Assessment Design Patterns for Computational Thinking Practices in *Exploring Computer Science*

August 2017

SRI Education™

A DIVISION OF SRI INTERNATIONAL

Authors

Eric Snow, Carol Tate, Daisy Rutstein, Marie Bienkowski, **SRI International**

Acknowledgments

This work originated in a December 2011 workshop supported by a National Science Foundation (NSF) planning grant. Attending the workshop was an interdisciplinary group of experts who gave us guidance on modeling the computational thinking domain for the purposes of designing assessments. Meeting participants included experts in assessment and evidence-centered design: Irvin Katz, Ph.D., Educational Testing Service; Drs. Britte Cheng and the late Geneva Haertel, SRI International; and Gail Chapman, University of California at Los Angeles/Into the Loop. Others were experts in computational thinking, computer science, and innovative ways of delivering content in both school and afterschool settings: Drs. Maggie Johnson, Google; Alexander Repenning, University of Colorado, Boulder; Chris Stephenson, Google (formerly at the Computer Science Teachers Association [CSTA)]); and Melissa Koch, M.A., Anita Borg Institute (formerly at SRI). Representing the Exploring Computer Science curriculum research and development project were Joanna Goode, Ph.D., University of Oregon; Jane Margolis, Ed.D., and Noelle Griffin, Ph.D., UCLA; Jenna Masser, CSTA; and Suyen Moncada-Machado, Los Angeles Unified School District. Experts in program evaluation, especially of information technology- and computer science-oriented curriculum, provided guidance on the use of assessments in research and evaluation: Girlie Delacruz, Ph.D., UCLA/CRESST; Jill Feldman, Ph.D., Westat; and Shaileen Pokress, Ed. M., Project Lead the Way.

After the planning grant and under a new NSF grant in July 2013, early versions of the Exploring Computer Science assessment design patterns were reviewed by an expert panel consisting of Dr. Katz and Drs. Lucia Dettori, Chris Schunn, David Reed, and Cindy Hmelo-Silver. The patterns were also reviewed during a meeting in November 2014 by that grant's advisory board members, Drs. Dan Garcia, UC Berkeley; Jill Denner, ETR Associates; Drew Gitomer, Rutgers University; and Terry Vendlinski, TVATE (formerly at SRI).

For information on our more general computational thinking design patterns, a companion report is available from SRI: *Design Patterns for Computational Thinking in Secondary Computer Science* available from https://pact.sri.com/resources.html

This material is based on work supported by the National Science Foundation under Grant Nos. CNS-1132232, CNS-1240625, and DRL-1418149. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or any collaborator or partner named herein.

Suggested Citation

Snow, E., Tate, C., Rutstein, D., & Bienkowski, M. (2017). *Assessment Design Patterns for Computational Thinking Practices in Exploring Computer Science*. Menlo Park, CA: SRI International.

Contents

Introduction

Exploring Computer Science (ECS) is an introductory course designed to provide high school students an accessible entry point to computing and a pathway to college-level study in computer science (http://www.exploringcs.org/). ECS was created in response to research showing extremely low participation in computing by girls and students of color. ECS attempts to address this inequity with a three-pronged approach: a yearlong curriculum, ongoing professional development for teachers, and policy initiatives to support adoption of introductory computer science courses. Introduced in Los Angeles in 2008, ECS has expanded rapidly across the nation with the support of the National Science Foundation (NSF).

ECS differs from earlier introductory computer science courses in that its goal is not simply to teach students to code in a particular programming language. Rather, ECS is designed to teach foundational computer science concepts and to engage students in computational thinking practices. ECS presents computer science as a creative endeavor, emphasizing that technology is a tool for solving problems and that computing has a social impact. Through the active, inquiry-based lessons and activities, students develop computational thinking skills by exploring topics that are relevant to their lives.

To teach ECS effectively, educators must understand what students know about computing and how learning progresses. Assessing learners' knowledge of new definitions and programming language commands is relatively straightforward, but discerning how they use computation to frame and solve problems and design creative computational artifacts is far more challenging. Assessing how learners apply their computational knowledge means searching for evidence of deeper understanding of the connection between problems to solve and the comprehension and production of coded solutions. In short, it means assessing computational processes as well as computational products. Assessments tuned to the ECS curriculum must measure this deeper learning and must also reflect the dedication to equity and inquiry-based instruction that are foundational to the program.

How, then, can we best support the design and development of assessments that measure how students think about problem solving with computation, apply abstraction, understand computational work, and design and implement creative solutions to problems?

This report provides an overview of a principled approach to designing assessment tasks that can generate valid evidence of students' abilities to think computationally. *Computational thinking* refers to the core disciplinary ideas and ways of knowing that constitute computer science. *Principled assessment* means designing assessment tasks to measure important knowledge and practices by specifying chains of evidence that can be traced from what students do (observable behaviors) to claims about what they know. This approach to assessment produces documents called *design patterns*, which serve as a template for designing tasks to elicit evidence of a students' ability in constructs of interest. Design patterns are meant to be generative of multiple tasks and to guide the design and development of assessments to measure both knowledge and skills in the context of specific learning experiences.

