Panel:

NSF-Sponsored Innovative Approaches to Undergraduate Computer Science

Stephen Bloch (Adelphi University)
Amruth Kumar (Ramapo College)
Stanislav Kurkovsky (Central CT State University)
Clif Kussmaul (Muhlenberg College)
Matt Dickerson (Middlebury College), moderator

Project	Web site(s)	Intervention	Delivery	Supervision
Program by Design Stephen Bloch **NSF awards 0010064 & 0618543**	http://programbydesign.org http://picturingprograms.org http://www.ccs.neu.edu/home/matthias/HtDP2e/ http://racket-lang.org http://wescheme.org	curriculum with supporting IDE, libraries, & texts	in class; software and textbook are free downloads or web-based	normally active, but can be done other ways
Problets Amruth Kumar **NSF award 0817187**	http://www.problets.org	in- or after-class problem-solving exercises on programming concepts	applet in a browser	none - teacher not needed, although some adopters use it in active mode too
Mobile Game Development Stan Kurkovsky **NSF award DUE-0941348**	http://www.mgdcs.com/	in-class or take-home programming projects	PC	passive - teacher as facilitator to answer Qs
POGIL Clif Kussmaul **NSF award TUES 1044679**	http://pogil.org http://cspogil.org	in-class activity	paper or web	passive - teacher as facilitator to answer Qs

Project	Course(s)	Language(s)	Focus
Program by Design Stephen Bloch	Middle school, pre-AP CS in HS, CS0, CS1, CS2 in college	Usually Scheme-like teaching languages leading into Java; has also been done in Python, ML, Java, Scala, ...	problem-solving process, particularly test-driven development and use of data types to guide coding & testing
Problets Amruth Kumar	AP-CS, CS I, CS 2. also as refresher or to switch languages in other courses	C, C++, Java, C#	code tracing, debugging, expression evaluation, predicting program state
Mobile Game Development Stan Kurkovsky	AP-CS, CS1, CS2	Java	core OO programming; intro to advanced subjects such as AI, networks, security
POGILClif Kussmaul	CS1, CS2, SE, *etc.* CS Principles (HS) (used across STEM)	often concept-based and language-independent; CS1 in Java & Python	knowledge construction, process skills

	Teacher			Students		
Project	**prep before class**	**during class**	**after class**	**#**	**during**	**after**
Program by Design Stephen	select examples	model problem-solving process; answer questions	feedback to students: how well did they follow the process?	solo or small team	mostly programming	
Problets Amruth	sign up to get URL; specify which problet to use when	none; not even in supervised mode	use report to select concepts to review in class	solo	solving problems on programming	
Mobile Game Develop-ment Stan	become familiar with the technical scaffolding provided by each project and with sample solution	explain objectives, demonstrate sample solution, help students with scaffolding	review completed programming projects	teams of 2	working on programming project	
POGIL Clif	make copies or post. (writing activities is time intensive)	facilitate teams, share conclusions	review team reports	teams of 2-4	working through activity	summary report (optional)

Program By Design

Stephen Bloch, Adelphi University

with Eli Barzilay, John Clements, Matthias Felleisen, Robby Findler, Kathi Fisler, Matthew Flatt, Kathy Gray, Shriram Krishnamurthi, Viera Proulx, Emmanuel Schanzer, ...

Curricular ideas

- Start in simple, beginner-friendly language
  - need beginner-friendly IDE & compiler
- Teach fundamental, transferable principles & habits
- Test-driven development from the beginning
  - need beginner-friendly testing harness
- Graphics, animation, GUI, music as motivators
  - need beginner-friendly libraries
- Then revisit same ideas in “mainstream” language (next semester or next year)

Pedagogical ideas

- Concrete design recipe
  - Identify input & output data types
  - Write test cases (guided by data types)
  - Write function skeleton (guided by data types)
  - Fill in gaps (guided by test cases)
  - Run test cases
- Each step is explicit & worth partial credit
- Writing test cases is *much* easier for functional than imperative code, so start in functional paradigm
  - even for graphics & interaction
- Functional GUI programming teaches model/view

Technical ideas

- Start in language subset…
  - *enforced by compiler*
  - Several concentric languages matching stages of curriculum
- Read-eval-print loop to encourage experimentation
  - like DrJava, BlueJ CodePad, irb, python, ghci, *etc.*
- “Image” is a data type, just like “integer” or “string”
  - *even in the REPL*
  - Can enter an image as a literal, interactively
  - Can see images as expression values, interactively
- Demo: http://screencast.com/t/12O3RGxFH

