

Developing and Investigating Online Programming Resources

Andrew Petersen

Department of Mathematical and Computational Sciences

University of Toronto

andrew.petersen@utoronto.ca



UNIVERSITY OF
TORONTO

Goals

- Introduce two of my areas of research
 - Investigating the student experience in introductory computer science (CS1)
 - Identifying student difficulties with coding exercises (using empirical methods)

Goals

- Introduce two of my areas of research
 - Investigating the student experience in introductory computer science (CS1)
 - Identifying student difficulties with coding exercises (using empirical methods)
- Describe an example of the active use of research in course and curriculum (re-)development

Timeline

2007: First year (CS1) redesigned (in Python) based around “live” coding in courses and pair-programming in labs

~2010: Student attrition is identified as a concern

2010-11: Interviews conducted with students in CS1

A Student Perspective on Prior Experience in CS1

Anya Tafliovich, Jennifer Campbell, Andrew Petersen
SIGCSE 2013



UNIVERSITY OF
TORONTO

Context

- Prior experience (PE) mattered* in our CS1
- Fail rate:
 - 15% (with PE) vs 31% (without)
- Marks:
 - Students with PE scored half a letter grade better

* We are currently re-running this study, so in a few months, I may be able to tell you to what extent it still matters.

Our CS1

Python-based, in an *objects-late* structure

- Programming concepts:
 - Variable assignment
 - Conditionals
 - Counted and conditional loops
 - Functions
- Software process
 - Testing and test-driven design
 - Modular design
- Some “intro to CS” topics
 - Complexity, simple algorithms

CS1: Structure

- 12 week term
 - 3 1-hour lectures per week
 - 1 2-hour closed lab per week
- The lab utilizes *pair programming*
 - 2 students at 1 computer
 - The *driver* operates the computer
 - The *navigator* focuses on design and looks ahead to identify issues
- Pair programming was enforced by the TA
- The lab handout specifies when students switch roles

Methodology

- Applied a *student focused approach*
 - Demographic survey at the beginning of the course
 - 2 semi-structured interviews (30 minutes) after CS1 and again after CS2

Methodology

- Interviews were coded using a *grounded theory* methodology
 - All investigators coded a subset of interviews
 - Codes converged after two rounds of discussion and coding
- Each interview was coded by 2 investigators and merged, when necessary, by the third
- Codes were aggregated and then themes were identified through manual categorization

Research Questions

- How does PE affect peer interaction?
- What are students' beliefs on the relationship between PE and success in the course?

Research Questions

- **How does PE affect peer interaction?**
- What are students' beliefs on the relationship between PE and success in the course?

PE and Peer Interaction

Students reported four venues for interaction:

1. *In class* conversations: usually informal, and occasionally “overheard”
2. *Closed labs*: forced pairing for pair programming
3. *Assignment partnerships*: either pre-existing relationships or based on interactions in class/lab
4. *Online discussion board*: mostly just questions about course material

PE and Peer Interaction

Partnership success often hinged on perception of ability, and early in the course, PE determined ability.

1. *Successful partnerships:* Students have similar PE / perceived similar ability levels
2. *Failed partnerships:* Perceived skill levels were different
3. *Choosing to program solo*

A Successful Partnership

“There were two people I tried to work with ... We had the same level of understanding, so we could work through the exercises together. No rushing ahead or feeling slowed down.”

“On the third assignment, I just picked a partner who had relatively the same skills that I do. We kind of shared the work and had lots of debates and stuff, like you know people have, and, well, it turned out good.”

A Failed Partnership

“My partners knew a lot more than I did, and I didn’t want to slow them down ... so I didn’t learn anything.”

“I cruised with this one guy ... about half of the labs ... He had done some Java before, so he knew what to do. He did most of the labs, and I watched ... I [hurt] myself for the tests ... I didn’t know what I didn’t know until I started the test.”

“I tried to work with a partner, but more or less they either didn’t do anything or they just watched me. Or they just looked at the screen.”

Choosing to Program Solo

Many *successful* students *chose to program solo* after experiencing or observing a failed partnership.

- This reinforced feelings of isolation in some cases.

“I just want to do the things by myself so that I will have the confidence and then I will feel more comfortable.”

“The last assignment was on your own, which was good because you are not really depending on anyone else, so you know that you can do the stuff yourself.”

Research Questions

- How does PE affect peer interaction?
- **What are students' beliefs on the relationship between PE and success in the course?**

PE and Success

- Students universally believed PE helped
- Some students articulated the advantage as “knowing how much time is required”
- Other students claimed that experience lead to confidence and that confidence was the key advantage.

Peers with PE

- Many interviews talked about a “*group of experts*”
 - They were highly visible, in particular on the discussion board
 - They were generally described as being “very few” and having “tons of experience”
- Some interviewees thought they were *beneficial*
 - Answering questions on the discussion board
 - Providing a goal / target

Peers with PE

- Many interviews talked about a “*group of experts*”
 - They were highly visible, in particular on the discussion board
 - They were generally described as being “very few” and having “tons of experience”
- Some interviewees thought they were *harmful*
 - They were intimidating: how can you compete?
 - They dominated the discussion

Peers with PE

- Many interviews talked about a “*group of experts*”
 - They were highly visible, in particular on the discussion board
 - They were generally described as being “very few” and having “tons of experience”
- But we never identified them.
 - Every interviewee agreed they existed.
 - None of our interviewees identified with being in this group
 - It’s possible that PE was being used as a *reason* for someone else’s perceived success / ability

Summary

- How does PE affect peer interaction?
 - The most successful groups had similar ability levels
 - Students who experience or observed a bad pair frequently chose to work alone
- What are students' beliefs on the relationship between PE and success in the course?
 - Students *might* be conflating PE and success
 - Students attributed PE to students who demonstrated knowledge or success
 - Everyone saw someone else with more experience

