COMPUTATIONAL CREATIVITY EXERCISES FOR IMPROVING STUDENT LEARNING AND PERFORMANCE

Leen-Kiat Soh¹, Markeya Peteranetz²

¹ Department of Computer Science and Engineering
² College of Engineering
University of Nebraska, Lincoln, NE
http://cse.unl.edu/agents/ic2think
RATIONALES

Increasingly more CS courses are taught to majors and non-majors with diverse backgrounds and motivations.

The Computational Creativity Exercises and our approach to integrate computational thinking and creative thinking into classrooms have been shown to improve student learning and performance in class.

Furthermore, the group-based, non-programming active learning activities also help:

• encourage exchange of different ideas
• put group members in more equal footings, and
• engage students in “thinking” about the process of coming up with a solution.
IN THIS WORKSHOP ...

You will learn about how to integrate computational thinking and creative thinking activities that have been shown to significantly improve student learning and performance in their classes via rigorous research investigations.

In particular, participants will

• be familiarized with the suite of Computational Creativity Exercises, practice hands-on how to complete such an exercise
• learn how to integrate and adapt them into their courses
• be exposed to the educational research studies behind the development, design, and administration of these exercises
WE ARE …

Leen-Kiat Soh is a Professor with the Department of Computer Science and Engineering whose research in Computer Science Education focuses on CS1/2, integrating creative thinking, computer-assisted instruction, intelligent education tools, instructional technology, and curriculum.

Markeya Peteranetz is the Learning Assessment Coordinator in the College of Engineering. She works on a variety of assessment and evaluation projects and supports faculty engaged in education research by providing expertise on research methodology, measurement, and statistical analysis. Her research interests include motivation, self-regulation, and instructional interventions at the college level.
THE PROJECT FUNDING ...

This material is based upon work supported by the National Science Foundation grants no. DUE-1122956 (NSF TUES) and DUE-1431874 (NSF IUSE)

Additional support was provided by a UNL Phase II Pathways to Interdisciplinary Research Centers grant
# AGENDA

<table>
<thead>
<tr>
<th>Session</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>• Presentation-based</td>
</tr>
<tr>
<td></td>
<td>• 30 minutes</td>
</tr>
<tr>
<td>Trial-Run: Pathfinding</td>
<td>• Hands-on, Group Activities</td>
</tr>
<tr>
<td></td>
<td>• Report Back</td>
</tr>
<tr>
<td></td>
<td>• Sharing insights and lessons learned</td>
</tr>
<tr>
<td></td>
<td>• 70 minutes</td>
</tr>
<tr>
<td>Research: Findings &amp; Instruments</td>
<td>• Presentation-based</td>
</tr>
<tr>
<td></td>
<td>• 40 minutes</td>
</tr>
<tr>
<td>Integration: Classroom</td>
<td>• Short presentation</td>
</tr>
<tr>
<td></td>
<td>• Q&amp;A discussion</td>
</tr>
<tr>
<td></td>
<td>• 40 minutes</td>
</tr>
</tbody>
</table>
PREMISE

By blending computational and creative thinking students can leverage their creative thinking skills to “unlock” their understanding of computational thinking

• *Should be able to improve student learning and achievement in CS courses*

• *Should be able to make computational thinking more generally applicable to STEM and non-STEM disciplines*
COMPUTATIONAL THINKING

A way of thinking for logically and methodically solving problems

- E.g., *purposeful*, *describable*, *replicable*

Includes *skills* such as

- Decomposition
- Pattern Recognition
- Abstraction
- Generalization
- Algorithm Design
- Evaluation
CREATIVE THINKING

Epstein’s Generativity Theory breaks creative thinking down to four core competencies

• *Capturing* novelty
• *Challenging* established thinking and behavior patterns
• *Broadening* one’s knowledge beyond one’s discipline
• *Surrounding* oneself with new social and environmental stimuli
## EXERCISE DESIGN | PRINCIPLES

### BALANCING OF ATTRIBUTES

<table>
<thead>
<tr>
<th>Computational Thinking</th>
<th>Creative Thinking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convergent thinking</td>
<td>Surrounding with new social and environmental stimuli</td>
</tr>
<tr>
<td>Linear and sequential “flow”</td>
<td>Challenging established solutions and algorithms</td>
</tr>
<tr>
<td>Rational &amp; logical processes</td>
<td>Broadening possible solutions through additional paradigms</td>
</tr>
<tr>
<td>Methodical</td>
<td>Capturing novelty and spontaneous outputs</td>
</tr>
</tbody>
</table>
EXERCISES | OVERALL DESIGN

