysis
- Verification
  - Ensure that algorithms are correct (methods: invariants, etc.)
  - Ensure that the design satisfies the requirements (validation)
- Implementation
  - Write application and test code

Software Life Cycle Phases - 2

- Testing
  - Verify that code works
  - Verify that the code meets the client's requirements
  - There are various types of testing techniques: unit testing, integration testing, system testing, user acceptance testing
- Refining
- Production
  - Package, distribute and install application and train users
- Maintenance
  - Add features
  - Fix bugs
- Documentation
  - Common to *all* the phases of the life cycle
  - Includes the user manual

Software Life Cycle and CMPT 225

- We are primarily concerned with four phases of the life cycle
  - **Design**
  - Verification
  - **Implementation**
  - Testing