Impact on our CS1

- Our courses changed significantly in 2013
 - Shifted to an hybrid (inverted) format without required labs
 - Significant effort was spent on online resources
 - In class time was spent on active learning
- The interview study was one of several factors
 - In 2012, U of T encouraged the development of MOOCS
 - For several years, we had been shifting towards the use of PI and other active pedagogies

Timeline

2007: First year (CS1) redesigned (in Python) based around “live” coding in courses and pair-programming in labs

~2010: Student attrition is identified as a concern

2010-11: Interviews conducted with students in CS1

Timeline

2007: First year (CS1) redesigned (in Python) based around “live” coding in courses and pair-programming in labs

~2010: Student attrition is identified as a concern

2010-11: Interviews conducted with students in CS1
PCRS developed for Peer Instruction (PI) in CS1

2013: Coursera MOOC leads to generation of videos for CS1
CS1 offerings include “hybrid” and online versions

2014: CS1 resources enhanced with Ontario government support
Digital design (258) incorporates online resources

2015: Systems programming (209) moves to a hybrid format (gov’t)
Databases (343) incorporate online resources

2016 (planned): CS3 moves to a hybrid format with U of T support

Timeline

2007: First year (CS1) redesigned (in Python) based around “live” coding in courses and pair-programming in labs

~2010: Student attrition is identified as a concern

2010-11: Interviews conducted with students in CS1
PCRS developed for Peer Instruction (PI) in CS1

2013: **Coursera MOOC** leads to generation of videos for CS1
CS1 offerings include “hybrid” and online versions

2014: CS1 resources enhanced **with Ontario government support**
Digital design (258) incorporates online resources

2015: Systems programming (209) moves to a hybrid format (**gov’t**)
Databases (343) incorporate online resources

2016 (planned): CS3 moves to a hybrid format **with U of T support**

Timeline

2007: First year (CS1) redesigned (in Python) based around “live” coding in courses and pair-programming in labs

~2010: Student attrition is identified as a concern

2010-11: Interviews conducted with students in CS1
PCRS developed for Peer Instruction (PI) in CS1

2013: Coursera MOOC leads to generation of videos for CS1
CS1 offerings include “hybrid” and online versions

2014: CS1 resources enhanced with Ontario government support
Digital design (258) incorporates online resources

2015: Systems programming (209) moves to a hybrid format (gov’t)
Databases (343) incorporate online resources

2016 (planned): CS3 moves to a hybrid format with U of T support

Facilitating Code-Writing in PI Courses

Dan Zingaro, Olessia Karpova, Yuliya Cherenkova,
Andrew Petersen
SIGCSE 2013



UNIVERSITY OF
TORONTO

What is Peer Instruction (PI)?

- Active learning pedagogy developed for physics
- Instead of traditional lectures ...
 - Teacher poses multiple choice question (MCQ)
 - **Individual vote**: students vote on their own
 - Students discuss in small groups
 - **Group vote**: students vote again
 - Teacher conducts whole class discussion
- In a 50 minute course, you might expect 2-3 of these cycles.

Why PI?

- Research shows considerable gains between the individual and group vote
 - e.g., 51% correct on solo, 63% on group¹
- Normalized gain (NG) is the typical metric
 - NG is the proportion of students that answer incorrectly that subsequently answer correctly
 - Typical reported NG values are 30-40%^{1,2}

1. D. Zingaro. Experience report: Peer instruction in remedial computer science. *Proceedings of the 22nd World Conference on Educational Multimedia, Hypermedia & Telecommunications*, pages 5030–5035, 2010.

2. B. Simon, M. Kohanfars, J. Lee, K. Tamayo, and Q. Cutts. Experience report: Peer instruction in introductory computing. *Proceedings of the 41st SIGCSE Technical Symposium on Computer Science Education*, pages 341–345, 2010.

PI and Code Writing

- MCQs require code reading and tracing.
 - Could also write Parson's problems MCQs.
- But what if we want students to write code?
 - Code writing is often the focus of exams
 - Evidence suggests that code reading and tracing are prerequisites for code writing – but don't imply that a student can write code.^{1,2}
- Immediate, formative feedback – for both students and instructors – is critical for the learning process.

1. M. Lopez, J. Whalley, P. Robbins, and R. Lister. Relationships between reading, tracing and writing skills in introductory programming. *Proceedings of the 4th International Workshop on Computing Education Research*, pages 101-112, 2008.
2. L. Murphy, S. Fitzgerald, R. Lister, and R. McCauley. Ability to 'explain in plain English' linked to proficiency in computer-based programming. *Proceedings of the 9th International Conference on International Computing Education Research*, pages 111-118, 2012



PCRS

- We developed a web tool to support instructors' use of PI in class
 - ... for programming courses
- Students use portable devices to answer questions posed by the instructor.
- Submissions are marked automatically, and the results of tests are used as a proxy for a MCQ choice.

Consider this part of a doctest message: `TestResults(failed=3, attempted=5)` How many tests passed (the actual output matched the expected ou...