Versions of the curriculum

- Bootstrap (middle school)
  - http://bootstrapworld.org
- *Picturing Programs* (high school pre-AP, college CS0)
  - http://picturingprograms.org
- *How to Design Programs* 2ed (college CS1)
  - http://www.ccs.neu.edu/home/matthias/HtDP2e/
  - or search “htdp2e”

Software support

- WeScheme (IDE in a browser, used with Bootstrap)
  - http://wescheme.org
- DrRacket (IDE, used with *PP* and *HtDP*)
  - http://racket-lang.org
- JavaLibWorld and JavaLibTester (support libraries for Java-based course)
  - search on GitHub

Who likes this approach?

- Grants from Exxon, DoEd, NSF, Google
- ACM SIGCSE “Outstanding Contribution to Computer Science Education” award (2011)
- ACM Karlstrom award (2009)

Who uses this approach?

Bootstrap:

Park Elementary School
Ballou High School
Edison Middle School
Yanbu International School
Crossroads School
Sedro-Woolley High School
Albuquerque Academy
124 more omitted

NYOS Charter School
Boston Latin Academy
United for Success Academy
Barnard Saturday Science Seminar
St. Andrew's-Sewanee School
Academy for Science and Design
International School Ho Chi Minh City

Who uses this approach?

Picturing Programs:

- University of Toronto
- University of California, Irvine
- Vassar College
- Adelphi University
- Georgia Regents University
- Indian Institute of Information Technology and Management-Kerala, Trivandrum, India
- St Francis Borgia HS
- Whitney Young HS
- The Fay School
- Lakehill Preparatory School
- Aberdeen HS
- Holy Name HS
- Owatonna HS
- Bancroft School
- Dighton-Rehoboth HS
- Augusta Preparatory Day School
- Nashoba Regional High School
- St Luke's School
- The Webb Schools
- oxfordcomputerscience.wikispaces.org (HS level)
- DuPont Manual HS (in Scala?)
- Evergreen Middle School
- at least one 4th-grade teacher (!)
- various others omitted

Who uses this approach?

How to Design Programs

- University of Chicago
- Northeastern University
- University of Delaware
- Westmont College
- Worcester Polytechnic Institute
- University of Notre Dame
- University of Waterloo
- Istanbul Bilgi University
- Seton Hall University
- Berry College
- Brown University
- Monmouth College
- University of Minnesota Morris
- Northwestern University
- Suffolk County Community College
- University of British Columbia (both traditional course and MOOC)
- Zefat Academic College
- UNAM
- Manhattanville College
- Rhode Island College
- University of Tübingen
- University of Freiburg
- University of Dallas
- South Carolina State University
- Pacific Union College
- Humboldt College
- University of Chile (in Python)
- Ochanomizu University (in OCaML)
- Carnegie-Mellon (in ML)
- various others omitted

Problets

Amruth Kumar,
amruth@computer.org
problets.org

Curricular Goals

- Learn programming concepts by solving problems
- Supplement classroom instruction
- Complement programming projects

What Problets do:

- Present problems
- Grade student’s answer
- Provide instant feedback
- Record student performance
- Provide summary to the instructor

