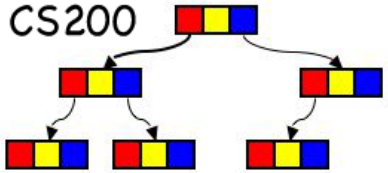


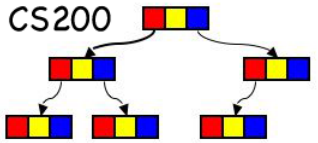
CS200 Spring 2015
Data Structures and Algorithms

Instructor:

Wim Bohm

Graduate Teaching Assistants:

Upulee Kanewala and Rahul Dutta

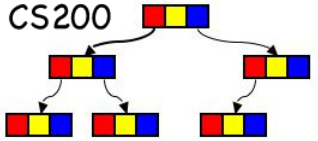


We live in the information age – fueled by computers.
An unprecedented amount of information is freely available.
How many of you have smart phones?
What apps/information do you store, manage and use on a daily basis on that phone.

This course is about the fundamentals of how that information is stored, managed and used
-- the theory and practice of representing and manipulating information

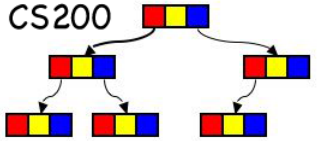
"scientia est potentia"
(knowledge is power)
Sir Francis Bacon or Thomas Hobbes

Class meetings



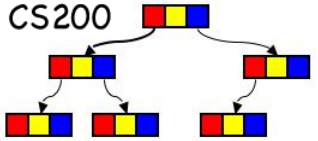
- Lectures
  - Concepts, programming assignment introduction, quizzes, tests.
- Recitation
  - Help with programming and written assignments, practice skills, reinforce/supplement material from lecture, a few programming quizzes.
  - Credit for attending and participating in recitations

Difference from CS160/161



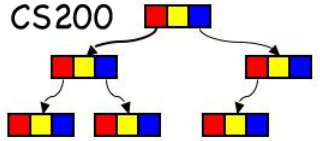
- More freedom in how to structure your program
- Larger program developed in an iterative, incremental manner over a number of assignments

Grading



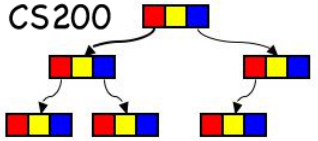
Programming assignments	20%
Written assignments	10%
Quizzes	10%
Recitations	10%
Midterm	20%
Final	30%

More Grading Specifics



- Exams:
  - Make-ups or reschedules for extreme circumstances only
  - Programming component given in lab section during the week of the exam
    - Open text book
    - Access to Java API descriptions, but not open Web!
  - Written component in lecture on specified date
    - Closed book

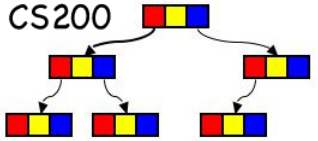
Policies



Be professional. Read the web site on this.

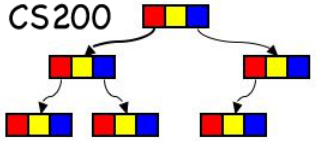
Let's talk about cheating

Cheating



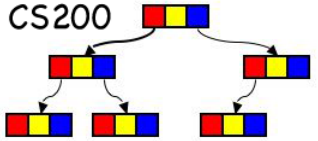
- What is cheating? What is not?
  - **Where would you find a definition?**
- What is gained / lost when cheating?
- What are the consequences?
- When / how does it happen?
  - **How can cheating be avoided?**

Late Policy



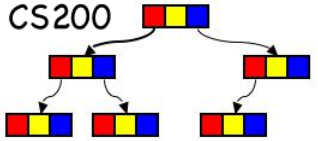
- Programming and Written Assignments
  - By due date/time: full credit
  - Within 48 hours after the deadline: 10% penalty
  - After 48 hours: 0

Distractions in the classroom



- Cell phones
  - Turn off (first choice) or on vibrate
  - If expecting an important call, sit close to the door and step out.
- Laptops & SmartPhones
  - Sit where you will not distract others
  - Do try to limit non-class related activities. Psychological evidence shows that we do not multi-task as well as we think we do.

