Welcome to our webinar!

• Please introduce yourself in the chat window:
  – Your name?
  – Your position?
  – Your city/state?
  – Number of years teaching middle school CS?
  – Other subject(s) taught?
  – Experience as a CS PD trainer?
  – Anything else you’d like for us to know!

We want to HEAR from you! Please choose the first option in the Audio Conference window (Dial-out) in order to communicate verbally. If unable to receive an incoming call, please choose one of the other options listed.
An Assessment for Introductory Programming Concepts in Middle School Computer Science

Shuchi Grover, Ph.D. (@shuchig)
Senior Research Scientist, Looking Glass Ventures
Visiting Scholar, Stanford University
July 25, 2019
Acknowledgements

SRI International

Looking Glass Ventures

Daisy Rutstein, Satabdi Basu, Matthias Hauswirth, Michelle Huynh, Kelly Powers, Todd Lash, Phil Bagge

Classroom teachers in SF and NYC

Bryan Twarek, Kathi Fisler, Judit Moscovich, Jeremy Roschelle

NSF #1543062
Roadmap

• Why assess (summatively)? To what end?
• What should we assess?
• How? (brief look into process, skills targeted)
• The MSCS assessment (VELA NSF Project)
• Examples of items (questions)
• Student performance in CS classrooms
(Summative) Assessment/Measurement in the service of understanding student learning, curriculum (re-)design, teacher preparation..

Providing evidence needed to draw inferences of what students understand/know and are able to do at the end of a term studying introductory programming.
Summative Assessment: To What End?

- Goals of CSForALL include learning concepts and skills but also:
  - Broadening participation and among women and URM s
  - Promoting equity, engagement, and interest development
  - Shape learner STEM/CS identities
  - Multi-faceted, multi-pronged assessment strategy (Grover, 2017)

- CSForALL (rightly!) resists the move to standardized testing

BUT.... YOU CANNOT IMPROVE IF / WHAT YOU DON’T MEASURE

- New curricula, new frameworks, new standards, newly trained teachers (NSF and industry funding) → need for feedback
  - Promote deeper learning/understanding (rigor is part of equity – CSForAll.org)
  - Understand and address student misconceptions
  - Refine curriculum and pedagogy
  - Measure students on some (objective) outcomes

- Measurement provides feedback on coverage and how well students are learning to teachers, districts, curriculum designers, funders.
Challenges Unique to K-12 CS Science/Programming

• Challenge #1: New domain/uncharted territory
  • What is important for middle school students to know and be able to do at the end of an introductory programming course? (Standards are addressing that to some extent.)

• Challenge #2: Developing good forms of assessment
  • How can we develop tasks/situations that elicit evidence of understanding of introductory programming in middle school?
  • Programming typically happens in open-ended programming environments (block- or text-based)

• Challenge #3: Eliciting valid evidence
  • Does the evidence support the inferences we want to make about student understanding of intro programming concepts?
An Assessment Focused on Intro Programming Concepts

• Learning to program is difficult (decades of research)
  • Syntax
  • Semantics (conceptual)
  • Practices

• Most gaps in understanding are related to conceptual issues
  • Block-based programming environments are engaging, help with syntactical issues but conceptual difficulties persist

• Programming projects are more “authentic” very time-consuming to score (scoring is subjective, not always an accurate indicator)
  • Such an assessment should be one of many forms of assessment (“Systems of Assessment” (Grover, 2017))
Operationalize the Domain of Middle School Introductory Programming

- K-12 National Framework
- ECS & AP CS Principles
- CSTA K-12 CS Standards
- Experts in the field
- CS Education Research

Introductory Programming

- Loops
- Variables
- Boolean Operators
- Conditionals
- Abstraction (Patterns)
- Expressions (Boolean, Arith., Relational)
Assessments designed at SRI for the Exploring Computer Science high school curriculum (PACT project) with ONE KEY DISTINCTION—MSCS (VELA) Assessment is not aligned to any one specific curriculum
Design begins with the claims we want to make about student learning.