Time*

2016-03-10 20:56:56

Section*

On Campus @ StG

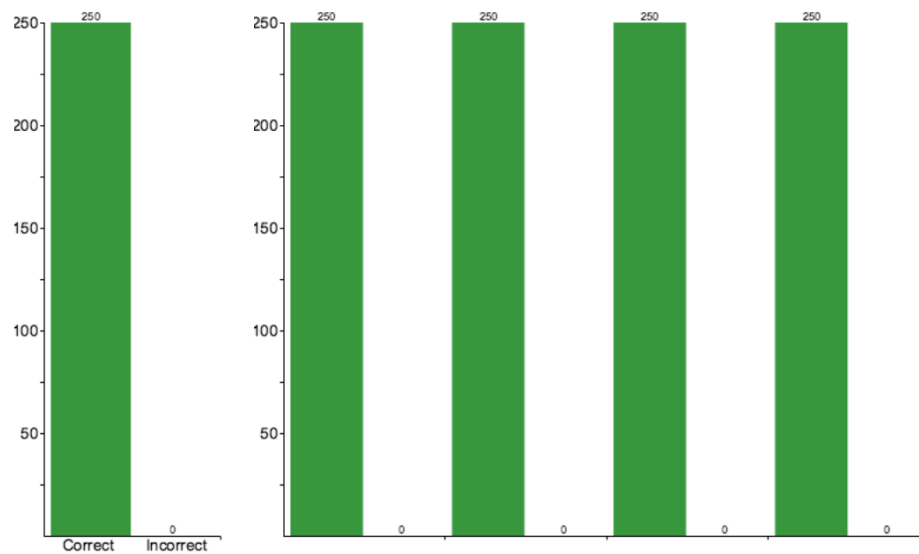
☐ Final

Go

Consider this part of a doctest message:

`TestResults(failed=3, attempted=5)`

How many tests passed (the actual output matched the expected output)?



Options

Option

1 0



UNIVERSITY OF
TORONTO

... Things Have Changed

What is PCRS anyway?

- 2010: Python Classroom Response System
- 2012: Python *Course Resource* System
- 2014: *Programming* Course Resource System

The changing titles match our understanding of what our courses needed.

Watch videos and complete problems to learn the Week 9 material.


Challenges

➤  Testing Automatically Using doctest ➡


Practice 0/2

➤  Writing a '__main__' program ➡


Practice 0/3

➤  Creating Your Own Types ➡


Practice 0/6

➤  Testing Automatically Using unittest ➡


Practice 0/2

➤  Choosing Test Cases ➡

Practice 0/2

➤  Testing Functions that Mutate Values ➡

Practice ✓

➤  Week 9 Prepare Exercise ➡

Credit 0/4





Testing Automatically Using doctest - Part 1 of 1

[Video summary](#)

Testing Automatically using doctest (Video 1, 6:17)

Learn how to test functions automatically using doctest.



[Download Video](#)

Multiple Choice Question

Consider this part of a doctest message:

```
TestResults(failed=3, attempted=5)
```

  7/4

[PCRS-C](#)[PCRS-Python](#)[PCRS-sql](#)[Download](#)[Contributors](#)

PCRS is an application for bundling interactive programming exercises with video-based instruction that has been developed at the University of Toronto under the supervision of [Andrew Petersen](#). Currently, PCRS supports [Python](#), [C](#), and [Relational Algebra and SQL](#), in addition to multiple choice exercises, and is in use in courses at the University of Toronto and Queen's University. The system is also an active research platform, with data from PCRS being used to explore novice programmer misconceptions in Python and C.

PCRS is open source, is [available for download](#), and can be used and modified freely. While most [contributors to date](#) have been based at the University of Toronto, we welcome collaborations in development and research.

<https://mcs.utm.utoronto.ca/~pcrs/pcrs/>



UNIVERSITY OF
TORONTO

PCRS in 2016

- Currently in use in four courses at U of T
 - ~5000 users per term
- Two other universities use PCRS materials
- Moving activities online, with logging, has created new research opportunities
 - Student (ab)use of multiple choice questions
 - Opportunities for mining submissions to code exercises
 - ...

Employing Multiple-Answer Multiple Choice Questions

Andrew Petersen, Michelle Craig, Paul Denny
ITiCSE 2016: Tips and Tricks session



UNIVERSITY OF
TORONTO

Multiple-Answer Multiple-Choice

- We noticed that in our formative, online context, students guess to circumvent the system.
 - Students can submit as many times as necessary.
 - The system provides immediate feedback.
- We deployed multiple-answer multiple-choice questions to deter guessing.

Raising the Grade



Consider the program below where one statement is replaced by `/* MISSING STATEMENT GOES HERE */`. Each of the answers below is a possible statement to insert. Select all the statements that result in the output "You earned 80! Yup 80!".

```
#include

int main() {

    int grade = 79;
    int *pgrade;
    pgrade = &grade;
    int raised_grade = 80;

    /* MISSING STATEMENT GOES HERE */

    printf("You earned %d! Yup %d!", grade, *pgrade);
    return 0;
}
```

- ☐ `grade = grade + 1;`
- ☐ `pgrade = &raised_grade;`
- ☐ `grade = raised_grade;`
- ☐ `*pgrade = raised_grade;`
- ☐ `*pgrade = *pgrade + 1;`
- ☐ `grade = 80;`
- ☐ `*pgrade = 80;`
- ☐ `grade = *pgrade + 1;`

History

Submit



UNIVERSITY OF
TORONTO

Impact of MAMCQs

- Previous criticisms of MAMCQs apply less in CS
 - Writing unambiguous stems and options is easier in a programming context.
- The problems reduce guessing
- The *facility* of our MAMC problems is .44 – more appropriate for formative feedback
- Questions written to cover “topic areas” may be effective predictors of exam performance