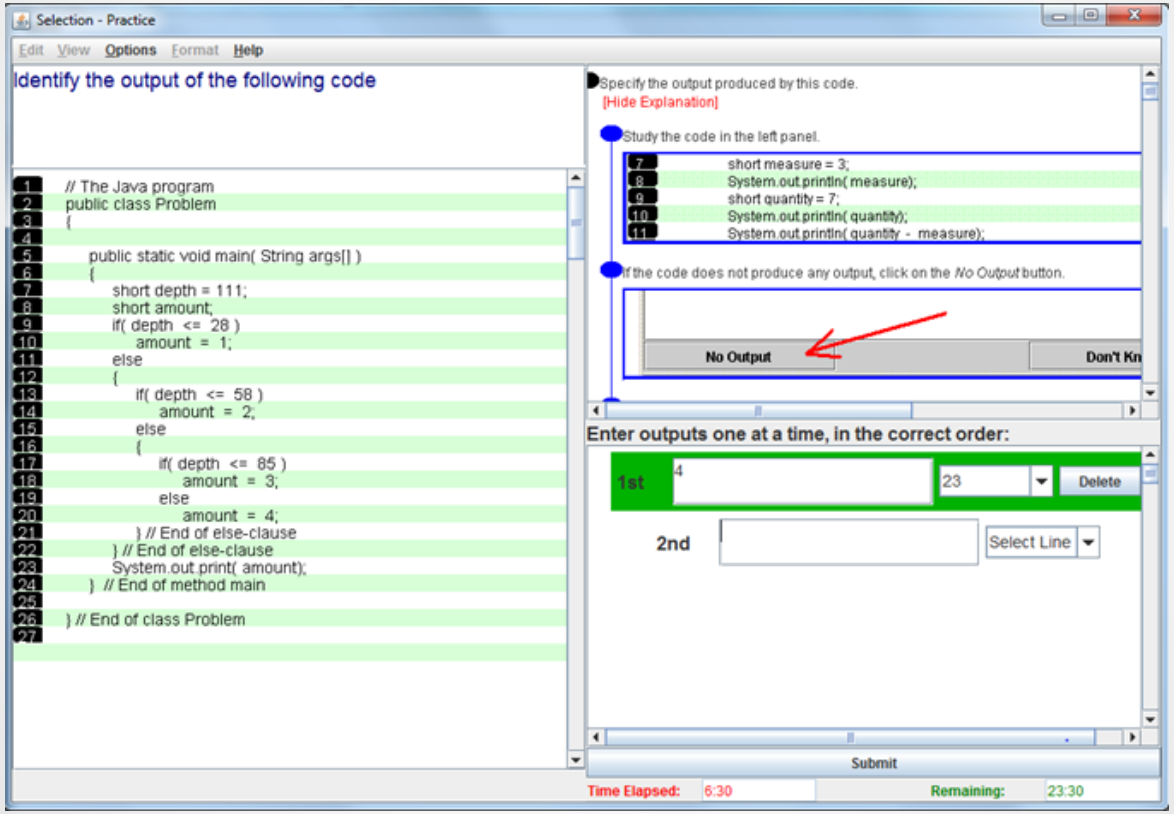
Types of problems

- Identify the output of a program
- Debug a program
- Resolve the state of program variables
- Evaluate expressions

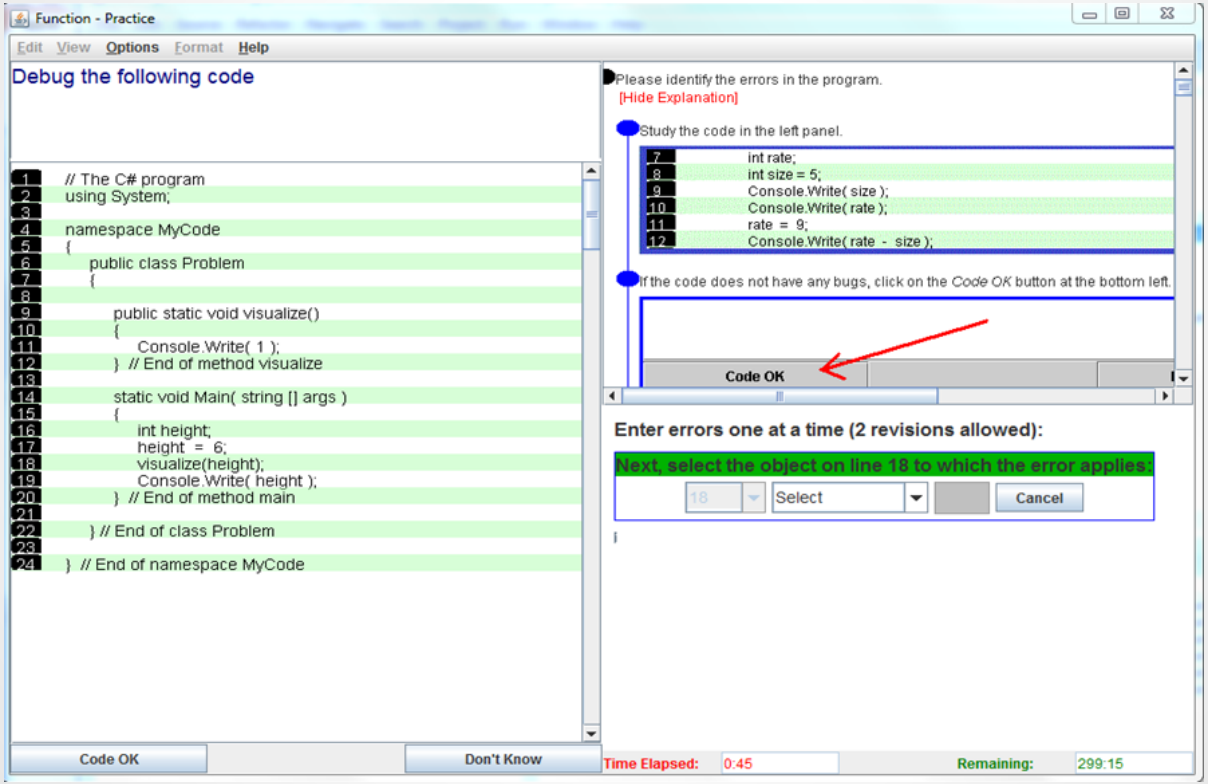
::