The design patterns presented in this report were developed under the project *Principled Assessment of Computational Thinking* (PACT). The long-term objectives in the ongoing PACT suite of projects (NSF awards CNS-1132232, CNS-1240625, CNS-1433065, CNS- 1640237, and DRL-1418149) are as follows:

- Analyze and model the computational thinking domain to elicit its underlying knowledge and skills and develop artifacts— standards mappings, design patterns, and tasks—to shape the assessment design and development process.
- Develop and validate measures of computational thinking by instantiating design patterns in the context of specific curricula to guide assessment design up to and including implementation, with an emphasis on delivering assessments online.
- Use assessments and other measures to identify the implementation factors for ECS and other computer science curricula that influence secondary students' learning of computational thinking.
- Investigate the design and delivery of high-quality assessment literacy materials and sustainable, ongoing training as part of the ECS teacher professional development workshops.

To reach these objectives, SRI International pursued the following specific short-term goals:

- Create design patterns for the major computational thinking practices covered in ECS that can be used to design and refine assessments as the curriculum evolves.
- Develop templates for tasks to assess computational thinking practices in the context of ECS.
- Create, pilot-test, and validate four unit assessments (ECS Units 1–4) and a cumulative assessment (across Units 1–4) and deliver these assessments online.

This report addresses the first short-term goal by presenting a set of design patterns that are intended to support the design and development of assessment tasks for the ECS curriculum. This set of design patterns models the foundational knowledge and skills that underlie the first four units of ECS:

1. Human Computer Interaction
2. Problem Solving
3. Web Design
4. Introduction to Programming.

We omitted ECS Units 5 and 6 because they reinforce and apply concepts from prior units and are implemented by secondary teachers in widely varying ways.

The next section provides an orientation to computational thinking in the ECS curriculum. Then we explain the evidence-centered design approach, detailing the development of the design pattern elements with an illustrative example of an assessment task. The appendix presents the design patterns for the first four ECS units.

Computational Thinking in Exploring Computer Science

As computing has evolved from a tool for solving equations and analyzing data to a way of discovering new knowledge, it has transformed how science is practiced in fields from biology and physics to economics and business management (Denning, 2007). Computational thinking is increasingly being recognized as an essential form of literacy for informed citizens in the modern world, as well as a way to investigate natural processes (Szalay & Gray, 2006). Although definitions of computational thinking vary (Grover & Pea, 2013), they generally include these practices:

> formulating problems in a way that enables us to use a computer and other tools to help solve them; logically organizing and analyzing data; representing data through abstractions such as models and simulations; automating solutions through algorithmic thinking (a series of ordered steps); identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources; and generalizing and transferring this problem solving process to a wide variety of problems. (Barr, Harrison, & Conery, 2011, p. 21)

Exploring Computer Science was created under NSF funding as a pre-Advanced Placement (AP) course that prepares students for future engagement in computer science education and careers. The course was founded on the idea that the route to computer science study should not begin with an advanced, exam-driven course (Margolis, Goode, & Chapman, 2015). ECS is intended to provide a path to college-level computer science by scaffolding students' developing computational thinking skills with activities that engage their prior knowledge and interests. By emphasizing the social and ethical dimensions of computing and by using an inquiry-based approach to learning computational thinking practices, ECS demonstrates the connections between computing and students' everyday lives. The ECS view on programming instruction is in line with that of other K–12 stakeholders whose definitions of computational thinking downplay an exclusive focus on programming skills and instead emphasize the problem solving and data representation aspects of computing.

ECS lessons focus on the computational thinking practices drawn from a proposed AP course in computer science (Arpaci-Dusseau et al., 2013) that capture the core ideas of computing:

- Apply abstractions and models.
- Analyze one's own computational work and the work of others.
- Design and implement creative solutions and artifacts.
- Analyze the effects of developments in computing.
- Communicate computational thought processes, procedures, and results to others.
- Collaborate with peers on computing activities.

Definitions of computational thinking and course objectives from ECS are important starting points for developing assessment materials targeting the construct of computational thinking, but they are insufficient for developing assessment tasks. Translating these objectives into specific learning targets that are amenable to valid, reliable measurement requires a complementary approach, as outlined in the next section.

Evidence-Centered Design

SRI specializes in using evidence-centered design (ECD) to develop assessments for hard-to-assess constructs. ECD is especially helpful when the knowledge and skills to be measured involve complex, multistep performances, such as those required in computational thinking. ECD helps to translate broad learning goals into statements of student ability and to describe the kinds of tasks that would elicit evidence of that ability. ECD also provides guidance on how the evidence can be aggregated to produce a model of what the student knows and can do. Our experience in difficult-to-assess domains[1] has been that the considerable up-front design work that ECD requires lays the groundwork for efficiently generating families of assessment items.

ECD makes explicit and provides the tools for the building of assessment arguments (Mislevy & Riconscente, 2006; Mislevy, Steinberg, & Almond, 2003). Messick (1994) summarized the essence of the assessment argument as follows:

> A construct-centered approach would begin by asking what complex of knowledge, skills, or other attributes should be assessed, presumably because they are tied to explicit or implicit objectives of instruction or are otherwise valued by society. Next, what behaviors or performances should reveal those constructs, and what tasks or situations should elicit those behaviors? Thus, the nature of the construct guides the selection or construction of relevant tasks as well as the rational development of construct-based scoring criteria and rubrics. (p. 16)

ECD is typically described in terms of five layers of work (Mislevy & Haertel, 2006). These layers and examples of key entities/artifacts created along the way are shown in Exhibit 1. Although the layers suggest steps in a sequential design process, cycles of iteration and refinement are intended, both within and across layers, and work in different layers can occur simultaneously.