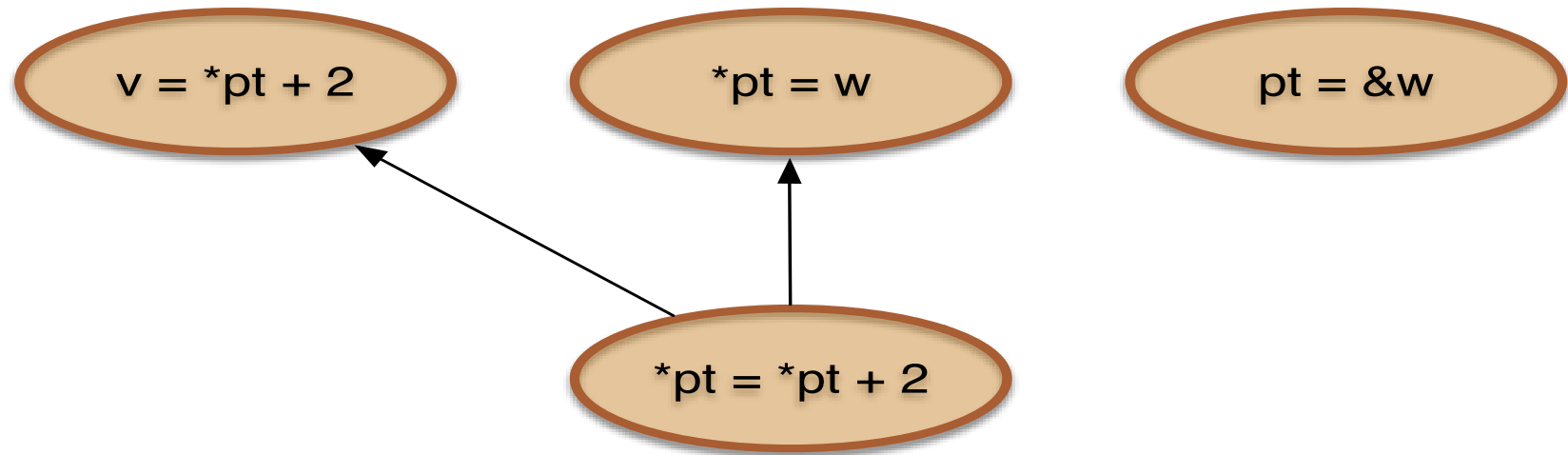
Student Difficulties with Pointer Concepts in C