- Step-by-step
- *Not* multiple-choice problems

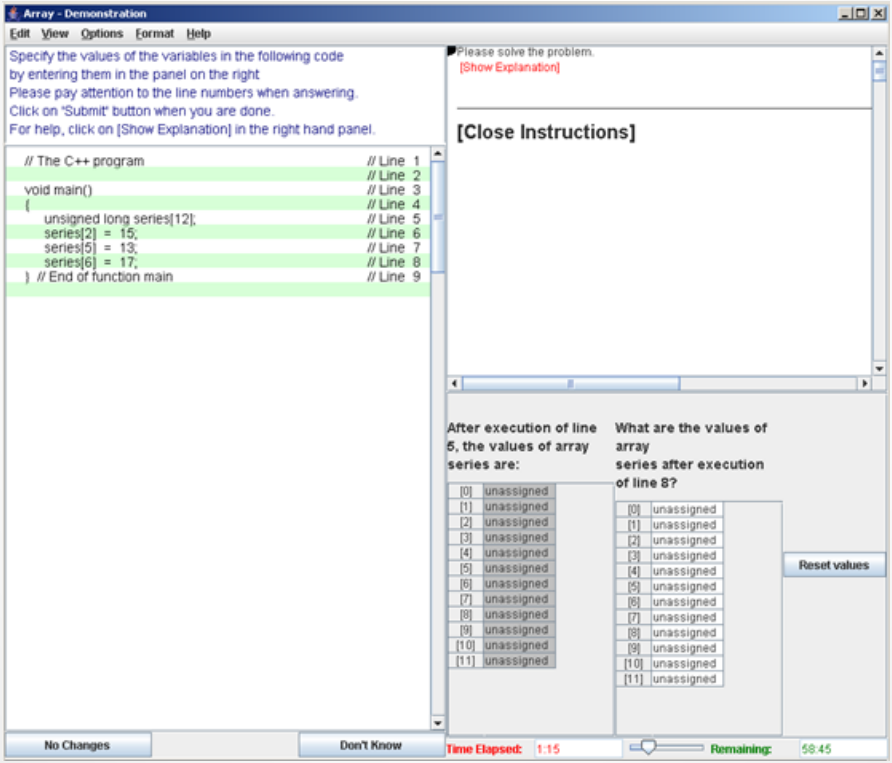
Identifying the output



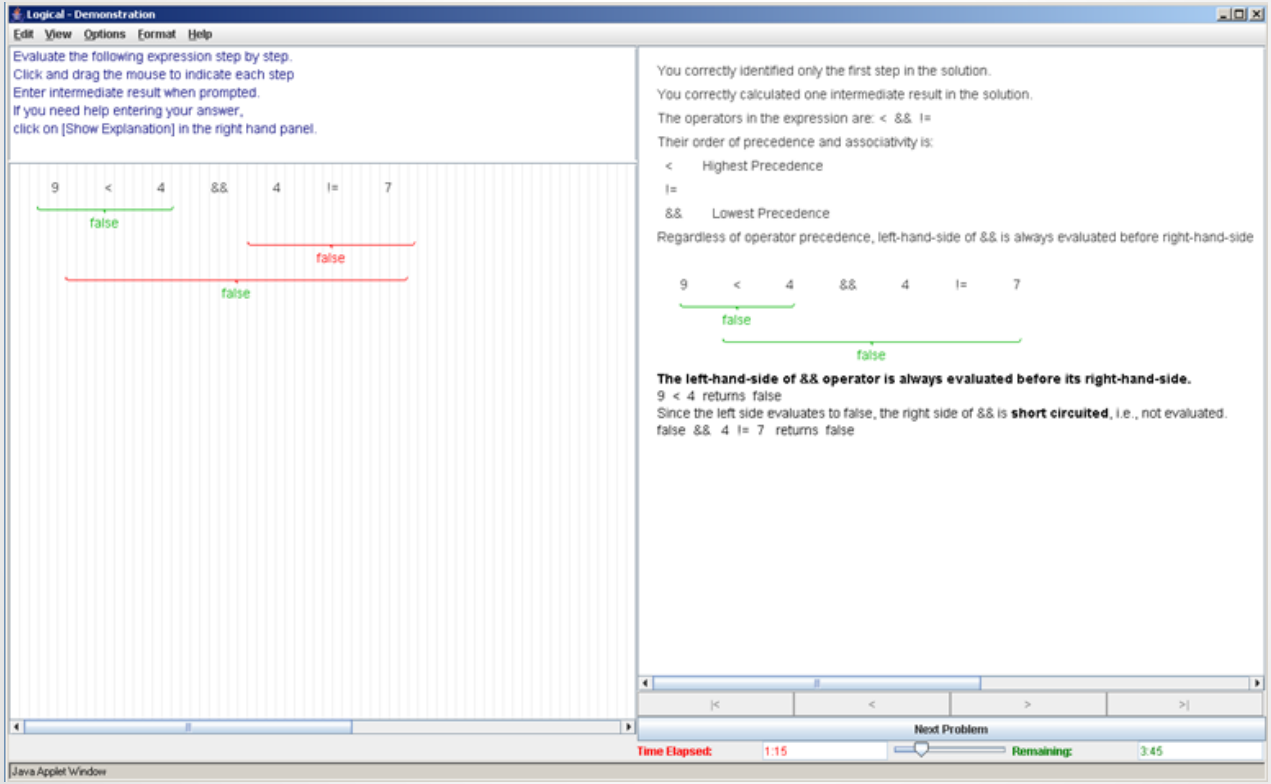
Debugging


Function - Practice
Edit View Options Format Help
Debug the following code
1 // The C# program
2 using System;
3
4 namespace MyCode
5 {
6 public class Problem
7 {
8
9 public static void visualize()
10 {
11 Console.Write(1);
12 } // End of method visualize
13
14 static void Main(string [] args)
15 {
16 int height;
17 height = 6;
18 visualize(height);
19 Console.Write(height);
20 } // End of method main
21
22 } // End of class Problem
23
24 } // End of namespace MyCode
Code OK
Don't Know
Please identify the errors in the program.
[Hide Explanation]
Study the code in the left panel.
7 int rate;
8 int size = 5;
9 Console.Write(size);
10 Console.Write(rate);
11 rate = 9;
12 Console.Write(rate - size);
If the code does not have any bugs, click on the Code OK button at the bottom left.
Code OK
Enter errors one at a time (2 revisions allowed):