In order to support claims about learning, we must gather evidence.

Tasks are designed to elicit evidence about student learning to support claims.
Evidence-Centered Design: A Principled Approach

1. **Domain Analysis**: What is important about this domain? What work and situations are central in this domain? What knowledge representations are central?

2. **Domain Modeling**: How do we represent key aspects of the domain in terms of assessment argument? *Conceptualization*

3. **Conceptual Assessment Framework**: Design structures: student, evidence, and task models. *Generativity*

4. **Assessment Implementation**: Manufacturing “nuts & bolts”: authoring tasks, automated scoring details, statistical models. *Reusability*

5. **Assessment Delivery**: Students interact with tasks, performances evaluated, feedback created. *Four-process delivery architecture*

*Mislevy, 2011; Mislevy & Risconscente, 2006*
A middle school CS (programming) curriculum will/should ideally aim for students to learn/understand...

- how programs are executed sequentially
- how simple loops work (fixed number of repetitions)
- how to create different pathways in programs using conditional statements
- algorithmic flow of control—how instructions are executed in sequence even when there are loops, except that the set of instructions within a loop are repeated
- that in control structures (like loops and conditionals), a collection of an arbitrary number of statements can be declared to act as a single statement by grouping them
- what Initialization (in general and of variables, specifically) is, why it is needed, and how to initialize
- that variables can only hold one value at a time
- how types define the set of values a variable can have, and the set of operators that can be used
- how variables are created, used, assigned values, and updated
- how variable values change within loops
- how to use expressions to make new variables from old ones
- Boolean variables, operators & expressions
- the idea of controlling loops and conditionals using Boolean conditions (may/may not involve variables and expressions)
- Boolean as a data type
- identifying and articulating patterns in real-world phenomena and problems, and abstracting them into structural components of a program (preconditions, repeating logic in a loop, any postcondition)
- how variables are an abstraction or representation of data in the program and the real world
- the importance of planning before programming
- the need for breaking down problems into smaller manageable tasks
- computational solutions are abstractions; and that these abstractions can be represented in different ways
- identifying and articulating patterns and abstracting them into structural components of a program.

<table>
<thead>
<tr>
<th>Ability to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Describe the execution that is performed by a (specific) given sequence of instructions (not incl. loops)</td>
</tr>
<tr>
<td>2. Describe what a given loop is doing</td>
</tr>
<tr>
<td>3. Describe the sequence that is executed in a given program when the program contains things inside and outside of the loop.</td>
</tr>
<tr>
<td>4. *Knowledge that a loop involves a repeating pattern, that will terminate under a specified condition or after a certain number of repetitions</td>
</tr>
<tr>
<td>5. Identify the repeating pattern within a loop</td>
</tr>
<tr>
<td>6. Describe the structural components of a pattern</td>
</tr>
<tr>
<td>7. Identify a pattern in a situation (could be a real-world phenomenon)</td>
</tr>
<tr>
<td>8. Describe how a conditional pathway would operate or create a conditional pathway</td>
</tr>
<tr>
<td>9. Create variables, assign values and update variables</td>
</tr>
<tr>
<td>10. Describe how a variable changes values in a loop</td>
</tr>
<tr>
<td>11. Use a variable within a loop</td>
</tr>
<tr>
<td>12. *Knowledge that a variable can only hold one variable at a time</td>
</tr>
<tr>
<td>13. Determine what variables are required in a program to achieve the goals of the computational solution</td>
</tr>
<tr>
<td>14. Evaluate a Boolean expression</td>
</tr>
<tr>
<td>15. Use Boolean operators (And, Or, Not) in a program</td>
</tr>
<tr>
<td>16. Create a Boolean expression for a given condition</td>
</tr>
<tr>
<td>17. Identify sub-parts of a computational solution</td>
</tr>
<tr>
<td>18. Create a Boolean expression to control a loop</td>
</tr>
<tr>
<td>19. Identify a constant vs. a variable in a program</td>
</tr>
<tr>
<td>20. Decide on an appropriate initial value for a variable</td>
</tr>
<tr>
<td>21. Create an expression using existing variables</td>
</tr>
<tr>
<td>22. Define an abstraction to represent a general solution</td>
</tr>
<tr>
<td>23. Identify the structural (sub-)parts of a program</td>
</tr>
</tbody>
</table>
~2 years of multiple rounds of design, piloting, scoring, and revision
- Expert Review of FKSAs & initial item set (with several extra items)
- Piloted in middle school CS classrooms with teacher feedback (2 rounds with 200 students)
- 1:1 think-alouds with students
- Review by an ELL expert