Exhibit 1. The Five Layers of Evidence-Centered Design and Key Entities for Computational Thinking

ECD Layer	Role	Key Entities & Examples
Domain analysis	Gather substantive information about the computational thinking domain of interest that has implications for assessment; how knowledge is constructed, acquired, used, and communicated	ECS concepts and terminology (e.g., abstraction, algorithms, debugging); tools (programming languages); representations (storyboards); and situations of use
Domain modeling	Express assessment argument in narrative form based on information from domain analysis	Specification of knowledge, skills, and other attributes to be assessed (e.g., describe result of running a program on given data); features of situations that can evoke evidence (find errors in programs); kinds of performances that convey evidence (use of sorting algorithms)
Conceptual assessment framework	Express assessment argument in structures and specifications for tasks and tests, evaluation procedures, measurement models	Student, evidence, and task models; student, observable, and task variables; rubrics; measurement models; test assembly specifications; task templates and task specifications
Assessment implementation	Implement assessment, including presentation-ready tasks and calibrated measurement models	Tasks, task materials (including supporting materials, tools, affordances); pilot test data to hone evaluation procedures and fit measurement models
Assessment delivery	Coordinate interactions of students and tasks: task- and test-level scoring; reporting	Tasks as presented; work products as created; scores as evaluated

Source: Adapted from Haertel et al. (2016).

1 For examples of domains where ECD has been applied, see DeBarger and Snow (2010) for life sciences; Mislevy, Riconscente, and Rutstein (2009) for model-based reasoning; and Cheng, Ructtinger, Fujii, and Mislevy (2010) for systems thinking.

This report presents work in the domain analysis and domain modeling layers of ECD and describes how these processes were applied to create assessments of computational thinking practices for ECS Units 1–4. Domain analysis helped us identify and define how the core computational thinking practices aligned with the learning objectives for the ECS units, and domain modeling resulted in design patterns that specified elements for ECS assessment design.

Domain Analysis

The main activity for domain analysis is reviewing available information on the topic of interest and how it is learned. For the work reported here, we were primarily interested in analyzing computational thinking practices as they were represented in the first four units of the ECS curriculum. Because we had already produced design patterns for computational thinking practices (CTP) as part of a previous PACT project, we had a considerable start on the domain modeling phase. The CTP design patterns were developed beginning in 2011 through consultation with various experts and working groups in computer science education and assessment (see Bienkowski, Snow, Rutstein, & Grover, 2015 for details). The domain analysis for the ECS design patterns included analysis of these CTP design patterns, as well as a thorough review of the learning objectives and lesson activities specified in the ECS curriculum. We also sought input from the curriculum design team and experienced ECS teachers. A panel of experts and advisors provided additional critical feedback (see Acknowledgments).

Domain Modeling

Domain modeling in ECD produces narrative descriptions of a domain for measurement, including specification of what is to be learned and what kinds of tasks and performances could be used to demonstrate that learning. Design patterns are a product of the domain modeling process. Design patterns consist of elements specifying all or part of a construct domain or subdomain in terms of the knowledge and skills to be measured, the observations or behaviors that can be used as evidence of knowledge and skills in that domain, and the tasks or activities that elicit the desired observations or behaviors (Mislevy & Riconscente, 2006). Exhibit 2 provides general descriptions of the elements that appear in all design patterns.

Exhibit 2. Design Pattern Elements

Element Title	Element Description
Focal knowledge, skills, and other attributes (FKSAs)	• The primary KSAs targeted by the design pattern and what we want to make inferences about. • For our initial work on computational thinking practices, we focused on skills rather than knowledge.
Additional KSAs	• Other KSAs that may be required for successful performance on the assessment tasks but are not the target skills that we are trying to assess. • For computer science, this may include knowledge of mathematics or programming languages and tools. • Additional KSAs may also be used to link across design patterns to show the interdependencies among skills.
Potential observations	• Features of the things students say, do, or make that constitute the evidence on which the inference about a student's performance will be based. • Potential observations are described using such qualities as accuracy, degree, completeness, and precision.
Potential work products	• Some possible artifacts or observations that one could see. • Work products are the artifacts scored during the assessment process.
Characteristic features	• Aspects of assessment situations that are likely to evoke the desired evidence or that are required to support the task.
Variable features	• Aspects of assessment situations that can be varied in order to shift difficulty or emphasis.

These elements specify the important ideas to be measured, and they can be used, reused, and refined to help generate many different forms of assessments. For example, one pattern could be used to generate a paper-and-pencil test, an online interactive test, or a rubric to score computational artifacts that students produce. Design patterns are sufficiently general to guide measurement of learning by traditional paper-pencil delivery, as well as by dynamic computer-based assessments.

The ECS curriculum provides overview statements of the content for each unit. The overviews for the ECS design patterns, which are adapted from those provided in v5 of the curriculum, are shown in Exhibit 3.

Additional design pattern elements specified during domain modeling describe what is within and outside the scope of the construct and give suggestions and guidance for assessment developers. The main elements of design patterns are described next.