Michelle Craig, Andrew Petersen
ACE 2016

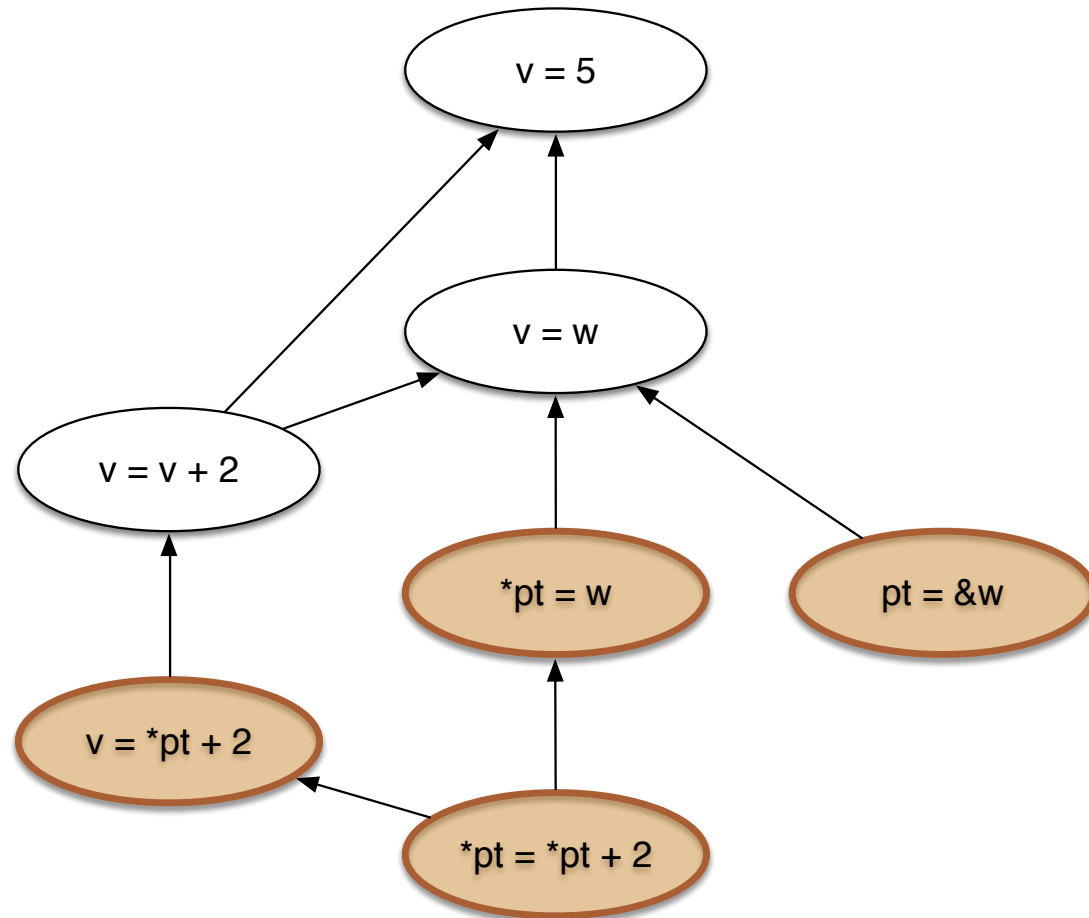


UNIVERSITY OF
TORONTO

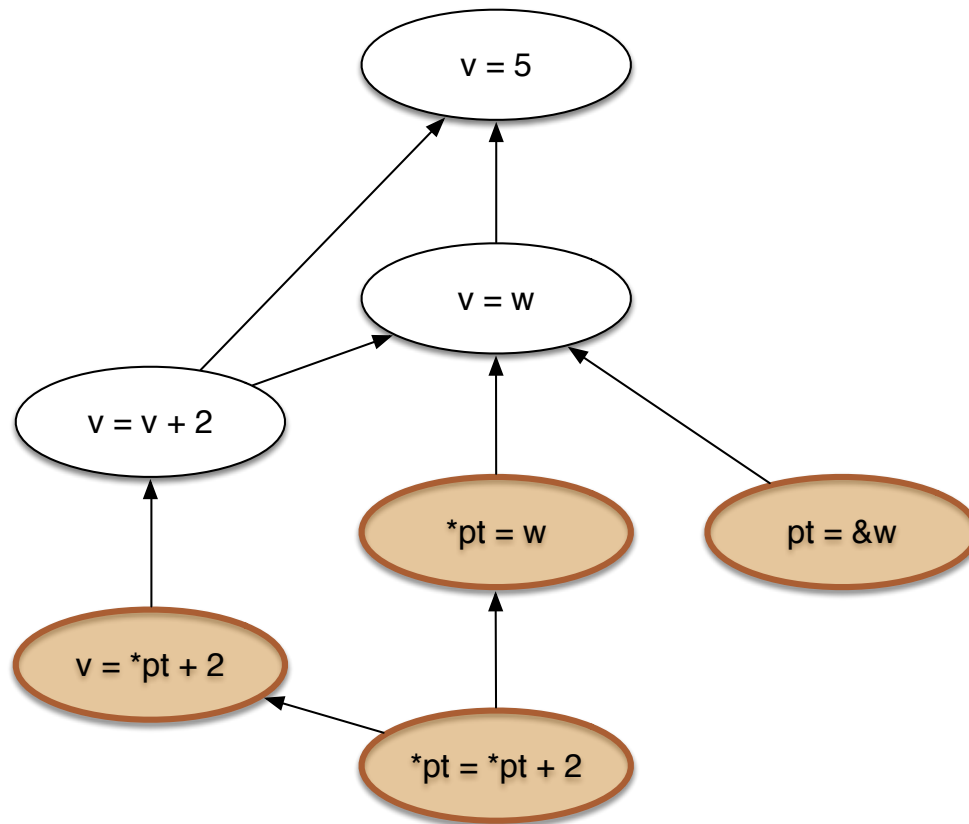
A Pointer Concept Taxonomy



A Pointer Concept Taxonomy



A Pointer Concept Taxonomy



- Double pointers
- Pointers as parameters to functions
- Pointers and arrays
- Pointer arithmetic

Context

- **The Course**
 - 2nd year C and systems programming class
 - taught for many years to 100's of students
- **The Lab**
 - Online delivery & submission of weekly labs
 - Drop-in help centre lightly used
 - Coding exercises and select all that apply (MAMCQ)
- **Lots of data (every time they press submit)**
 - 341 students consented to participate
 - over 10,000 submissions (code and MAMCQ)

Testing the Pointer Taxonomy

- **Created MAMCQ questions to match taxonomy**
- Two versions (pre & post)
 - Two treatment groups to validate equivalency
- Metrics for performance?
 - Unlimited attempts with no penalty
 - No marks for partially correct answers
 - Interested in relative performance **per option**

Raising the Grade



Consider the program below where one statement is replaced by `/* MISSING STATEMENT GOES HERE */`. Each of the answers below is a possible statement to insert. Select all the statements that result in the output "You earned 80! Yup 80!".

```
#include

int main() {

    int grade = 79;
    int *pgrade;
    pgrade = &grade;
    int raised_grade = 80;

    /* MISSING STATEMENT GOES HERE */

    printf("You earned %d! Yup %d!", grade, *pgrade);
    return 0;
}
```

- ☐ `grade = grade + 1;`
- ☐ `pgrade = &raised_grade;`
- ☐ `grade = raised_grade;`
- ☐ `*pgrade = raised_grade;`
- ☐ `*pgrade = *pgrade + 1;`
- ☐ `grade = 80;`
- ☐ `*pgrade = 80;`
- ☐ `grade = *pgrade + 1;`

History

Submit



UNIVERSITY OF
TORONTO

Testing the Pointer Taxonomy

- Created MAMCQ questions to match taxonomy
- **Two versions (pre & post)**
 - Two treatment groups to validate equivalency
- Metrics for performance?
 - Unlimited attempts with no penalty
 - No marks for partially correct answers
 - Interested in relative performance **per option**

Testing the Pointer Taxonomy

- Created MAMCQ questions to match taxonomy
- **Two versions (pre & post)**
 - Two treatment groups to validate equivalency
- Metrics for performance?
 - Unlimited attempts with no penalty
 - No marks for partially correct answers
 - Interested in the relative performance **per option**

Metrics for Performance on MAMCQ

- Churn
 - How many times does the student flip this option's answer?
- Last to Change
 - In what percentage of the students is this the last option fixed to get a fully correct submission?

Metrics for Performance on MAMCQ

- Churn
 - How many times does the student flip this option's answer?
- Last to Change
 - In what percentage of the students, is this the last option fixed to get a fully correct submission?

Metrics for Performance on MAMCQ

- Churn
 - How many times does the student flip this option's answer?
- Last to Change
 - In what percentage of the students, is this the last option fixed to get a fully correct submission?

Pointer Concept Difficulty

Concept	Code	Cor	Churn	Until	Last
Assign a constant to a non-pointer	<code>v = 5;</code>	95	0.19	0.56	3
Assign a non-pointer to a non-pointer	<code>v = w;</code>	92	0.31	0.84	7
Update a non-pointer	<code>v = v + 2;</code>	94	0.21	0.62	6
Assign to a pointer	<code>pt = &w;</code>	92	0.32	0.87	10
Dereference on the LHS	<code>*pt = w;</code>	84	0.41	1.2	19
Dereference on the RHS	<code>v = *pt + 2;</code>	88	0.40	1.1	13
Dereference LHS & RHS	<code>*pt = *pt + 2;</code>	89	0.34	1.1	12