**Paper & pencil assessment** (now also available as a **computer-based version on an NSF-supported assessment platform**)
- To be completed in a **45-50 minute period**
- Include **scoring rubrics** and additional materials to support interpretation.
Teacher Feedback

In general, how well do you think your students will do on this question? Put an X in the box that best describes your answer:

<table>
<thead>
<tr>
<th></th>
<th>A majority of my students will get this item incorrect</th>
<th>About half of my students will get this correct</th>
<th>A majority of my students will get this item correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Below is a list of the learning goals addressed in this question. Please indicate how well these learning goals were covered for this class.

<table>
<thead>
<tr>
<th>Ability to evaluate a Boolean expression</th>
<th>Not covered (i.e., spent no instructional time on this goal)</th>
<th>Somewhat covered (i.e., spent some instructional time on this goal)</th>
<th>Fully covered (i.e., spent a lot of instructional time on this goal)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>FKSA Coverage</td>
<td>Sub-parts</td>
<td>Concepts</td>
</tr>
<tr>
<td>------</td>
<td>---------------</td>
<td>-----------</td>
<td>----------</td>
</tr>
<tr>
<td>1</td>
<td>1, 8, 13</td>
<td>a</td>
<td>Boolean Expression + Conditional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>Boolean Expression + Conditional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c</td>
<td>Boolean Expression + Conditional</td>
</tr>
<tr>
<td>2</td>
<td>1, 8, 12</td>
<td>a</td>
<td>Relational Expression + Conditional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>Variables + Conditional</td>
</tr>
<tr>
<td>3</td>
<td>2, 3, 9, 10</td>
<td>a</td>
<td>Variables + Boolean Expression + Loops</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>Variables + Boolean Expression + Loops</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c</td>
<td>Variables + Loops</td>
</tr>
<tr>
<td>4</td>
<td>2, 3, 4, 8, 10, 11, 14, 15, 16, 17, 18, 21, 23</td>
<td>a</td>
<td>Abstraction (Pattern Generalization)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>Abstraction (Pattern Generalization)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c</td>
<td>Abstraction (Pattern Generalization)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d</td>
<td>Variables + Arithmetic Expression</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e</td>
<td>Variables + Boolean Expression + Loops</td>
</tr>
<tr>
<td></td>
<td></td>
<td>f</td>
<td>Variables + Arithmetic Expression + Loops</td>
</tr>
<tr>
<td></td>
<td></td>
<td>g</td>
<td>Variable + Boolean Expression</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td></td>
<td>Boolean Expression</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td></td>
<td>Boolean Expression</td>
</tr>
<tr>
<td>7</td>
<td>4, 8, 9, 13, 15, 16, 18, 19, 20, 21</td>
<td>a</td>
<td>Variables</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>Variables</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c</td>
<td>Variable + Arithmetic Expression</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d</td>
<td>Variable + Arithmetic Expression</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e</td>
<td>Variable + Boolean Expression</td>
</tr>
<tr>
<td></td>
<td></td>
<td>f</td>
<td>Variables + Arithmetic Expression + Loops</td>
</tr>
<tr>
<td>8</td>
<td>4, 5, 6, 7, 11, 22</td>
<td>a</td>
<td>Pattern Recognition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>Abstraction (Pattern Generalization)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c</td>
<td>Loops</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d</td>
<td>Loops</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e</td>
<td>Variables + Loops</td>
</tr>
<tr>
<td>9</td>
<td>2, 3, 4, 5</td>
<td></td>
<td>Loops</td>
</tr>
</tbody>
</table>