Exhibit 3. Design Pattern Overviews, ECS Units 1–4

ECS Unit	Design Pattern Overview
1: Human Computer Interaction	This unit introduces students to the concepts of a computer and computing while investigating the major components of computers and the suitability of these components for particular applications. Students will experiment with Internet search techniques, explore a variety of websites and web applications and discuss issues of privacy and security. Fundamental notions of Human Computer Interaction (HCI) and ergonomics are introduced. Students will learn that “intelligent” machine behavior is not “magic” but is based on algorithms applied to useful representations of information, including large data sets. Students will learn the characteristics that make certain tasks easy or difficult for computers, and how these differ from those that humans characteristically find easy or difficult. Students will gain an appreciation for the many ways in which computing-enabled innovations have had an impact on society, as well as for the many different fields in which they are used. Connections among social, economic and cultural contexts will be discussed.
2: Problem Solving	This unit provides students with opportunities to become “computational thinkers” by applying a variety of problem-solving techniques as they create solutions to problems that are situated in a variety of computational contexts. The range of contexts motivates the need for students to think abstractly and apply known algorithms where appropriate, but also create new algorithms. Analysis of various solutions and algorithms will highlight problems that are not easily solved by a computer and for which there are no known solutions. This unit also focuses on the connections between mathematics and computer science. Students will be introduced to selected topics in discrete mathematics including Boolean logic, functions, graphs and the binary number system. Students are also introduced to searching and sorting algorithms and graphs.
3: Web Design	This unit prepares students to take the role of a developer by expanding their knowledge of algorithms, abstraction, and web page design and applying it to the creation of web pages and user documentation. Students will explore issues of social responsibility in web use. They will learn to plan and code their web pages using a variety of techniques and check their sites for usability. Students learn to create user-friendly websites. Students will apply fundamental notions of Human Computer Interaction (HCI) and ergonomics.
4: Introduction to Programming	This unit introduces students to some basic issues associated with program design and development. Students design algorithms and create programming solutions to a variety of computational problems using an iterative development process in a blocks-based language such as Scratch. Programming problems include mathematical and logical concepts and a variety of programming constructs.

Focal Knowledge, Skills, and Attributes

Focal here means central or core—the knowledge, skills, and other attributes related to the student that we want to assess. The FKSAs should cover the main ideas within the construct of interest. To illustrate the level of detail used in a FKSA, we show five ECS FKSAs in Exhibit 4.

Note that we deliberately express our FKSAs as "Ability to…" in order to capture the focus on practices, which we believe are best represented as the *application* of skills. This is in contrast to FKSAs that may be better captured as knowledge statements. Complete modeling of a domain requires both, however, and we are expanding our ECS design patterns to include knowledge-focused FKSAs to capture the computer science conceptual knowledge underlying the practices.

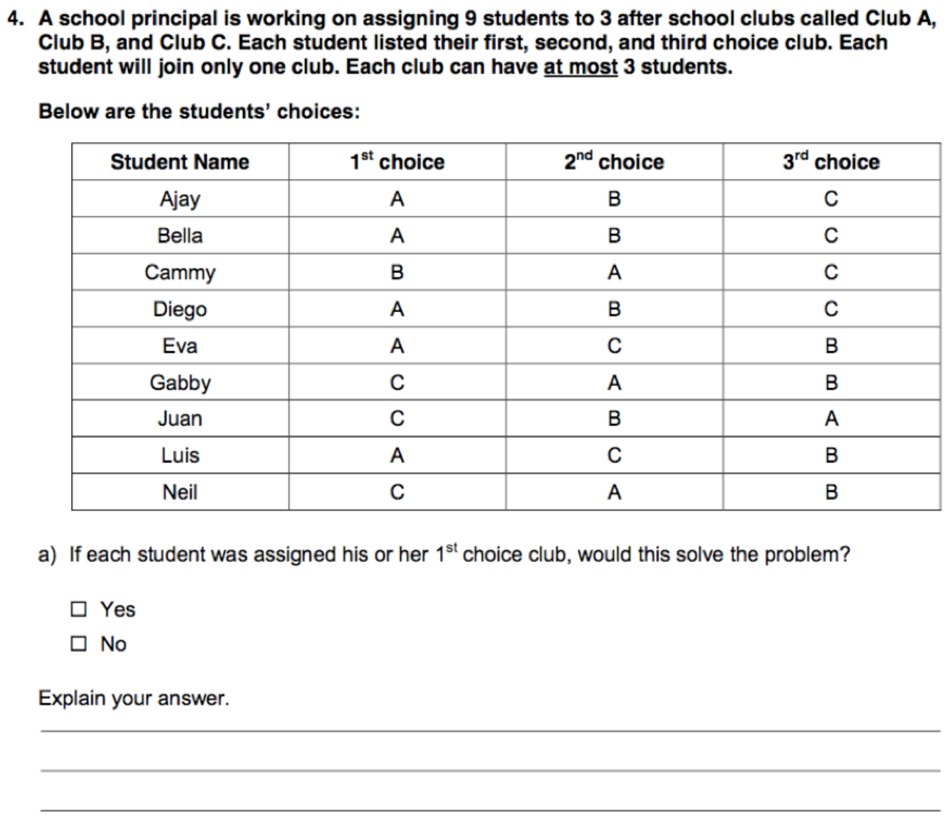
The FKSAs describe practices that students should be learning. We purposely restricted our vocabulary for these abilities—the verbs that we use to describe what students should be able to do—to a finite set that we can measure. For example, rather than stating our expectations in vague terms, such as "Students will *understand* algorithms," we have asked students to *state*, *explain*, and *compare*. We do not prompt students to *recognize*, but to *name, identify, represent,* and *describe*. Students *design* and *generate* work products and explain and justify their thinking.