Next, select the object on line 18 to which the error applies:
18
Select
Cancel
Time Elapsed: 0:45
Remaining: 299:15

State of a variable



Expression Evaluation



Topics (17 modules)

- Expression evaluation
  - Arithmetic, Relational, Logical, Assignment, Bitwise
- Selection
  - if, if-else, switch, nested statements
- Loops
  - while, for, do-while, nested loops, infinite loops
- Functions - behavior, bugs, recursion
- Arrays, Access in Classes, C++ pointers

Topics and Problems

Topic	Sub-Topic	Used Since	No. Problems	Learning Objectives
Expressions	Arithmetic	Fall 2004	192	25
	Relational	Fall 2004	268	24
	Logical	Fall 2006	280	21
	Assignment	Fall 2008	255	19
	Bitwise	Fall 2010	303	28
Selection	If/if-else	Spring 2005	165	12
	switch	Spring 2010	147	12
Loops	while	Fall 2004	201	9
	for	Fall 2004	213	10
	do-while	Fall 2010	125	15
	Advanced	Spring 2010	139	13
Functions	Debugging	Fall 2009	117	9
	Tracing	Fall 2009	95	10
	Recursion	Spring 2013	68	10
Arrays	1-D	Fall 2010	172	14
Classes	Access	Spring 2013	128	18
Total			**2868**	**249**