**Ability to...**

1. Describe the execution that is performed by a (specific) given sequence of instructions (not incl. loops)
2. Describe what a given loop is doing
3. Describe the sequence that is executed in a given program when the program contains things inside and outside of the loop.
4. *Knowledge that a loop involves a repeating pattern, that will terminate under a specified condition or after a certain number of repetitions*
5. Identify the repeating pattern within a loop
6. Describe the structural components of a pattern
7. Identify a pattern from a real-world phenomenon
8. Describe how a conditional pathway would operate or create a conditional pathway
9. Create variables, assign values and update variables
10. Describe how a variable changes values in a loop
11. Use a variable within a loop
12. *Knowledge that a variable can only hold one variable at a time*
13. Determine what variables are required in a program to achieve the goals of the computational solution.
14. Evaluate a Boolean expression
15. Use Boolean operators (And, Or, Not) in a program
16. Create a Boolean expression for a given condition
17. Identify sub-parts of a computational solution
18. Create a Boolean expression to control a loop
19. Identify a constant vs. a variable in a program
20. Decide on an appropriate initial value for a variable
21. Create an expression using existing variables
22. Define an abstraction to represent a general solution
23. Identify the structural (sub-)parts of a program
<table>
<thead>
<tr>
<th>Item</th>
<th>FKSA Coverage</th>
<th>Sub-parts</th>
<th>Concepts</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, 8, 13</td>
<td>a</td>
<td>Boolean Expression + Conditional</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>Boolean Expression + Conditional</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c</td>
<td>Boolean Expression + Conditional</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1, 8, 12</td>
<td>a</td>
<td>Relational Expression + Conditional</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>Variables + Conditional</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2, 3, 9, 10</td>
<td>a</td>
<td>Variables + Boolean Expression + Loops</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>Variables + Boolean Expression + Loops</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c</td>
<td>Variables + Loops</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2, 3, 4, 8, 10, 11, 14, 15, 16, 17, 18, 21, 23</td>
<td>a</td>
<td>Abstraction (Pattern Generalization)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>Abstraction (Pattern Generalization)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c</td>
<td>Abstraction (Pattern Generalization)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d</td>
<td>Variables + Arithmetic Expression</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e</td>
<td>Variables + Boolean Expression + Loops</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>f</td>
<td>Variables + Arithmetic Expression + Loops</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>g</td>
<td>Variable + Boolean Expression</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td></td>
<td>Boolean Expression</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td></td>
<td>Boolean Expression</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>4, 8, 9, 13, 15, 16, 18, 19, 20, 21</td>
<td>a</td>
<td>Variables</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>Variables</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c</td>
<td>Variable + Arithmetic Expression</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d</td>
<td>Variable + Arithmetic Expression</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e</td>
<td>Variable + Boolean Expression</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>f</td>
<td>Variables + Arithmetic Expression + Loops</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>4, 5, 6, 7, 11, 22</td>
<td>a</td>
<td>Pattern Recognition</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>Abstraction (Pattern Generalization)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c</td>
<td>Loops</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d</td>
<td>Loops</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e</td>
<td>Variables + Loops</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>2, 3, 4, 5</td>
<td></td>
<td>Loops</td>
<td>4</td>
</tr>
</tbody>
</table>