- **Objectives:** Computational, Creative, and Collaborative
- **Tasks:** 2-3 weeks
- **CS Lightbulbs**
- **Questions:** Analysis and Reflections
## EXERCISE EXAMPLES 1

<table>
<thead>
<tr>
<th>Name</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Everyday Object</td>
<td>Identify an “everyday” object (such as nail clipper, a paper clip, Scotch tape) and describe the object in terms of its inputs, outputs and functionalities</td>
</tr>
<tr>
<td>Cipher</td>
<td>Devise a three-step encoding scheme to transfer the alphabet letters into digits and encode questions for other teams to compete to decode</td>
</tr>
<tr>
<td>Story Telling</td>
<td>Develop a chapter (100-200 words) individually and independently in week 1 and work as a team in week 2 to resolve all conflicts or inconsistencies</td>
</tr>
<tr>
<td>Exploring</td>
<td>Explore sensory stimuli at a particular site (sounds, sights, smell, etc.) and document observations</td>
</tr>
</tbody>
</table>
## EXERCISE EXAMPLES 2

<table>
<thead>
<tr>
<th>Name</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simile</td>
<td>Poses “simile” descriptions and participate in team-to-team Q&amp;As to solicit guesses and descriptions relevant to a particular object</td>
</tr>
<tr>
<td>Machine Testing</td>
<td>Devise ways to test a black-box mysterious machine without causing harm to humans while attempting to reveal the functionalities of the machine</td>
</tr>
<tr>
<td>Calendar</td>
<td>Build a calendar for a planet with two suns, four different cultural groups with different resource constraints and industrial needs</td>
</tr>
<tr>
<td>Big Five Profiles</td>
<td>Revises a text snippet such that at least one the text snippet’s Big Five profile changes significantly</td>
</tr>
<tr>
<td>Dividing Alphabet</td>
<td>Finds a rule to divide up the alphabet letters based on some sample data points on how some initial letters are divided</td>
</tr>
</tbody>
</table>
**EXERCISE EXAMPLES 3**

<table>
<thead>
<tr>
<th>Name</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Path Finding I</strong></td>
<td>Create a step-by-step instruction on drawing lines to create a quilt pattern on a n x n grid and identify similar structures in other teams’ quilt patterns</td>
</tr>
<tr>
<td><strong>Path Finding II</strong></td>
<td>Use rotation, reflection, and loop to generate a more complex quilt pattern based on simpler base pattern</td>
</tr>
<tr>
<td><strong>Marble Maze I</strong></td>
<td>Each team member creates a sub-structure allowing a marble to travel at least for n seconds in week 1 and the team puts all sub-structures together to make a super-structure in week 2</td>
</tr>
<tr>
<td><strong>Marble Maze II</strong></td>
<td>Teams are broken up and now must adapt their own sub-structure to work with other sub-structures in their new teams</td>
</tr>
<tr>
<td><strong>Marble Maze III</strong></td>
<td>All teams bring together their super-structures and build a mega-structure</td>
</tr>
</tbody>
</table>
K-12 version of several of our lesson modules (i.e., computational creativity exercises) can be found at Google's Exploring Computational Thinking website https://www.google.com/edu/computational-thinking

- Ciphering a Sentence, Describing an Everyday Object, Exploring Your Environment, Machine Testing, Writing a Story, Solving a Guessing Game with Data

Versions also available on Ensemble’s Computing Portal (search key: IC2THINK) http://www.computingportal.org

Versions also available on EngageCSEdu (search key: Computational Creativity) https://www.engage-csedu.org
FINDINGS (BRIEF) 1

The findings support our contention that the incorporation of CCEs can improve learning of computational thinking and achievement in CS courses

- CCEs appear to have an *independent instructional effect*. In all studies, the effect was distinct from general student achievement as students’ overall GPA was *controlled*

We believe the CCEs impact student achievement and learning because they make students deal with computational principles and skills abstracted from coding

- Enhances ability to connect their CS and computational thinking knowledge to more diverse applications
FINDINGS (BRIEF) 2

The effect of the CCEs does not appear to depend on individual student motivation or engagement

• *If students do the CCEs, they benefit, whether or not they are self-aware of the benefit*
REFERENCES