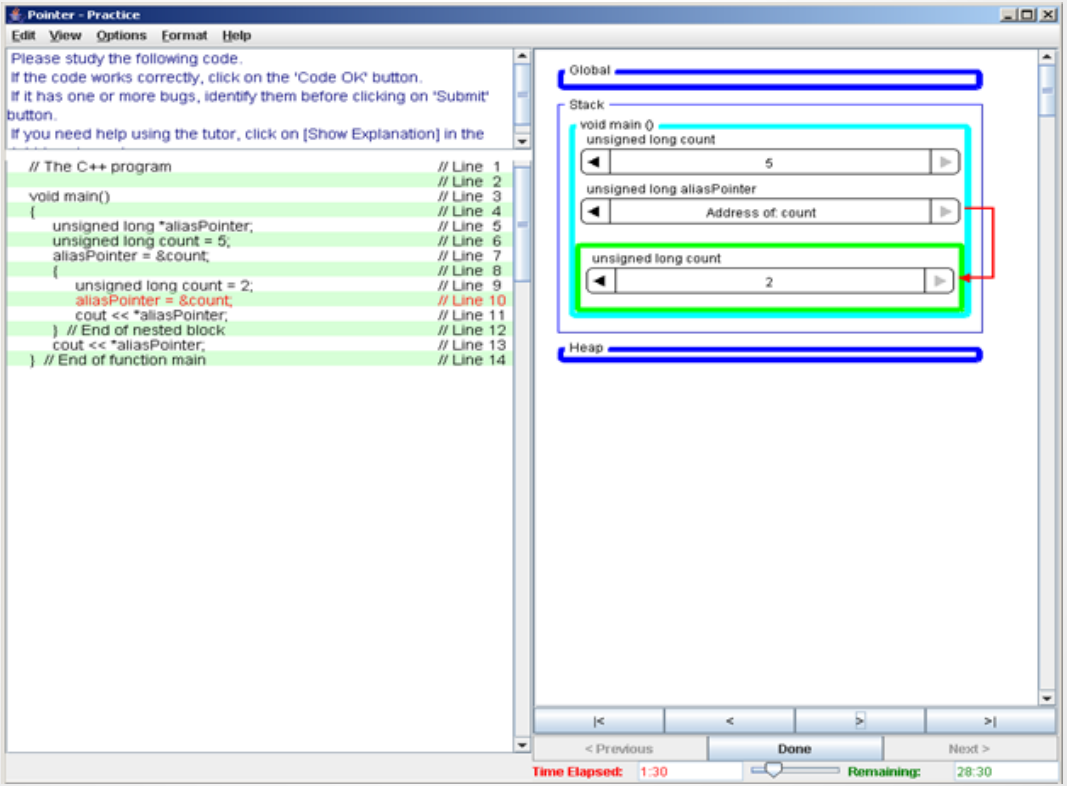
Target

- Languages:
  - Java, C, C++, C#, some Visual Basic
- Audience:
  - CS I, CS II, AP-CS
  - Refresher for advanced courses/language change
- Institutions:
  - High school, 2-year, 4-year colleges

Pedagogy

- Learn by solving problems
  - Mastery learning
- Step-by-step explanation of correct solution
- Adaptive problem-sequence
- Randomized problem set
- Learning at one’s pace on one’s time
  - Any time, as often as necessary
- Extensively evaluated over 14 years