Communication



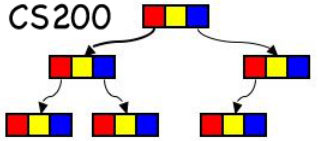
- Check course website often:

  http://www.cs.colostate/~cs200

- Let's go check it out

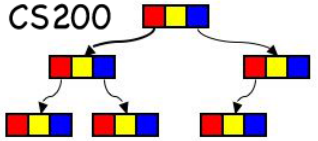
- RamCt will be used minimally
  - to post grades

Course Goals



- CS160: mostly procedural programming, using objects, logic
- CS161: objects, linear data structures, inheritance, induction, counting
- CS200
  - Logical view
    - Program = Algorithms + Data Structures
    - Understand their relationship and use them correctly, efficiently
  - Implementation
    - Program = Objects + Methods
    - Practice design and implementation of object-oriented programs in Java
  - Connect theory to programming concepts, complexity

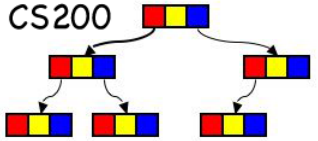
Course Goals



- An understanding of a variety of common data structures
- A practical understanding of where they are applicable
- Understanding the complexity of programs
  - Time complexity: what is the Order of Magnitude time this algorithm takes given an input of size n
  - Space complexity: what is the Order of Magnitude space this algorithm takes given an input of size n

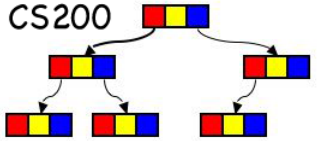
What does order of magnitude mean?

Programming Assignments



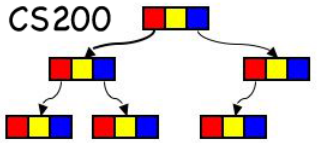
- Based on **expressions** and **assignments**
  - Postfix expressions and evaluation
  - Infix expressions, parsing, representation, evaluation
  - Assignments, symbol tables
  - Analysis: dependences

Language, grammar



- Postfix expressions form a **language**: a set of valid strings ("sentences"), so do infix expression
- In order to manipulate these sentences we need to know which strings are **valid sentences** (belong to the language)
- To define the valid sentences we need a mechanism to construct them: **grammars**
- A grammar defines a set of **valid symbols** and a set of **production rules** to create sentences out of symbols.

Postfix expressions: symbols



- Symbols: integer numbers and operators

  int : digit sequence

- There are many mechanisms to define a digit sequence, e.g. regular grammars (next lecture) or regular expressions:

  dig: "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"

  num: dig^{+}

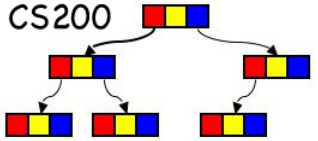
- operator: "+" | "-" | "*" |"/"

| stands for: OR (choice)

+ stands for: 1 or more of these (repetition)

***** don't confuse the META symbols | $^{+}$ with the language symbols "…"**

Postfix expressions



- A postfix expression is

  a number, or

  two postfix expressions followed by an operator

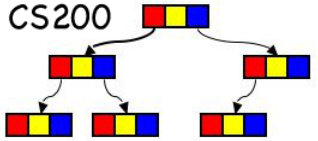
Notice that the operators in this example are **binary**

- The mechanism (context free grammar, next lecture) to describe this needs more than choice and repetition, it also needs to be able to describe (block) structure

  PFE ::= num | PFE PFE operator

Notice that context free grammars are recursive in nature.

Quick check



Which are valid PFEs:

a b +

1 2 3 * +

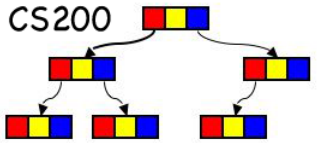
1 2 3 + *

1 2 * +

11 22 – 33 + 44 *

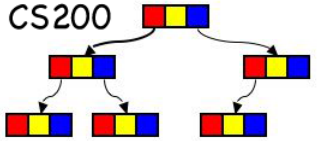
If valid, what is their corresponding infix expression?

Design for Change Principle



- Anticipate how systems will evolve and design to accommodate change.
  - Lack of attention to this principle can result in changes that make system unstructured and difficult to understand and maintain.

Assignment 1



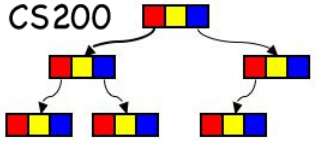
- First step: scan in postfix expressions. In this case the symbols (numbers and operators) are delimited by spaces. One PFE per line.
- Second step: evaluate PFE

  We need a nice data structure, that allows us to manipulate PFEs: a Stack

- What is a stack?

- How would you use it to evaluate PFEs?

Java Scanner Class



- Scanner divides an input stream (e.g., from a file or String) into words separated by delimiters.
- Scanner defines a grammar for syntax of numbers and uses *regular expressions* to define delimiters.

The theory of grammars and regular expressions will be covered in next lectures.