# AGENDA

<table>
<thead>
<tr>
<th>Session</th>
<th>Format</th>
</tr>
</thead>
</table>
| **Introduction**                 | • Presentation-based  
                                   | • 30 minutes                                           |
| **Trial-Run: Pathfinding**       | • Hands-on, Group Activities  
                                   | • Report Back                                           |
|                                  | • Sharing insights and lessons learned  
                                   | • 70 minutes                                           |
| **Research: Findings & Instruments** | • Presentation-based  
                                   | • 40 minutes                                           |
| **Integration: Classroom**       | • Short presentation  
                                   | • Q&A discussion                                       |
|                                  | • 40 minutes                                           |
Using a grid, you will be designing a geometric visual pattern. You will then create a set of written instructions (an algorithm) for another group to follow which will accurately generate your geometric pattern on another grid.

Here is a sample grid with a base pattern. In this example, the right triangle repeats six times with different rotations. It has 23 line segments, or 23/42 = 55% of all possible line segments.

1.2. **Generate Written Instructions**
Once you have created your base pattern, write the instructions to generate it on another grid. The first step of your instructions must start at (0,0); subsequent steps may start at any intersection. Here are some sample instructions for generating this example base pattern by drawing:

**Initial Set of Instructions**
1. Start at (0,0) and draw to (0,3)
2. Start at (0,0) and draw to (3,0)
3. Start at (1,0) and draw to (1,1) (and so forth... . . . )
PATHFINDING 1 | Part 2

Each group follows the written instructions of another group by drawing on a blank grid labeled following the same (0,0) numbering.

When you think you have followed the other group’s instructions correctly, post an image of the other group’s base pattern (a jpg of your grid drawing) on that group’s page.

If you think that the other group’s instructions are invalid and you cannot complete the pattern, post your reasoning on that group’s page along with an image of your incomplete pattern.
The proper use of **boundary conditions** is extremely important in computer science. When small errors in the boundary conditions are made, such as a mistake in array indexing or an improperly used pointer, the source code that you write may not compile or may generate an error message. These condition mistakes pales in comparison to the errors that can arise from a lack of proper debugging. In real-world programming projects, these mistakes can take on a whole new meaning. For example, when they are found in an application that returns the wrong output to the end users, the consequences can be severe for the developers and the company they work for. Developers must be able to look at in-line comments or source code and understand immediately the purpose for the module in question. However, real-world applications often contain thousands or even millions of lines of code making this understanding impossible to achieve. In these real-world applications, modules lacking a clear set of instructions are extremely difficult to use properly, and may result in missed deadlines or additional expense for a company. Furthermore, one must always keep in mind that the end users of the vast majority of real-world applications are not software developers. These end users need a clear set of instructions on how to use each module in the application or they will become upset very quickly and stop using it. Nevertheless, both developers and end users are often impatient and unwilling to spend hours reading hundreds of pages of instructions. As a result, programmers need to think creatively on how to provide a set of instructions that provides all the details while being as concise as possible.
PATHFINDING 1 | Analysis & Reflection

These questions give you

• A chance to dig deeper
• A chance to look at the process you went through in Parts 1 & 2
• A chance to connect the exercise activity to computer science (or to another discipline)

Analysis [10 points]. Respond to these questions: (1) Analyze your group’s instructions. Are they ambiguous? Can they be simplified? How could you simplify your instructions using loops? (2) If you combine two sets of instructions, your group’s and another group’s, to create a composite pattern, how would you make the combination a more efficient set by removing steps that draw the same line segments? Would you be able to retain certain subsets of steps without changing them at all? Why or why not?
PATHFINDING 1 |
Analysis & Reflection, Cont’d ...

Don’t try to do too much

• Aim for 3-5 relevant sentences!
• When students do these, they will also respond to another student giving them the opportunity to expand their own thinking

Reflection [10 points]. Respond to these questions: (1) These instructions for drawing patterns are essentially steps of an algorithm. Do you see parts of all these algorithms (including those of the other groups) that are common (i.e., creating the same shape or same combination of line segments)? If yes, please highlight them here. If no, please explain. (2) Again, these instructions for drawing patterns are essentially steps of an algorithm. Do you see parts of your pattern repeated in other groups’ patterns? You should. Please identify at least one part that consists of more than four segments. Is this part achieved using the same “sequence of steps” in other groups’ instructions? Likely it is not. Please explain why different groups designed their instructions differently for the same common sub-pattern.
PATHFINDING 1 | Evaluation

- What surprised you doing this exercise?
- What was challenging?
- What did you learn?
- How might this activity relate to your teaching? Your course? Your students?
PATHFINDING 2 | Part 1

Using a base 3 x 3 pattern from Exercise Path Finding I, build a complex 9 x 9 pattern using three operations:

• The *shift* operation allows a base pattern to be replicated at a different location

• The *reflect* operation allows a base pattern to be reflected horizontally or vertically

• The *rotate* operation allows a base pattern to be rotated 90 degrees clockwise or counterclockwise
Figure 1: Examples of Base Pattern along with Reflect, Rotate and Shift Down.