**Ability to...**

1. Describe the execution that is performed by a (specific) given sequence of instructions (not incl. loops)
2. Describe what a given loop is doing
3. Describe the sequence that is executed in a given program when the program contains things inside and outside of the loop.
4. *Knowledge that a loop involves a repeating pattern, that will terminate under a specified condition or after a certain number of repetitions
5. Identify the repeating pattern within a loop
6. Describe the structural components of a pattern
7. Identify a pattern from a real-world phenomenon
8. Describe how a conditional pathway would operate or create a conditional pathway
9. Create variables, assign values and update variables
10. Describe how a variable changes values in a loop
11. Use a variable within a loop
12. *Knowledge that a variable can only hold one variable at a time
13. Determine what variables are required in a program to achieve the goals of the computational solution
14. Evaluate a Boolean expression
15. Use Boolean operators (And, Or, Not) in a program
16. Create a Boolean expression for a given condition
17. Identify sub-parts of a computational solution
18. Create a Boolean expression to control a loop
19. Identify a constant vs. a variable in a program
20. Decide on an appropriate initial value for a variable
21. Create an expression using existing variables
22. Define an abstraction to represent a general solution
23. Identify the structural (sub-)parts of a program
7. You are programming a maze game.
   - The screen shows a cat, maze walls, and gold coins.
   - You make a cat move around the screen trying to find gold coins
   - The maze walls and coins stay in the same location every time you play the game.

   - You start with a score of 0.
   - You have 5 lives at the start.
   - If the cat touches a gold coin-
     - You gain a life, and
     - your score increases by 100.
   - If the cat touches a wall-
     - you lose a life, and
     - your score decreases by 100.

   The game ends when you reach a score of 500 or have no lives left.
   If your score is 500 when the game ends, you win. If your score is less than 500
   when the game ends, you lose.

   a. For each of the following, does it change as you play the game?

<table>
<thead>
<tr>
<th></th>
<th>Changes</th>
<th>Does not change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lives</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wall</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coins</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Cat's Position</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   b. Describe 1 variable that is needed to program this game.
      - What would you name the variable?
      - Describe the purpose of the variable.
      - What is the starting value of the variable?

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Describe how this variable is used in the program (or what is the purpose of this variable)</th>
<th>Starting value of the variable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   c. What happens when the cat touches a gold coin?
      - Use the IF block below
      - Write the expressions one below the other to show how the value of the variables
        'Score' & 'Lives' in the game changes.
        ('Score' keeps track of the score and 'Lives' keeps track of number of lives left)

   d. What happens when the cat runs into a Wall?
      - Use the IF block below
      - Write the expressions one below the other to show how the value of the variables
        'Score' & 'Lives' in the game changes.
        ('Score' keeps track of the score and 'Lives' keeps track of number of lives left)

   e. When does the game end?
      - Write an expression to show when the game ends.
      - The expression you write is meant to be used in a Repeat Until block.

   f. Write an ‘if’ statement to decide if the player has won or lost. (Use variables)
4. Kayla is saving money to buy a new bicycle. Kayla has six uncles. On Kayla’s birthday, each uncle gave her money to help her buy the bicycle. To figure out if she has enough money to buy the bicycle, Kayla creates the following program:

The program has been divided into three sections (A, B and C). In the table below match the section of the program (A, B or C) to the description of what it does.

<table>
<thead>
<tr>
<th>What the program does:</th>
<th>Which section of the program?</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Sums up the amount of money that Kayla receives from her uncles</td>
<td>□ A □ B □ C</td>
</tr>
<tr>
<td>b. Compares that amount to the price of the bicycle and tells Kayla whether or not she has enough money to buy the bicycle.</td>
<td>□ A □ B □ C</td>
</tr>
<tr>
<td>c. Tells Kayla to enter the price of the bicycle</td>
<td>□ A □ B □ C</td>
</tr>
</tbody>
</table>

d. Choose the instruction below that belongs in the space marked 1 in the program.

- set TotalMoneyReceived to answer
- set TotalMoneyReceived to 0
- change TotalMoneyReceived by answer
- change TotalMoneyReceived by 0

f. Choose the instruction below that belongs in the space marked 3 in the program.

- BicyclePrice = TotalMoneyReceived and BicyclePrice > TotalMoneyReceived
- BicyclePrice = TotalMoneyReceived and BicyclePrice < TotalMoneyReceived
- BicyclePrice < TotalMoneyReceived or BicyclePrice > TotalMoneyReceived
- BicyclePrice < TotalMoneyReceived and BicyclePrice > TotalMoneyReceived
Kayla’s friend, Simone, wants to buy the same bicycle. Simone does not have anyone (like Kayla had her uncle) who can gift her the money.