Visualization



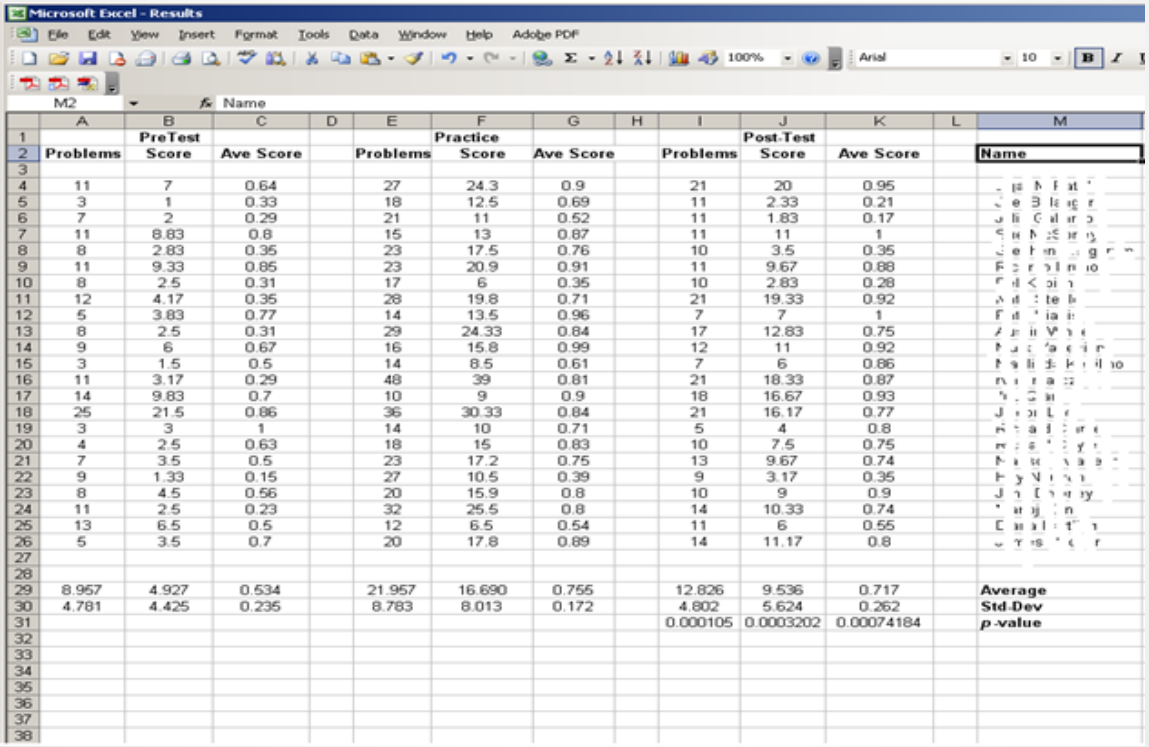
Usage

- Closed-Lab exercises
- After-class assignments (24 x 7)
- Language refreshers
  - As many as necessary
  - When necessary
  - As often as necessary
- Continuous third-party use since **fall 2004**
  - 60+ schools in spring 2014

Adoption

- No software installation necessary - Web-based
- No supervision necessary - self-administering
- Report available on demand
  - By problems, learning objectives
- Free for educational use

Snapshot of a report



Microsoft Excel - Results

File Edit View Insert Format Tools Data Window Help Adobe PDF

M2 Name

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		PreTest				Practice				Post-Test			
2	Problems	Score	Ave Score		Problems	Score	Ave Score		Problems	Score	Ave Score		Name
3													
4	11	7	0.64		27	24.3	0.9		21	20	0.95		
5	3	1	0.33		18	12.5	0.69		11	2.33	0.21		
6	7	2	0.29		21	11	0.52		11	1.83	0.17		
7	11	8.83	0.8		15	13	0.87		11	11	1		
8	8	2.83	0.35		23	17.5	0.76		10	3.5	0.35		
9	11	9.33	0.85		23	20.9	0.91		11	9.67	0.88		
10	8	2.5	0.31		17	6	0.35		10	2.83	0.28		
11	12	4.17	0.35		28	19.8	0.71		21	19.33	0.92		
12	5	3.83	0.77		14	13.5	0.96		7	7	1		
13	8	2.5	0.31		29	24.33	0.84		17	12.83	0.75		
14	9	6	0.67		16	15.8	0.99		12	11	0.92		
15	3	1.5	0.5		14	8.5	0.61		7	6	0.86		
16	11	3.17	0.29		48	39	0.81		21	18.33	0.87		
17	14	9.83	0.7		10	9	0.9		18	16.67	0.93		
18	25	21.5	0.86		36	30.33	0.84		21	16.17	0.77		
19	3	3	1		14	10	0.71		5	4	0.8		
20	4	2.5	0.63		18	15	0.83		10	7.5	0.75		
21	7	3.5	0.5		23	17.2	0.75		13	9.67	0.74		
22	9	1.33	0.15		27	10.5	0.39		9	3.17	0.35		
23	8	4.5	0.56		20	15.9	0.8		10	9	0.9		
24	11	2.5	0.23		32	25.5	0.8		14	10.33	0.74		
25	13	6.5	0.5		12	6.5	0.54		11	6	0.55		
26	5	3.5	0.7		20	17.8	0.89		14	11.17	0.8		
27													
28													
29	8.957	4.927	0.534		21.957	16.690	0.755		12.826	9.536	0.717		Average
30	4.781	4.425	0.235		8.783	8.013	0.172		4.802	5.624	0.262		Std-Dev
31									0.000105	0.0003202	0.00074184		*p*-value
32													
33													
34													
35													
36													
37													
38													