Note that depending on your pattern, rotating or reflecting may produce the same visual result.

You are also allowed to use a loop structure to repeat to generate multiple shifts, reflections, or rotations. For example,

Loop 3 times

   Rotate-counter-clockwise()

End Loop

In the above example, a base pattern will be rotated 3 times counter-clockwise.
# AGENDA

<table>
<thead>
<tr>
<th>Session</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>• Presentation-based</td>
</tr>
<tr>
<td></td>
<td>• 30 minutes</td>
</tr>
<tr>
<td>Trial-Run: Pathfinding</td>
<td>• Hands-on, Group Activities</td>
</tr>
<tr>
<td></td>
<td>• Report Back</td>
</tr>
<tr>
<td></td>
<td>• Sharing insights and lessons learned</td>
</tr>
<tr>
<td></td>
<td>• 70 minutes</td>
</tr>
<tr>
<td>Research: Findings &amp; Instruments</td>
<td>• Presentation-based</td>
</tr>
<tr>
<td></td>
<td>• 40 minutes</td>
</tr>
<tr>
<td>Integration: Classroom</td>
<td>• Short presentation</td>
</tr>
<tr>
<td></td>
<td>• Panel Q&amp;A discussion</td>
</tr>
<tr>
<td></td>
<td>• 40 minutes</td>
</tr>
</tbody>
</table>
## METHODS

### General Methods

<table>
<thead>
<tr>
<th>Data</th>
<th>Details</th>
</tr>
</thead>
</table>
| Multiple Web-Surveys of Motivation, Studying, Self-Directed Learning | • Surveys done Beginning (Motivation only), Middle, and End of the semester  
• Done in-class                                                  |
| Knowledge Test of CS and computational thinking                    | • Based on CS-1 curriculum                                             
• Done end of semester as part of end-of-semester survey        |
| Course grades                                                       | • Obtained from course instructors after semester                       |
| Overall Grade Point Average and Subsequent Course Enrollment       | • Obtained from University Records                                     |
## METHODS

### RESEARCH | Approaches

<table>
<thead>
<tr>
<th>Phases</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Evidence</td>
<td>• No comparison (&quot;control&quot;) group</td>
</tr>
<tr>
<td></td>
<td>• Exercise completion was voluntary</td>
</tr>
<tr>
<td></td>
<td>• Compared students completing different numbers of CCEs</td>
</tr>
<tr>
<td></td>
<td>• Controlled for things like motivation and general academic performance</td>
</tr>
<tr>
<td>Second “Wave” of Studies</td>
<td>• Intervention vs. control comparisons</td>
</tr>
<tr>
<td></td>
<td>• No random assignment (difficult in classroom research)</td>
</tr>
<tr>
<td></td>
<td>• Quasi-experimental</td>
</tr>
<tr>
<td></td>
<td>• Controlled for things like motivation and general academic performance</td>
</tr>
<tr>
<td>Most Recent Research</td>
<td>• More rigorous designs</td>
</tr>
<tr>
<td></td>
<td>• Propensity score matching (PSM)</td>
</tr>
<tr>
<td></td>
<td>• Closely aligned intervention and control sections</td>
</tr>
</tbody>
</table>
Quick overview of constructs measured and example items.
Computational thinking knowledge test developed by CSCE faculty

- 13 conceptual and problem-solving questions based on the core content from CS-1 classes
- Standardized, non-grade outcome that can be used to compare across courses, instructors, and semesters
2. When solving problems, what are the benefits of using repetition statements?

I. A repetition statement allows one to control a block of code to be executed for a fixed number of times or until a certain condition is met.

II. A repetition statement allows one to repeat some sequence of operations repeatedly, thus saving the number of lines of code.