- Simone decides to create a page on the internet to receive donations.
- When Simone receives enough money to buy the bicycle, she wants to stop accepting more donations.

- Create an expression for a Repeat-Until loop to make the program run until Simone has enough money to buy the bicycle. Use variables TotalMoneyReceived & BicyclePrice.

```
repeat until
```

Scoring for Q4 (max=9 points):
4a: 1 point if correct; 0 otherwise
4b: 1 point if correct; 0 otherwise
4c: 1 point if correct; 0 otherwise
4d: 1 point if correct; 0 otherwise
4e: 1 point if correct; 0 otherwise
4f: 2 points if correct; 0 otherwise
4g: 2 points if correct (A, B or C); 0 otherwise

A: (BicyclePrice = TotalMoneyReceived OR BicyclePrice < TotalMoneyReceived)
BicyclePrice <= TotalMoneyReceived
TotalMoneyReceived >= BicyclePrice

B: BicyclePrice = TotalMoneyReceived

C: BicyclePrice < TotalMoneyReceived ; TotalMoneyReceived > BicyclePrice
8. A pattern is shown below.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>∗</td>
<td>∗</td>
<td>∗</td>
<td>△</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Step 1**  **Step 2**  **Step 3**

a. Draw the next step (Step 4) in the pattern.

b. Describe in words what the pattern would look like for Step 600.

c. If you were to program Step 600 for this pattern, pick ONE of the following Scratch blocks that you would need:

- [ ] `if [ ] then`
- [ ] `else`
- [ ] `repeat [ ]`
- [ ] `wait [ ] secs`
- [ ] `say [ ]`
- [ ] `forever`

---

d. The following program draws Step 2 of the pattern:

- Draw a star
- Move to the right
- Draw a star
- Move to the right
- Draw a triangle
- Move to the right
- Draw a triangle

Now write a program to draw Step 600 of the pattern. Use the Scratch block you picked in part c. You may use the Move and Draw commands as shown above.

---

e. Now, write a "pattern generator" program to draw the pattern for ANY Step number that a user inputs. You may use the Move and Draw commands as shown above, along with Scratch commands.

The first part of the program looks like this—

When [ ] clicked

- Set [StepNumber] to [answer] (StepNumber is a variable whose value is set to whatever a user inputs (in the Ask command). Use StepNumber in your program instead of a value of a particular step.)

Continue the program below—

- Ask "What step of the pattern do you want to draw?" and wait
- Set [StepNumber] to [answer]
Data and Analyses from Classroom Research

<table>
<thead>
<tr>
<th>SCHOOL CHARACTERISTICS</th>
<th>Grade 6</th>
<th>Grade 7</th>
<th>Grade 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low SES</td>
<td>85%</td>
<td>40%</td>
<td>87%</td>
</tr>
<tr>
<td>Math Proficiency</td>
<td>55%</td>
<td>79%</td>
<td>34%</td>
</tr>
<tr>
<td>English Proficiency</td>
<td>54%</td>
<td>80%</td>
<td>41%</td>
</tr>
<tr>
<td>Science Proficiency</td>
<td>62%</td>
<td>84%</td>
<td>34%</td>
</tr>
<tr>
<td>Learning Disabilities</td>
<td>12.8%</td>
<td>3.5%</td>
<td>15%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Pre-Test Mean</th>
<th>Post-Test Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grade 6 (n=27)</td>
<td>39.2</td>
<td>52.5</td>
</tr>
<tr>
<td>Grade 7 (n=31)</td>
<td>65.5</td>
<td>80.0</td>
</tr>
<tr>
<td>Grade 8 (n=14)</td>
<td>56.3</td>
<td>68.5</td>
</tr>
</tbody>
</table>