Contact Information

Additional information at:

www.problets.org

If interested, please contact:

amruth@ramapo.edu

Acknowledgements: NSF CCLI DUE 0088864

Mobile Game Development

Stan Kurkovsky
Central Connecticut State University
http://www.mgdcs.com/

with Archana Chidanandan and Delvin Defoe

Overarching Goals

- Improve student engagement and motivation
- Decrease attrition in introductory CS courses

- Method: use a relevant learning context

Curricular Objectives

- Expose students to advanced topics
- Strengthen student mastery of the core concepts
- *CS is more than just coding!*

- Method: project-based learning modules

Learning Modules

- Context
  - A well-known game (arcade, board, etc.)
  - Casual games
- Learning objectives
  - Introduce an advanced topic (e.g. networking)
  - Reinforce a core topic (e.g. for loops)
- Game implementation
  - Working demo

Target

- Language
  - Java: J2ME, Android
- Audience
  - AP-CS, CS I, CS II
  - Also: advanced topical courses
- Institutions
  - High school, 2- and 4-year colleges

Pedagogy

- Context-based learning
- Relevance to everyday life
- Hands-on experiences
- Teamwork
- Instant gratification

Sample Modules

- Battleship - computer networking
- Connect Four - artificial intelligence
- Frogger - software engineering
- Space Bears - human-computer interaction
- Craps - security
- Text Twister - algorithms

Process Oriented Guided Inquiry Learning (POGIL)

Clif Kussmaul, Muhlenberg College

http://cspogil.org http://pogil.org

POGIL - Curricular Goals

- Across CS (& other STEM) disciplines, we should help our students learn to:
  - analyze, design, synthesize ideas
  - read, write, & debug code & docs
  - communicate (oral & written)
  - work in teams, manage time
  - learn or create ideas on their own

POGIL - Pedagogy

- Research shows that learning is improved when people:
  - work in **teams** with other people
  - construct knowledge through a **learning cycle** (explore, invent, apply)
  - receive prompt constructive **feedback**
  - **reflect** on learning process & outcomes

Process Oriented Guided Inquiry Learning

- Students work in **teams** with assigned **roles**(e.g. manager, recorder, speaker)
- Teams work on classroom activities that present a **model** followed by **questions**.
- Instructor is a facilitator, not a lecturer.
- Activities are designed to guide students to:
  - construct understanding of key ideas
  - develop key process skills

POGIL Example: 1st Day of CS1

Hi-Lo: Guessing Game

- Two players – A and B.
- A picks a number 1-100.
- B guesses a number.
- A responds “too high”, “too low”, or “you win”.
- Continue to play until B wins (or gives up).

Questions

1. Play the game 3 times.
2. Identify 4-5 strategies B could use to play.
3. (Discuss with class.)
4. Rank by # of guesses.
5. Rank by difficulty.
6. Plot rankings & describe.

POGIL Example: 1st Day of CS1

Strategy	S	D	Max	Avg
Random	4	1	100	50
Count up by 1.	3	2	100	50
Count up by 10, down by 1.	2	3	20	10
Split range in ½ each time.	1	4	7	6

Questions

1. Max # of guesses?
2. Avg # of guesses?
3. (Discuss with class.)
4. Repeat for 1-1000.
5. Repeat for any N.
6. Describe insights.
7. (Discuss with class.)

CS-POGIL

cspogil.org

- NSF TUES project (2011-2014)
  to develop POGIL materials for CS
  - CS2, Data Structures, Software Engineering
  - sci comp, CS1 (Java, Python), theory, AI, ...
- Numerous CS collaborators, including:
  - Helen Hu, Lisa Olivieri, Matt Lang, Chris Mayfield, Heidi Ellis, Stoney Jackson, Tammy Pirmann
- $$$ available to attend POGIL workshops

The POGIL Project
pogil.org

- Non-profit to support use of POGIL & related approaches
- Long history of NSF funding (15+ years)
- 3-day regional summer workshops
- Review POGIL activities, support classroom implementation

POGIL - Implementations

- Students: 10-200; HS, undergrad, grad
- Models: UML, Code, API doc
- Media
  - Paper copies for each team or student
  - Google Doc for each team
  - Presentation slides
- Team structure
  - Teams of 4, split for pair programming

DISCUSSION

- What might the approach *not* accomplish or do well?
- When would you *not* use it as opposed to when *would* you use it?
- How would your approach combine well with another of the approaches outlined here?