Although we did not impose any hierarchy on these abilities (as we are not describing a progression of learning), we did use different verbs to differentiate the degree to which students should be able to apply their knowledge. For example, for some skills we thought that it was enough for students to be able to *describe* the phenomenon, whereas for others we wanted students to be able to *explain*. For our purposes, explanation includes drawing relationships, engaging in interpretation, and comparing and analyzing. In other areas, we indicate that students should engage with the concepts at an even higher level by stating the FKSAs as the ability to *evaluate*. Evaluation encompasses explanation, justification, and comparison. Exhibit 5 shows how some of the Unit 2 FKSAs are embedded in a realistic scenario about choosing afterschool clubs.

Exhibit 4. Example Focal Knowledge and Skills for ECS Unit 2: Problem Solving

1. Ability to state what an algorithm would output given a set of inputs
2. Ability to explain the inputs of an algorithm, how it operates on that input, and what the outputs are
3. Ability to evaluate the extent to which an algorithm solves a stated problem
4. Ability to compare the trade-offs between different algorithms for solving the same problem
5. Ability to create (using a narrative description or a representation) an algorithm that addresses a set of specifications

Exhibit 5. Example Focal KSAs of an Assessment Task in ECS Unit 2: Problem Solving



4. A school principal is working on assigning 9 students to 3 after school clubs called Club A, Club B, and Club C. Each student listed their first, second, and third choice club. Each student will join only one club. Each club can have <u>at most</u> 3 students.

Below are the students' choices:

Student Name	**1st choice**	**2nd choice**	**3rd choice**
Ajay	A	B	C
Bella	A	B	C
Cammy	B	A	C
Diego	A	B	C
Eva	A	C	B
Gabby	C	A	B
Juan	C	B	A
Luis	A	C	B
Neil	C	A	B

a) If each student was assigned his or her 1st choice club, would this solve the problem?

☐ Yes
☐ No

Explain your answer.

Example FKSAs

Ability to state what an algorithm would output given a set of inputs

Ability to evaluate the extent to which an algorithm solves a stated problem

Potential Observations and Work Products

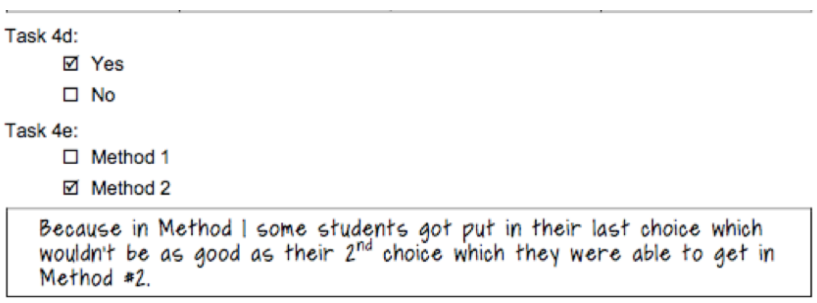
After delineating the FKSAs underlying a construct, we specify what we could observe the student doing or producing and how those behaviors or artifacts could provide evidence of the FKSAs.

Potential observations and associated work products are shown in Exhibit 6 for the same afterschool clubs task in Exhibit 5, displaying the student's checked boxes and open-text response. Note that by "potential," we intend to capture the idea that these elements can and should evolve as we learn more about teaching and learning in computer science.

Often the potential work products are what it is the student might actually produce, while the potential observations are the basis for the rubrics that would be developed to score these products.

Evaluating students' computational thinking is not simply a matter of looking for correct solutions to computational problems. There is often more than one correct solution and more than one way to approach a problem or task. We are more interested in students' thinking processes than their final product. We also look for appropriateness and the degree to which a work product exhibits a given characteristic. The potential observations list these characteristics of student work, and they are further elaborated in the scoring rubrics.

Exhibit 6. Example Potential Work Products, Potential Observations and Rubric Components for an Assessment Task in ECS Unit 2: Problem Solving



Task 4d:

[✓ Yes]

No

Task 4e:

Method 1

[✓ Method 2]

Because in Method 1 some students got put in their last choice which wouldn't be as good as their 2nd choice which they were able to get in Method #2.

Example Potential Work Products

The comparison of the trade-offs between different algorithms for solving a stated problem

Example Potential Observations

- Appropriateness of the evaluation
- Appropriateness of the comparison
- Appropriateness of the explanation

Example Rubric Components

1 point for selecting Method 2

1 point for providing an appropriate explanation that supports use of Method 2 over Method 1