III. A repetition statement allows one to represent a loop logically.

IV. A repetition statement allows one to divide a set of operations into different groups to handle different situations.

A. I
B. I & II
C. I, II, & III
D. I, II, III, & IV
7. When we decide whether to use a function, which of the following should we consider?

I. Are there places where you are performing the same action more than once?

II. Are there places where you are performing almost the same action on different inputs?

III. How can you break this problem into smaller pieces? Can you break it down even more?

IV. Are there small problems whose outputs will be used as input to a larger problem?

V. Are there loops?

A. I, II, & IV
B. I, II, & III
C. I, III, & IV
D. I, II, III, & IV
Please rate how confident you are about your knowledge of or ability to do each of the following on a scale from 0 (Completely Unconfident) to 100 (Completely Confident).

- Your ability to use computational algorithms to solve problems in your field.
- Your ability to conceptualize data in your field in ways that can be analyzed computationally.
# Course Goals-Class Goal Orientation Scale

**Learning** | **Performance** | **Task (Work)**
--- | --- | ---
**Approach** | **Avoid** | **Approach** | **Avoid** | **Approach** | **Avoid**
Learning new knowledge or skills in the class just for the sake of learning them | Remembering material long enough to get through the tests after which you can forget about it | Doing better than the other students in the class on tests and assignments | Keeping others from thinking you are dumb | Doing my best on course assignments and tests | Getting through the course with the least amount of time and effort;
Really understanding the class material | Getting this course done even though you don’t care about the content | Impressing the teacher/instructor with your performance | Avoiding looking like you don’t understand the class material | Getting a good grade in the class | Getting a passing grade with as little studying as possible
### Future Time Perspective Scale [Husman & Shell]

<table>
<thead>
<tr>
<th>Career Connectedness</th>
</tr>
</thead>
<tbody>
<tr>
<td>One should be taking steps today to help realize future career goals</td>
</tr>
<tr>
<td>What will happen in the future in my career is an important consideration in deciding what action to take now.</td>
</tr>
</tbody>
</table>
**Perceptions of Instrumentality Scale** [Husman & Hilpert]

<table>
<thead>
<tr>
<th>Endogenous Instrumentality</th>
<th>Exogenous Instrumentality</th>
</tr>
</thead>
<tbody>
<tr>
<td>I will use the information I learn in this CS class in the future.</td>
<td>The only thing useful to me in this class is the grade I get.</td>
</tr>
<tr>
<td>What I learn in this CS class will be important for my future occupational success.</td>
<td>The only aspect of this class that will matter after graduation is my grade.</td>
</tr>
</tbody>
</table>
**RESEARCH | Motivation - Career**

<table>
<thead>
<tr>
<th>Career Aspirations [PISA]</th>
<th>Career Information [PISA]</th>
</tr>
</thead>
<tbody>
<tr>
<td>In whatever career I choose, I would like to work in a job that uses computer science applications, programming, or computational thinking.</td>
<td>How much do you know about careers and jobs that involve computer science applications, programming, or computational thinking.</td>
</tr>
<tr>
<td>In my career, I would like to work with projects that involve a lot of computer science applications, programming, or computational thinking.</td>
<td>How much do you know about what has to be done in order to get a job in your chosen career that uses computer science applications, programming, or computational thinking.</td>
</tr>
</tbody>
</table>
### Implicit Theories of Intelligence Scale [Dweck] vs. Implicit Theories of CS Ability Scale [project team]

<table>
<thead>
<tr>
<th>Incremental</th>
<th>Entity</th>
<th>Incremental</th>
<th>Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>No matter who you are, you can significantly change your intelligence level.</td>
<td>Your intelligence is something about you that you can't change very much.</td>
<td>No matter how much CS or programming ability you have, you can always improve it quite a bit.</td>
<td>You have a certain amount of programming and CS ability and you can't change it no matter how much you learn about CS.</td>
</tr>
</tbody>
</table>
### In this class, I tried to determine the best approach for studying each assignment.