Pointer Concept Difficulty

Concept	Code	Cor	Churn	Until	Last
Assign a constant to a non-pointer	<code>v = 5;</code>	95	0.19	0.56	3
Assign a non-pointer to a non-pointer	<code>v = w;</code>	92	0.31	0.84	7
Update a non-pointer	<code>v = v + 2;</code>	94	0.21	0.62	6
Assign to a pointer	<code>pt = &w;</code>	92	0.32	0.87	10
Dereference on the LHS	<code>*pt = w;</code>	84	0.41	1.2	19
Dereference on the RHS	<code>v = *pt + 2;</code>	88	0.40	1.1	13
Dereference LHS & RHS	<code>*pt = *pt + 2;</code>	89	0.34	1.1	12

Pointer Concept Difficulty

Concept	Code	Cor	Churn	Until	Last
Assign a constant to a non-pointer	<code>v = 5;</code>	95	0.19	0.56	3
Update a non-pointer	<code>v = v + 2;</code>	94	0.21	0.62	6
Assign a non-pointer to a non-pointer	<code>v = w;</code>	92	0.31	0.84	7
Assign to a pointer	<code>pt = &w;</code>	92	0.32	0.87	10
Dereference LHS & RHS	<code>*pt = *pt + 2;</code>	89	0.34	1.1	12
Dereference on the RHS	<code>v = *pt + 2;</code>	88	0.40	1.1	13
Dereference on the LHS	<code>*pt = w;</code>	84	0.41	1.2	19

Pointer Concept Difficulty

Concept	Code	Cor	Churn	Until	Last
Assign a constant to a non-pointer	<code>v = 5;</code>	95	0.19	0.56	3
Update a non-pointer	<code>v = v + 2;</code>	94	0.21	0.62	6
Assign a non-pointer to a non-pointer	<code>v = w;</code>	92	0.31	0.84	7
Assign to a pointer	<code>pt = &w;</code>	92	0.32	0.87	10
Dereference LHS & RHS	<code>*pt = *pt + 2;</code>	89	0.34	1.1	12
Dereference on the RHS	<code>v = *pt + 2;</code>	88	0.40	1.1	13
Dereference on the LHS	<code>*pt = w;</code>	84	0.41	1.2	19

Pointer Concept Difficulty

Concept	Code	Cor	Churn	Until	Last
Assign a constant to a non-pointer	<code>v = 5;</code>	95	0.19	0.56	3
Update a non-pointer	<code>v = v + 2;</code>	94	0.21	0.62	6
Assign a non-pointer to a non-pointer	<code>v = w;</code>	92	0.31	0.84	7
Assign to a pointer	<code>pt = &w;</code>	92	0.32	0.87	10
Dereference LHS & RHS	<code>*pt = *pt + 2;</code>	89	0.34	1.1	12
Dereference on the RHS	<code>v = *pt + 2;</code>	88	0.40	1.1	13
Dereference on the LHS	<code>*pt = w;</code>	84	0.41	1.2	19

Pointer Concept Difficulty

Concept	Code	Cor	Churn	Until	Last
Assign a constant to a non-pointer	<code>v = 5;</code>	95	0.19	0.56	3
Update a non-pointer	<code>v = v + 2;</code>	94	0.21	0.62	6
Assign a non-pointer to a non-pointer	<code>v = w;</code>	92	0.31	0.84	7
Assign to a pointer	<code>pt = &w;</code>	92	0.32	0.87	10
Dereference LHS & RHS	<code>*pt = *pt + 2;</code>	89	0.34	1.1	12
Dereference on the RHS	<code>v = *pt + 2;</code>	88	0.40	1.1	13
Dereference on the LHS	<code>*pt = w;</code>	84	0.41	1.2	19

Pointer Concept Difficulty

Concept	Code	Cor	Churn	Until	Last
Dereference LHS & RHS	<code>*pt = *pt + 2;</code>	89	0.34	1.1	12
Dereference on the RHS	<code>v = *pt + 2;</code>	88	0.40	1.1	13
Dereference on the LHS	<code>*pt = w;</code>	84	0.41	1.2	19

Symmetric Dereference Easier

- After completing the lab, all four metrics showed that symmetric dereference was easier
- Analysis of lab revealed ...

Symmetric Dereference Easier

- After completing the lab, all four metrics showed that symmetric dereference was easier
- Analysis of lab revealed ...

NO questions **required** RHS or LHS dereference alone

Declaring and Assigning Pointers

```
int main(int argc, char ** argv) {  
    int friends = atoi(argv[1]);  
    char *arch_enemy = argv[2];  
    /* Create a variable called friends_ptr and  
     * set it to point to friends.  
     * Create a variable enemy_ptr and set it  
     * to point to the location where  
     * arch_enemy is stored.  
     */  
  
    return 0;  
}
```

Declaring and Assigning Pointers

```
int main(int argc, char ** argv) {  
    int friends = atoi(argv[1]);  
    char *arch_enemy = argv[2];  
    /* Create a variable called friends_ptr and  
     * set it to point to friends.  
     * Create a variable enemy_ptr and set it  
     * to point to the location where  
     * arch_enemy is stored.  
     */  
    int *friends_ptr = &friends;  
    char **enemy_ptr = &arch_enemy;  
    return 0;  
}
```

Common Declaration Errors

Error	Example	% Students
Defining incorrect pointer type		
... <i>char</i> * instead of <i>char</i> **	<code>char *enemy_ptr;</code>	81
... failing to declare a pointer	<code>char enemy_ptr;</code>	53
... <i>int</i> * instead of <i>char</i> **	<code>int *enemy_ptr;</code>	31



Common Declaration Errors

Error	Example	% Students
Defining incorrect pointer type		
... <i>char</i> * instead of <i>char</i> **	<code>char *enemy_ptr;</code>	81
... failing to declare a pointer	<code>char enemy_ptr;</code>	53
... <i>int</i> * instead of <i>char</i> **	<code>int *enemy_ptr;</code>	31

- Most of these errors are revealed on the char pointer
- They result from a guessing process for solving the exercise.