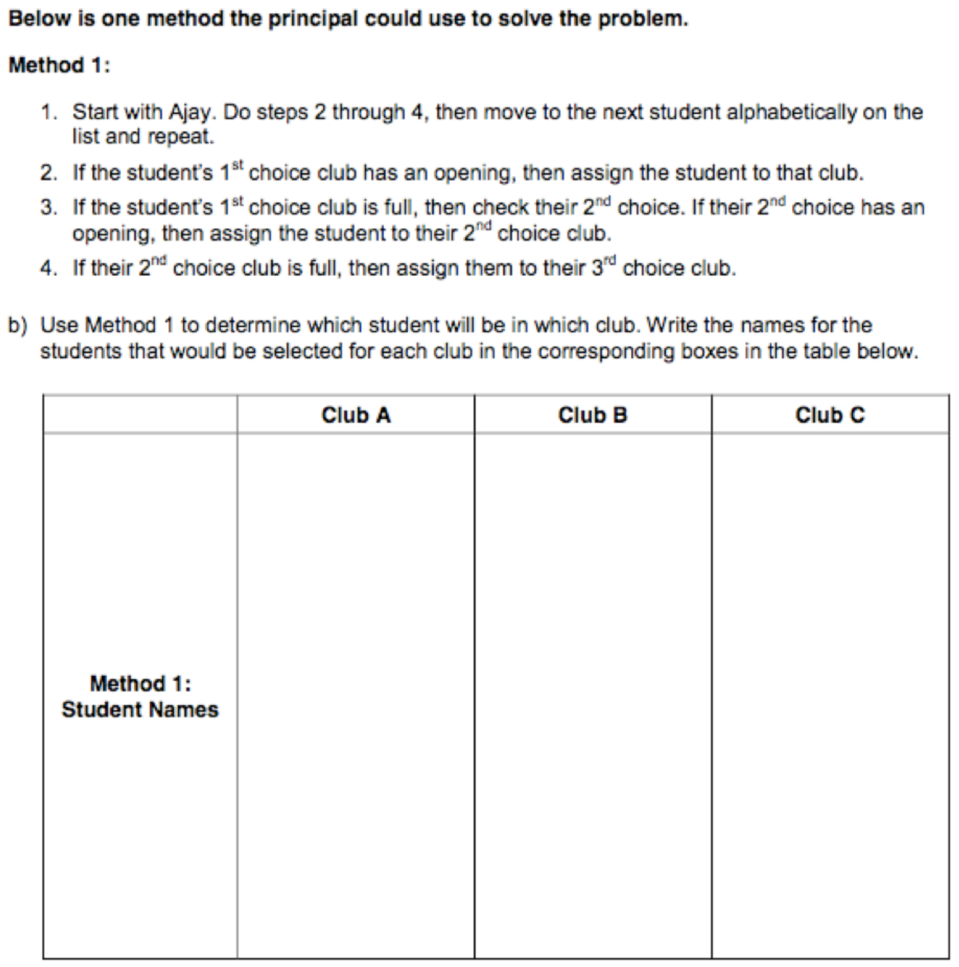
An appropriate explanation must include a benefit of Method 2 or a disadvantage of Method 1 (e.g., more people in Method 2 get their first choice or some students in Method 1 get their third choice while no students in Method 2 get their third choice).

Characteristic and Variable Features

In designing tasks, once we have thought about what it is we might want students to produce, we can work backward to the features of tasks that would be necessary to elicit the observations we are seeking. It is also important to think about how a task can be varied—to make it easier or harder or to remove potential barriers such as those due to language or culture. An ECD design pattern captures these as *characteristic* and *variable features*.

Characteristic features specify the *required* features of a task, features that must be present in order for the task to elicit evidence of the FKSAs. Variable features are features that may vary and may or may not be present in a particular task measuring the construct of interest. How features vary will depend on the measurement goals for the task and may involve changing the difficulty of an item or allowing for additional KSAs to be measured within the same task. Specifying the variable features ahead of time helps to highlight decisions that should be made when developing individual items. Exhibits 7 and 8 show how the characteristic and variable features for the FKSA "Ability to state what an algorithm would output given a set of inputs" appear in the afterschool clubs task.

Exhibit 7. Example Characteristic Feature for an Assessment Task in ECS Unit 2: Problem Solving



Below is one method the principal could use to solve the problem.

Method 1:

1. Start with Ajay. Do steps 2 through 4, then move to the next student alphabetically on the list and repeat.
2. If the student's 1st choice club has an opening, then assign the student to that club.
3. If the student's 1st choice club is full, then check their 2nd choice. If their 2nd choice has an opening, then assign the student to their 2nd choice club.
4. If their 2nd choice club is full, then assign them to their 3rd choice club.