- Average total score ~70% across all grades
- Female and male students had comparable average scores on the assessment
- Score distributions were skewed toward the high end of the score distribution (for 7th & 8th graders)
- Significant pre-to-post gains → sensitivity to a curricular intervention // test-retest reliability
- Positive (& significant) correlations with English & Math scores on CA state standardized tests (SBAC)
Analysis and Results

- **High internal consistency among the items**

- **The assessment items reflected a fairly high level of reliability at 82%**

- Item #2 most difficult (ambiguous wording)

- Item #6 – very easy and low DI

- Task difficulty levels range from .55 to .87

- Discrimination analysis revealed that items targeting combinations of programming concepts rather than individual concepts (#4, #7, #8, #9) had higher discriminatory power
9. What does the fox say when the green flag is clicked? Write down what the fox says, in order.

The fox says:
2a. Jenny wrote a program that asks the user to enter their age, and then tells the user their school type.
   - "Pre-school" for children age 3 to 5.
   - "Elementary School" for children age 6 to 12.
   - "Middle or High School" for children age 13 to 16.

Jenny created the program below.

2b. Now Jenny wants to add to her program. She will ask the user’s name and age. She wants to use the name to first say "Hi" to the person. Then she will use their age to tell their school type.

Her program shown above does not do the "Hi" part correctly. How would you fix it?

- Yes
- No
2a. For her 13th birthday, Jenny is organizing a party with the "Teen Party Rule"

"Teen Party Rule" is that only her teenage friends (friends who are 13 to 18 years old, that is, 13, 14, 15, 16, 17, or 18 years old) can attend the party.

She writes the program below that checks if a person can attend the party by asking their age.

![Program diagram]

(1) In the program shown above, for which values of 'Age' will the program say “Yes, you can attend the party”?

- For any value of 'Age' that is greater than or equal to 18
- For any value of 'Age' that is less than or equal to 18
- For any value of 'Age' that is less than 13
- For any value of 'Age' that is greater than 18

(2) Has this program been coded correctly based on Jenny's "Teen Party rule"?

- Yes
- No

If your answer is 'No', what is the problem with the code?

2b. Now Jenny creates another program for her Teen party that should do the following—

- Ask guests for their name and greet the guest by name (for example, the program says "Hi Amy", if the guest entered the name "Amy")
- Ask if they would like Pizza (P) or Burger (B) so that she can keep count for the food order

![Program diagram]

(a) What will the program say when Jenny tests her program and enters

Jenny when asked "What's your name?", and 'P' when asked "Would you like to eat Pizza (enter 'P') or Burger ('enter 'B')?"

(b) Is there any problem with the code based on what Jenny wanted the program to do?

- Yes
- No

If your answer above is Yes, what is the problem with the code?
Piloted with N=108 students (6th: n=48; 7th: n=40; 8th: n=20)

Some minor edits made based on piloting the revision

Many 6th graders do not understand the relational operator (\(<, >\))

The majority of students did not know about the “join” operator and that `[join “hi” <answer>]` will concatenate “hi ” and the value in the variable `<answer>`.

Many students’ explanation for part (b) suggested that the problem with the code was that does not have it saying “Hi Jenny” anywhere ➔ They did not understand that “answer” is a variable that contains a string value (“Jenny”)

CSTA standards for 6-8 mentions understanding variables and variable types.

Example: Alternatively, students could write a program that prompts the user for their name, stores the user’s response in a variable (e.g. `userName`), and uses this variable to greet the user by name.
Use of the MSCS Assessment

- Nascent stages of CS in K-12
- Several factors contribute to student performance in an assessment, including (among others) the CS curriculum, students’ background (and prior experience with programming), teacher CS experience and expertise, etc.
- Affective factors at play such as identity, interest, motivation, and engagement

Crucial that such a test not be the only measure for “grading” students at the end of the course

- Should be **ONE of MANY** varied forms of assessment (such as final projects, among other tasks) that can provide a holistic measure of student learning and engagement
Interested in using the MSCS assessment?

• For a pencil-paper assessment, please write to shuchi.grover@gmail.com

• For an online version of the assessment, create a teacher account on edfinity.com (and get in touch with me for further steps)

• A detailed paper describing the assessment along with rubrics will be shared online.