Common Declaration Errors

Error	Example	% Students
Defining incorrect pointer type		
... <i>char</i> * instead of <i>char</i> **	<code>char *enemy_ptr;</code>	81
... failing to declare a pointer	<code>char enemy_ptr;</code>	53
... <i>int</i> * instead of <i>char</i> **	<code>int *enemy_ptr;</code>	31

- Most of these errors are revealed on the char pointer
- They result from a guessing process for solving the exercise.

```
char *enemy_ptr = arch_enemy;
```

format '%s' expects argument of type 'char *', but
argument 2 has type 'int'

Common Declaration Errors

Error	Example	% Students
Defining incorrect pointer type		
... <i>char</i> * instead of <i>char</i> **	<code>char *enemy_ptr;</code>	81
... failing to declare a pointer	<code>char enemy_ptr;</code>	53
... <i>int</i> * instead of <i>char</i> **	<code>int *enemy_ptr;</code>	31

- Most of these errors are revealed on the char pointer
- They result from a guessing process for solving the exercise.

```
char *enemy_ptr = arch_enemy; →  
int *enemy_ptr = arch_enemy;
```

Type mismatch

Common Declaration Errors

Error	Example	% Students
Defining incorrect pointer type		
... <i>char</i> * instead of <i>char</i> **	<code>char *enemy_ptr;</code>	81
... failing to declare a pointer	<code>char enemy_ptr;</code>	53
... <i>int</i> * instead of <i>char</i> **	<code>int *enemy_ptr;</code>	31

- Most of these errors are revealed on the char pointer
- They result from a guessing process for solving the exercise.

```
char *enemy_ptr = arch_enemy; →  
int *enemy_ptr = arch_enemy; →  
int *enemy_ptr = *arch_enemy; →  
char enemy_ptr = arch_enemy; ...
```

Common Assignment Errors

Error	Example	% Students
Defining incorrect pointer type ... <i>char</i> * instead of <i>char</i> ** ... failing to declare a pointer ... <i>int</i> * instead of <i>char</i> **	<pre>char *enemy_ptr; char enemy_ptr; int *enemy_ptr;</pre>	81 53 31
Missing '&' operator	<pre>... = arch_enemy; ... = friends;</pre>	62 35
Extraneous '*' operator	<pre>... = *arch_enemy; *enemy_ptr = ...</pre>	25 18
Combining the '&' and '*' operators	<pre>... = &*arch_enemy;</pre>	25

Functions with Pointer Parameters

```
/* Write a void function invest that takes your  
* money and multiplies it by the rate */
```

```
int main(int argc, char ** argv) {  
    double principle = atof(argv[1]);  
    double rate = atof(argv[2]);  
    invest(&principle, rate);  
    printf("%.2f\n", principle);  
    return 0;  
}
```

- Note: Students previously solved a problem that had them *call* a very similar function.

Functions with Pointer Parameters

```
/* Write a void function invest that takes your
 * money and multiplies it by the rate */
void invest(double *v1, double v2) {
    *v1 = *v1 * v2;
}

int main(int argc, char ** argv) {
    double principle = atof(argv[1]);
    double rate = atof(argv[2]);
    invest(&principle, rate);
    printf("%.2f\n", principle);
    return 0;
}
```

Common Parameter Errors

Error	Example	% Students
Missing Dereference	<code>... = v1 * v2;</code>	27

- Okay, so the most common error isn't actually a parameter error ...

```
total = v1 * v2;
```

invalid operands to binary expression
(`'double'` and `'double *'`)

Common Parameter Errors

Error	Example	% Students
Missing Dereference	<code>... = v1 * v2;</code>	27
Dereference Double	<code>... *v2 ...</code>	11

- Okay, so the most common error isn't actually a parameter error ...

```
total = v1 * v2; →  
*total = *v1 * *v2;
```

- But it demonstrates how students preferentially apply operators symmetrically to try to resolve errors.

Common Parameter Errors

Error	Example	% Students
Missing Dereference	<code>... = v1 * v2;</code>	27
Incorrect Parameter Type		
... too few pointers	<code>invest(double, double)</code>	19
... too many pointers	<code>invest(double*, double*)</code>	14
... <i>int</i> , not <i>double</i>	<code>invest(double*, int)</code>	13
Dereference Double	<code>... *v2 ...</code>	11

- Students tended to pick non-pointer parameters, if they made an error in the parameters.
- They often attempted to correct the error by making both parameters into pointers.

Takeaways

... Students favor symmetric solutions

... and $*a = *a + b$ does not test $*a = b$ or $x = *a$

Takeaways

... Students favor symmetric solutions

... and $*a = *a + b$ does not test $*a = b$ or $x = *a$

... Many students use a guess-and-check process if their first attempt is incorrect

... The feedback they receive can lead them in very incorrect directions

Takeaways

... Students favor symmetric solutions

... and $*a = *a + b$ does not test $*a = b$ or $x = *a$

... Many students use a guess-and-check process if their first attempt is incorrect

... The feedback they receive can lead them in very incorrect directions

... Having a taxonomy is extremely valuable

... Identify topics that are missed

... Building tests that cover every topic

Summary



UNIVERSITY OF
TORONTO

Education Research and Teaching

Shifting from a model of “scholarly teaching” to one of active scholarship has changed the game for U of T

- Scholarly teaching creates opportunities for research
 - And the research performed is relevant and authentic
- Research feedback yields courses of higher quality
 - Instructors have incentives to remain “current”
 - Redesigned courses require engaged faculty

Why the Continuous Redesign?

- Research

- Initially, we were swayed by active learning research
- Later, our own research was compelling

- Opportunity

- Provincial and university funds were available for the creation of online resources

Why the Continuous Redesign?