- **In this class, I tried to determine the best approach for studying each assignment.**
- **In this class, I tried to examine what I was learning in depth.**
- **In this class, I couldn’t figure out how I should study the material.**

### In this class, I tried to monitor my progress when I studied.

- **In this class, I tried to monitor my progress when I studied.**
- **As I studied a topic in this class, I tried to consider how the topic related to other things I know about.**
- **In this class, I relied on someone else to tell me what to do.**
<table>
<thead>
<tr>
<th>High-level</th>
<th>Low-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>In this class, I asked questions to more fully understand the topics we were learning.</td>
<td>In this class, I asked questions so that I could find out what information the instructor thought was important.</td>
</tr>
</tbody>
</table>
• Please indicate the average number of hours per week you spend studying for YOUR CLASS.
  1 = Less than 2 hours
  2 = 2 to 4 hours
  3 = 4 to 6 hours
  4 = 6 to 8 hours
  5 = 8 to 10 hours
  6 = 10 to 12 hours
  7 = over 12 hours

• Please indicate which of the following best describes your own perception of the effort you put forth studying for YOUR CLASS.
  1 = I put forth much less effort studying than most students.
  2 = I put forth somewhat less effort studying than most students.
  3 = I put forth about the same effort studying as most students.
  4 = I put forth somewhat more effort studying than other students.
  5 = I put forth much more effort studying than other students.
### Positive and Negative Affect Scale (PANAS) 
[Watson & Tellegen]

<table>
<thead>
<tr>
<th>Positive Emotions</th>
<th>Negative Emotions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interested, excited, inspired, proud</td>
<td>Upset, frustrated, distressed, nervous</td>
</tr>
</tbody>
</table>
• Found a “dosage” effect that higher grades and learning of core CS content were associated with increasing CCE completion from 0-1 to 4 CCEs. [FIE 2013]

• We found that the dosage effect was consistent for CS majors and non-CS majors, and for upper class and freshmen students. [SIGCSE 2014]
RESEARCH | Initial Evidence

Lower Division (100-200 Level)
- CS1: Engineering & Science Focus (3 CCEs)
- Computer Organization (4 CCEs)

Upper Division (300-400 Level)
- Data Structures and Algorithms (3 CCEs)
- Programming Language Concepts (3 CCEs)
- Automata (2 CCEs)

Found same “dosage” effect that higher grades and learning of core CS content were associated with increasing CCE completion. [SIGCSE 2017]
RESEARCH | Second-Wave

Implementation semester: *Spring 2013, with students completing three or four CCEs*

Control semester: *Fall 2013, same CS1-Engineering course*

Students in the CCE implementation semester had higher knowledge test scores and higher self-efficacy than students in the “control” semester with no CCEs. [FIE 2014]
Dosage effect and quasi-experimental findings are consistent and compelling

• But, the correlational and quasi-experimental designs used do not allow for strong causal conclusions about the CCE effects

Challenge. Because our work is in real classes, we do not have the option of randomly assigning students to classes or semesters
Propensity Score Matching (PSM) is one of the most powerful alternative research design to randomized trials for studying causal effects.

- When random assignment is not possible, PSM can be used to equate intervention and control groups by matching students in the intervention and control groups based on multiple characteristics known to be associated with the outcomes of interest.
RESEARCH | Recent Studies

Study 1 [SIGCSE 2018]
- All course levels
- Implementation during Fall 2015; Control group from Fall 2014 & Spring 2015

Study 2 [IEEE Trans. on Ed., 2018]
- Implementation and control were two sections of CS1 for engineers in the same semester
- Instructors worked together to closely align instruction and assessment during the semester
Students in the implementation semester had higher course grades, knowledge test scores, and self-efficacy, but not higher creative competency, than PSM-matched students in the control semesters. The effect of the CCEs was consistent across lower- and upper-division courses [SIGCSE 2018]

<table>
<thead>
<tr>
<th></th>
<th>Control</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standardized Course Grade</td>
<td>.01 1.03</td>
<td>.42 .64</td>
</tr>
<tr>
<td>Knowledge Test</td>
<td>6.68 3.12</td>
<td>7.84 3.36</td>
</tr>
<tr>
<td>Creative Competency</td>
<td>2.51 .69</td>
<td>2.47 .84</td>
</tr>
<tr>
<td>Self-efficacy</td>
<td>63.11 21.81</td>
<td>70.88 18.41</td>
</tr>
</tbody>
</table>
Students in the CCE implementation section had higher knowledge test scores, but not higher self-efficacy than students in the PSM matched “control” section. [IEE Trans. On Ed. 2018]
## AGENDA