• To support the ongoing research and collect data on the use of the assessment, please consider consenting to share your anonymized data (with only relevant information such as student grade) for informing future use and perhaps revision of the assessment.
The program below sets an Alarm. Answer the following questions based on the given program.

1. For her 13th birthday, Jenny is organizing a party with the "Teen Party Rule".

2. Below is a program written in Scratch:

3. Kayla is saving money to buy a new bicycle. Kayla has six uncles. On Kayla's birthday, each uncle gave her money to help her buy the bicycle. To figure out if she has enough money to buy a bicycle, Kayla creates the following program:

4. Kathy takes an umbrella to school if it is raining. She also takes an umbrella if it is cloudy.

5. When the car is running it keeps a warning if your seatbelt is not fastened.

6. You are programming a maze game.

7. A pattern is shown below.
Thoughts? / Questions?

• How do you think your students would fare? (6th / 7th / 8th grade?)

• Are there topics you don’t cover? (or don’t cover well?)

• Are items (or sub-items) that you think some of your students might struggle with? Why?

• Any other thoughts?
Pilot study 1
- Two middle school CS teachers helped pilot the assessments
- ~100 6th, 7th, and 8th grade students participated
- Students had completed an introductory CS course using Scratch
- Teacher feedback

Cognitive think-alouds
- Students from a summer camp targeted at underrepresented minority students; they mirrored the demographics of the pilot sample

Pilot study 2
- Four middle school CS teachers helped pilot the assessments
- ~100 6th, 7th, and 8th grade students participated
- Students had completed an introductory CS course using Scratch
- Teacher feedback

Pre – post assessment
- Three middle school CS teachers in an urban diverse school district
- 74 students in 6th, 7th & 8th
- Curricular intervention on introductory CS (programming)

Scoring & Analyses
- Data from 71 students (who completed both pre & posttest) used for validity, reliability and difficulty/discrimination analyses
(a) Write down, in order, what will appear on the screen in the fox’s speech box, after the green flag is clicked.

(b) Does the value of \textbf{Number} change in the loop?  
☐ Yes  ☐ No  
If Yes, explain how it changes.

(c) Does the value of \textbf{NumberOfTimes} change in the loop?  
☐ Yes  ☐ No  
If Yes, explain how it changes.
Validity

“the degree to which evidence and theory support the interpretations of test scores entailed by proposed uses of tests” (AERA, APA, NCME, 1999, p. 9)

- Focuses on supporting assessment inferences through —
  - Test Content
  - Student Response Processes
  - Internal Structure
  - Relations to other Variables
  - Testing Consequences

- Collected validity evidence based on test content and student responses processes to help us refine and improve the assessments
  - Test content (Expert Review for alignment between the FKSAs & learning goals)
  - Student response processes (Cognitive think-aloud interviews with students)
Analysis and Results: Validity Evidence Based on Internal Structure

- Principal Factors Analysis using the pre-post scores data at the item level indicated that the assessment was measuring one overall factor → high internal consistency among the items

- The assessment items reflected a fairly high level of reliability at 82%

- Item #2 most difficult (badly worded)
- Item #6 – very easy and low DI
- Task difficulty levels range from .55 to .87

- Discrimination analysis revealed that items targeting combinations of programming concepts rather than individual concepts (#4, #7, #8, #9) had higher discriminatory power
Upcoming Activities

• We are currently finalizing the details for an entire year’s worth of activities in the community.

• Follow us on Twitter (@csforalltchrs) for the latest updates.
Are You a Member of CS for All Teachers?

Membership is FREE! Sign up in order to:

• Connect with other CS teachers
• Participate in free webinars
• Access free resources
• Learn more about ECS and CSP curricula
• Download the latest ECS assessments

https://csforallteachers.org/user/register
CS for All Teachers

Email: csforallteachers@air.org
Website: https://csforallteachers.org/
Twitter: @CSforallTchrs