Overview of CS 211

Lecture 1

No permission needed to switch sections
Pick one section and attend it
Sections may cover material not covered in class
Each section will be led by a TA
II Sections

Instructors:
Professor David Stewart
Professor Keiho Yoshida

Jane Yoon
Yudena Yng
Daniel Stenzel
Ali Shahbaba
Sebana Periade
Mike Walewski
Jeff Hoyer
James Ho

TA's:
Please weigh the may change.
Final exam: 25% of grade.
Two problems: 15% of grade each.
Answers: 10% of grade.
About 2 assignments involving both programming and written.

Coursework

Substantial project.
More intense version of CS 211.
Required for CS majors.
1 Grace project course.

CS 212

Introduction differences.
Student discussed and diverse are permitted in case of
Monopoly is strongly encouraged, particularly in
Finding a partner: feel to network.
You can do them by yourself if you like.
Assignments must be done by teams of two students.

Assignments

Time and place: TBA

Bootcamp will be taught by Professor Schwartz.

Bootcamp

Students with little or no Java knowledge: attend Java
methods, instance variables.

CS 211 assumes basic Java knowledge: classes, objects.

Java Bootcamp
Sam Loyd's 8-puzzle

Goal: Given an initial configuration of tiles, find a sequence of moves that will lead to the sorted configuration.

Moves are: blank space moves, 4:2, 1:3, 2:1, 3:4.

Data, structures, arrays, lists, stacks, queues, trees, hash-tables.

4: Concurrent.
3: Exception handling.
2: Interprocess, interthreads.
1: Classes, objects, interfaces.

Learn the following:

Objectives of CS 211

One credit course based on attendance.

Two-hour labs in which students work together in co-operative Academic Excellence Workshops.
State Transition Diagram of 8-puzzle

State Transition Diagram for a 2x2 Puzzle
Stack: an array of integers

Stack starts at address 0 and grows to larger addresses.

Stack grows when integer is "pushed" on top.

Stack shrinks when integer is "popped" from top.

Stack operations are used to implement SaM commands.

They are NOT SaM commands themselves.

They are used to implement SaM commands.

They are NOT SaM commands themselves.

Stack operations are used to implement SaM commands.

Stack: an array of integers

Stack starts at address 0 and grows to larger addresses.

Stack grows when integer is "pushed" on top.

Stack shrinks when integer is "popped" from top.

Stack operations are used to implement SaM commands.

They are NOT SaM commands themselves.

Stack operations are used to implement SaM commands.
SaM commands

ADD

//pops two values from top of stack
//adds them and pushes result

SUB

//pops two values from stack (say top and below)
//and pushes result of doing (below - top)

TIMES

//pushes that integer on stack

GREATER

//boolean values are simulated using 0/1 (false/true)
//logical AND

AND

STOP

//terminate execution of program

ALL arithmetic/logical operations pop values from stack
perform operation and push result.

Booleans are simulated in SaM with integers
True -> 1, False -> 0

PUSHIMM  16

in example, we would push 0.
if Vtop < Vbelow push 1 else push 0
in example, Vtop = 7 and Vbelow = 2
in example, Vtop = 7 and Vbelow = 7
if Vtop = Vbelow pop and Vtop from stack
if Vtop <= Vbelow

SaM commands

SaM commands
By the end of CS 211, you will be able to design and write

Here are two simple Stack programs:

1. When GPU do we use?
2. How do we load programs from a file?
3. How do we represent the internal state of Stack?
4. How do we interpret code to interpret each of the operators?
5. How do we use a top of stack?