- Research
 - Initially, we were swayed by active learning research
 - Later, our own research was compelling
- Opportunity
 - Provincial and university funds were available for the creation of online resources
- Peer and Institutional Pressure
 - The department has a strong teaching culture – and we drive each other
 - The institution made a concerted effort to reward innovation

Drop me a line!

*<http://andrewpetersen.info/>
andrew.petersen@utoronto.ca*

I'm in Auckland until mid-June and would love to chat
... especially if you know someone who is graduating

Backup Slides: Results of Flipping



UNIVERSITY OF
TORONTO

Results?

Horton, Campbell, and Craig have studied this since 2013.

1. Comparing Outcomes in Inverted and Traditional CS1

<http://dl.acm.org/citation.cfm?id=2677273>

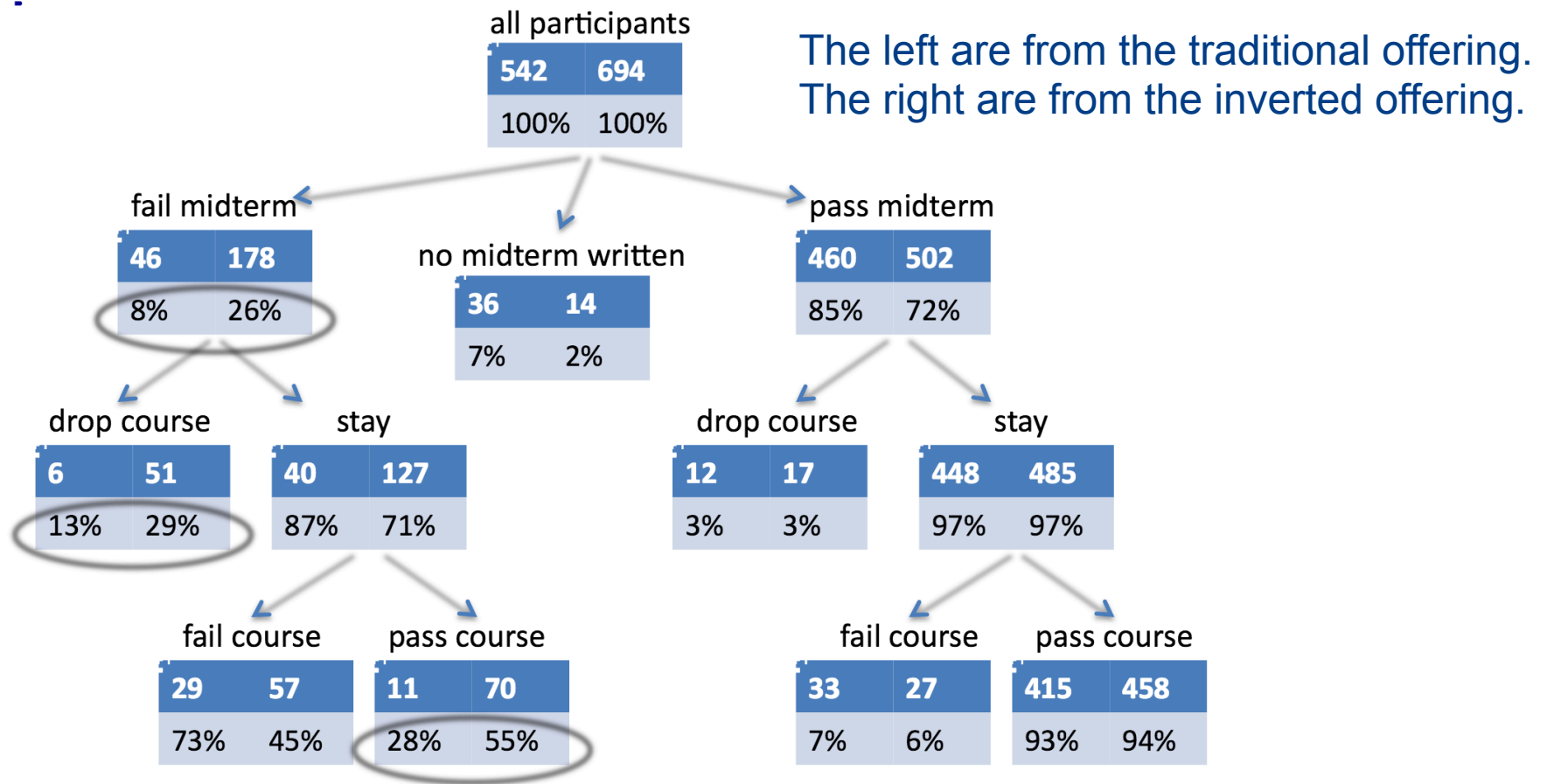
2. Online CS1: Who Enrols, Why, and How Do They Do?

<http://dl.acm.org/citation.cfm?id=2844578>

3. Drop, Fail, Pass, and Continue: Persistence in CS1 ...

<http://dl.acm.org/citation.cfm?id=2591752>

Success Rates remain Constant but the Paths to Success Differ



From Horton and Craig. "Drop, Fail, Pass, Continue: Persistence in CS1 and Beyond in Traditional and Inverted Delivery." In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 2015.

Other Observations

- Students appreciated the online materials.
 - In particular, they started valuing lecture *less* when online materials became available.
- Students in the inverted offering tended to do much better on the final, summative assessment.
- Online courses tend to attract students from outside of the major.
 - ... and enthusiasm for the course increased, too.

Reflections

- The cost of producing materials and running these courses is significant.
 - Early efforts were not appropriately resourced.
- We've had to develop in-house experience in developing tools and online content.
 - Central support is useful – but local expertise is necessary, too.
- As of 2016, we appear to be entering a new phase of *re-development*.
 - We're not just tweaking: courses are being redesigned to take advantage of new resources.