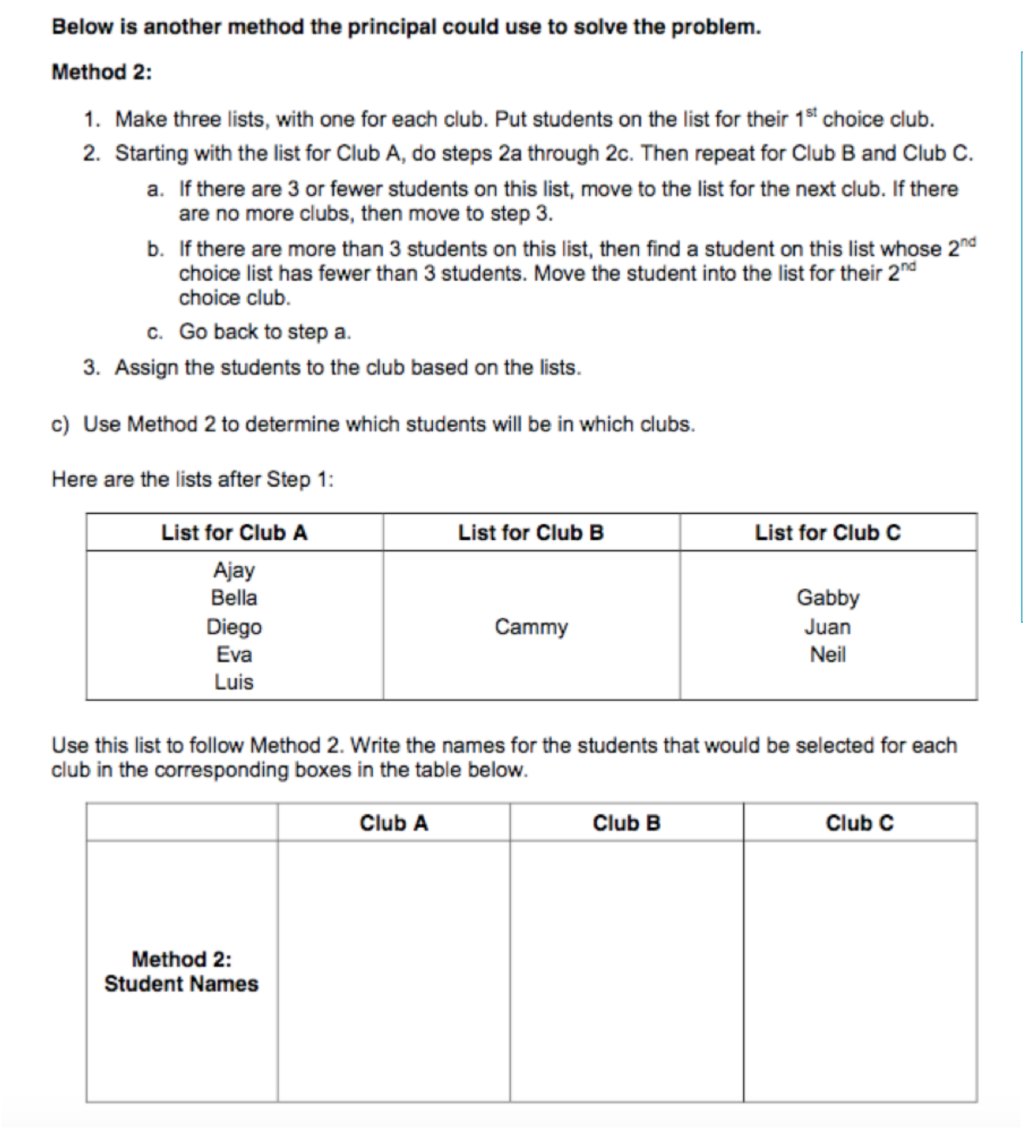
b) Use Method 1 to determine which student will be in which club. Write the names for the students that would be selected for each club in the corresponding boxes in the table below.

	Club A	Club B	Club C
Method 1: Student Names			

Example Characteristic Feature

The presence of an algorithm and a set of inputs is a characteristic feature of the task; the particular nature of the algorithm and inputs may vary.

Exhibit 8. Example Variable Features for an Assessment Task in ECS Unit 2: Problem Solving



Below is another method the principal could use to solve the problem.

Method 2:

1. Make three lists, with one for each club. Put students on the list for their 1st choice club.
2. Starting with the list for Club A, do steps 2a through 2c. Then repeat for Club B and Club C.
   a. If there are 3 or fewer students on this list, move to the list for the next club. If there are no more clubs, then move to step 3.
   b. If there are more than 3 students on this list, then find a student on this list whose 2nd choice list has fewer than 3 students. Move the student into the list for their 2nd choice club.
   c. Go back to step a.
3. Assign the students to the club based on the lists.

c) Use Method 2 to determine which students will be in which clubs.

Here are the lists after Step 1:

List for Club A	List for Club B	List for Club C
Ajay Bella Diego Eva Luis	Cammy	Gabby Juan Neil

Use this list to follow Method 2. Write the names for the students that would be selected for each club in the corresponding boxes in the table below.

	Club A	Club B	Club C
Method 2: Student Names			

Example Variable Feature

The complexity of the algorithm and the number of outputs required are also variable, allowing for adjustments in difficulty.

The degree of scaffolding provided in the task determines the level of cognitive load.

Moving from Design Patterns to Assessment Tasks

When domain analysis and domain modeling are complete, the assessment argument represented in design patterns is further described as specifications for tasks and assessments, evaluation procedures, and measurement models. In this ECD layer, known as the conceptual assessment framework (see Exhibit 1), the nuts and bolts of how an assessment will become operational are determined. Assessment designers decide which FKSAs the tasks will cover and what modifications (if any) need to be done to make them align with particular learning objectives in a curriculum. At this stage, the FKSAs from the design patterns are expressed as variables that characterize the student model.

Designers also articulate an evidence model, detailing how tasks will be scored, what measurement models will be applied to the scores, and the meaning of the scores. It is the combination of these two models—student and evidence—that defines precisely what will be inferred about the student's performance based on the assessment. Assessment designers also define the task model at the conceptual assessment framework layer. The task model specifies the number and types of tasks to be included and describes the specific requirements of the assessment, such as the format of the items (e.g., paper and pencil) and practical details of administration such as the amount of time a student has to complete it.

Note that the student, evidence, and task models are not created in a strict linear manner; rather, assessment designers refine the models simultaneously, iterating among them to bring them into alignment and to prepare a logically coherent foundation for the new assessment. Thus, as the task model is created, it may influence the earlier work on the student and evidence models.

Finally, in the assessment implementation and delivery layers, items and assessment forms are developed, reviewed, and validated in accordance with standards for validity (American Educational Research Association, American Psychological Association, & National Council on Measurement in Education, 2014). For example, early piloting work can include think-alouds with students as they complete the tasks, expert reviews, and pilot testing with a sample of students close to the target population to calibrate the item difficulty and determine whether differences exist among subgroups of students (e.g., differential item functioning analyses; see Osterlind & Everson, 2009). Later field-testing may involve larger and more diverse samples of students. Further psychometric work would be conducted at that point.

