CSC148H
Introduction to Computer Science
(Summer 2009)

Instructor: Robert Danek
(rdanek@cdf.toronto.edu)

Lectures: BA1220 R4-6
What Computer Science Is Not

"Computer Science is no more about computers than astronomy is about telescopes."

Edsger W. Dijkstra
What Computer Science Is Not

PROGRAMMING != COMPUTER SCIENCE
What is Computer Science?

- The study of problems, problem-solving, and the solutions that come out of the problem solving process
Steps to solving a CS problem

- **Specification**
  - Clear, precise descriptions
- **Design**
  - structure your solution carefully
  - employ abstraction
- **Analysis**
  - reason about an algorithm's efficiency and correctness
- **Implementation**
  - implement solution in some language
  - recursion vs. iteration?
  - which data structures to use?
- **Verification**
  - Unit testing
How the customer explained it
How the project leader understood it
How the programmer wrote it
What the customer really needed
Steps to solving a CS problem

- **Specification**
  - Clear, precise descriptions

- **Design**
  - Structure your solution carefully
  - Employ abstraction

- **Analysis**
  - Reason about an algorithm's efficiency and correctness

- **Implementation**
  - Implement solution in some language
  - Recursion vs. iteration?
  - Clean, modular, easy to understand code

- **Verification**
  - Unit testing
  - Write clear docs for tests
Abstraction

• **Abstraction** is an integral part of problem solving
  – Ignore certain details to make the problem easier to solve.
  – The details still need to be dealt with
  – Simplifies the process of problem solving
Abstract Data Types (ADTs)

- Fundamental computer science concept
- **Abstract**: no mention of the implementation
- **Data Type**:  
  1) the data being stored, and  
  2) the operations that can be performed on the data
Steps to solving a CS problem

- **Specification**
  - Clear, precise descriptions

- **Design**
  - structure your solution carefully
  - employ abstraction

- **Analysis**
  - reason about an algorithm's efficiency and correctness

- **Implementation**
  - implement solution in some language
  - Clean, modular, easy to understand code

- **Verification**
  - Unit testing
  - Write clear docs for tests
The Value of Testing

“Beware of bugs in the above code; I have only proved it correct, not tried it.”

Donald Knuth
ADT examples from CSC108/A08H

- **List**
  - Data: a sequence of objects, in order
  - Operations: append, index into, find, ...

- **Dictionary**
  - Data: a collection of key-value pairs
  - Operations: insert pair, lookup value with key, ...

- **Both ideas are abstract, since**
  - no mention of how data is stored in memory
  - how operations are performed
Stack ADT (2.3)

- A sequence of objects.
- Objects are removed in the **opposite** order they are inserted.
- Last-In-First-Out (LIFO)
- Like a stack of plates
- The object last inserted is at the top.

- Operations:
  - **push(o)** Add a new item to the top of the stack
  - **pop()** Remove and return top item
  - **peek()** Return top item
  - **isEmpty()** test if stack is empty
  - **size()** return # of items in stack
Uses For A Stack

- Keep track of pages visited in a browser tab
- Keep track of function calls in a running program
- Check for balanced parentheses
Python Stack Class

- How will we store the data?
- What effect does this decision have on speed?
- Lets explore in Wing.
Queue ADT (2.4)

- A sequence of objects.
- Objects are removed in the same order they are inserted.
- First-In-First-Out (FIFO)
- Like a store line up

Operations:
- `enqueue(o)` Add `o` to the end of the queue
- `dequeue()` Remove and return object at the front of the queue
- `front()` Return object at the front of queue
- `isEmpty()` test if queue is empty
- `size()` return # of items in queue
Uses for a Queue

- Queues are used in operating systems to keep track of processes waiting for a turn to use the CPU

- Graphical User Interfaces (GUIs)
  - Queues keep track of events waiting to be handled, like multiple button clicks
Implementation of a Queue

• Implementation of Queue using Python Lists
Priority Queue ADT

- A sequence of objects.
- Objects are removed in order of their priority.
- Like a line up in a bank where the customer with largest bank account goes to the front.

Operations:
- `insert(o)` Add `o` to the queue
- `extractMin()` Remove and return object with minimum value
- `min()` Return object with min. value
- `isEmpty()` test if queue is empty
- `size()` return # of items in queue
In Closing ...

- We covered the following:
  - Section 1.1-1.3 (What is Computer Science?)
  - Section 2.3 (Stacks), 2.4 (Queues)
  - Python Style Rules
- You may also want to read Section 1.4 if you need a review of Python
- Assignment 1 is now posted. It is due in two weeks.
- Next week: More Stacks and Queues, Exceptions, and OOA/OOD.