<table>
<thead>
<tr>
<th>Session</th>
<th>Format</th>
</tr>
</thead>
</table>
| **Introduction**                 | • Presentation-based  
                                  | • 30 minutes                                                          |
| **Trial-Run: Pathfinding**       | • Hands-on, Group Activities  
                                  | • Report Back  
                                  | • Sharing insights and lessons learned  
                                  | • 70 minutes                                                          |
| **Research: Findings & Instruments** | • Presentation-based  
                                  | • 40 minutes                                                          |
| **Integration: Classroom**       | • Short presentation  
                                  | • Q&A discussion  
                                  | • 40 minutes                                                          |
IMPLEMENTATION

Implementing UNL’s Computational Creativity Exercises

The Computational Creativity Project

Leen-Kiat Soh, Elizabeth Ingraham, Duane Shell, Markeya Peteranetz and Lee Dee Miller, University of Nebraska-Lincoln

Abstract

This teaching paper provides an overview of the Computational Creativity Exercises (CCEs) that can be used in introductory computing courses to improve student learning and performance in computational thinking. The exercises are group-based, unplugged activities. This encourages students to practice thinking and communicating their ideas, without having to worry about coding and syntax, and be constrained to the programming environment. The goal of this paper is to provide other instructors with guidance on implementing these exercises by addressing logistical issues and providing tips for adopting and adapting for individual classroom needs. As part of this overview, we summarize research findings from our multi-year studies on the impact of these exercises.

1. INTRODUCTION

The premise behind integrating computational thinking and creative thinking is that they complement each other in improving student learning and performance in class [1]. Whereas computational thinking brings a structured, convergent, and analytic approach to problem-solving situations, creative thinking introduces novelty and innovative, divergent, non-standard solutions. While numerous components of computational thinking have been identified (e.g., [2]), the skills that we focus on in these exercises are abstraction, algorithmic thinking, evaluation, generalization, pattern recognition, and problem decomposition. The theory of creativity underlying these Computational Creativity Exercises (CCEs) is
LOGISTICS | TIPS 1

When exercises were fully integrated into the class, students enjoyed them or appreciated the rationales behind the exercises

• Discuss each exercise in class (5-10 minutes) going through the assignment

• Explicitly map activities in exercise to topics in class

• Relate both computational and creative objectives to real-world problems

• Count 3-5% towards the final course grade depending on the number of exercises you assign
Group management: may need to revise groups due to lack of participation from some students

Assign “team coordinators” or have groups divide up key tasks (managing deadlines, evaluating the final product, submitting the assignment)

Reflection and analysis questions help students gain insights and “meta-knowledge”

Students might “resist” the exercises as non-technical, or non-CS

Adapt lightbulbs to meet your needs
ISSUES & CHALLENGES | Doing The Exercises

BAD NEWS

• There is friction
• Many students dislike group work, regardless of what it is
• Online responses have inherent limits
• Online work has a guaranteed level of frustration
**GOOD NEWS**

- Considering different points of view and resolving conflicts is part of the exercise objectives.
- Learning how to have “creative abrasion” while avoiding interpersonal conflict is a key workplace skill*
- Groups can tackle bigger tasks.
- Randomly assigned groups DO offer diversity and CAN allow every student to speak

ISSUES & CHALLENGES | Doing The Exercises

A POSSIBLE MODIFICATION

- Consider giving groups the opportunity to meet in class
- Even 10 minutes can go a long way to encourage community and to get questions to surface so that the procedure can be clarified
ISSUES & CHALLENGES | Connecting to course content

- Exercises MUST be graded to get student participation
- Providing context to students (even 10 minutes) IN CLASS and pointing out direct connections to course content BEFORE and AFTER the exercise will increase students’ positive perception of the activity and their participation.
ISSUES & CHALLENGES | Grading

GOOD NEWS

• Grading is easy and Yoda-esq (“Do or do not. There is no try.”)
• Rubrics require evidence of participation in the exercise activity (e.g. labeled contribution or version history)
• Analysis and Reflection requires 3-5 relevant sentences and a response to another student’s reflection/analysis, also 3-5 relevant sentences
• Student has to post before seeing others' response.
ISSUES & CHALLENGES | Benefits

More good news

• IF students DO the exercises they WILL benefit
• They are likely to improve their learning and achievement in the course
• They will get hands-on experience with effective collaboration techniques
ISSUES & CHALLENGES | Bottom Line

To do these exercise most effectively, students and teachers need to

• Build community
• Encourage communication
• Work together
• Allow play
• Allow—use, welcome—failure and “not-knowing” . . . a beginner’s mind
MORE ...

lksoh@cse.unl.edu

http://cse.unl.edu/agents/ic2think