Summary and Next Steps

Evidence-centered design guides our collection of evidence for building a validity argument for the ECS assessments. The SRI team has conducted two years of pilot testing as part of the CS3 study (DRL-1418149) and expects to release a technical report with preliminary validity evidence during summer 2017.

The design patterns currently focus on computational thinking practices and therefore do not cover all of the knowledge also required for engaging in computational thinking. This is in part because computational thinking practices can be engaged in in different contexts, so depending on the context or content area, the knowledge required may be different. When developing an assessment for a particular content area, it is important to identify the knowledge that may go along with the practices. By attending to both context and relevant knowledge, the assessment developer can create tasks that discern whether a student is struggling with conceptual knowledge or with computational practices. For building assessments of practices, the domain-specific knowledge may serve as an additional rather than a focal knowledge or skill. Such items would provide classroom teachers with diagnostic information.

To this end, the SRI team is in the process of extending the ECS design patterns to include FKSAs for the computer science conceptual knowledge underlying the computational thinking practices in each unit and leveraging the revised design patterns to guide the development of multiple-choice assessment items measuring the computer science conceptual knowledge. We are also exploring the integration of dynamic, interactive features in some of these assessment items. Finally, we are developing an item bank to house all the ECS assessment items in a web-based, searchable format. Initially, ECS teachers will be able to log in to the item bank, search for items by curriculum unit and learning objective, review items, and download assessment forms. Future iterations will also allow ECS teachers to search for items aligned with relevant standards (e.g., Computer Science Teachers Association [CSTA]) and to build their assessments to particular uses (e.g., formative vs. summative evidence). The new items and item bank will be developed and piloted during the 2017–18 school year and will be available to ECS teachers starting the 2018–19 school year.

The ECS assessment tasks were purposely designed to each include several short constructed-response questions, which are more complex and time-consuming to score than multiple-choice questions. To address this challenge, SRI is exploring the application of an automated scoring engine. This engine would take in student responses and provide individual scores for the students. Initial tests have shown promise in that the degree to which the scores from the engine matched human scorers' final scores was similar to the degree that multiple human scorers agreed with each other.

We also know that ECS teachers struggle with effectively using assessment results to guide their instruction. To help ECS teachers address this challenge, SRI is launching a new study, *Teacher Assessment Literacy for Exploring Computer Science* (TALECS; CNS – 1640237), to investigate the design and delivery of high-quality assessment literacy materials and sustainable ongoing training as part of the ECS teacher professional development workshops. As computer science education reform efforts scale nationally, it will be essential to equip teachers new to the discipline with the skills and knowledge they need to gather evidence of student learning and design experiences to move it forward. More information about the study can be found at http://pact.sri.com/projects.html.

Our experience is that a rigorous and principled approach to assessment design yields not only valid assessments of well-defined knowledge and skills, but also templates for new sets of assessments that can be used to measure the same knowledge or skills. The design patterns described in this report represent a first step to building such templates so that computational thinking practices in the ECS curriculum can be assessed with different delivery formats.

References

American Educational Research Association, American Psychological Association, & National Council on Measurement in Education. (2014). *Standards for educational and psychological testing.* Washington, DC: American Educational Research Association.

Arpaci-Dusseau, A., Astrachan, O., Barnett, D., Bauer, M., Carrell, M., Dovi, R., … Uche, C. (2013). Computer science principles: Analysis of a proposed advanced placement course. In *Proceeding of the 44th ACM technical symposium on computer science education* (pp. 251–256). ACM. https://doi.org/10.1145/2445196.2445273

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology, 38*(6), 20–23. Retrieved from http://eric.ed.gov/?id=EJ918910

Bienkowski, M., Snow, E. B., Rutstein, D. W., & Grover, S. (2015). *Assessment design patterns for computational thinking practices: A first look. Menlo Park, CA: SRI International.* Retrieved from http://pact.sri.com/resources.html

Cheng, B. H., Ructtinger, L., Fujii, R., & Mislevy, R. (2010). *Assessing systems thinking and complexity in science (large-scale assessment technical report 7).* Menlo Park, CA: SRI International.

DeBarger, A. H., & Snow, A. (2010). *Design pattern on model use in interdependence among living systems (large-scale assessment technical report 13).* Menlo Park, CA: SRI International.

Denning, P. J. (2007). Computing is a natural science. *Communications of the ACM, 50*(7), 13–18.

Grover, S., & Pea, R. (2013). *Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42*(1), 38–43.

Haertel, G. D., Vendlinski, T. P., Rutstein, D., DeBarger, A., Cheng, B. H., Snow, E. B., … Ructtinger, L. (2016). General introduction to evidence-centered design. In H. Braun (Ed.), *Meeting the challenges to measurement in an era of accountability* (pp. 107–148). New York, NY: Routledge.

Margolis, J., Goode, J., & Chapman, G. (2015). An equity lens for scaling: A critical juncture for exploring computer science. *ACM Inroads, 6*(3), 58–66.

Messick, S. (1994). The interplay of evidence and consequences in the validation of performance assessments. *Educational Researcher, 23*(2), 13–23.

Mislevy, R. J., & Haertel, G. D. (2006). Implications of evidence-centered design for educational testing. *Educational Measurement: Issues and Practice, 25*(4), 6–20.

Mislevy, R. J., & Riconscente, M. M. (2006). Evidence-centered assessment design. In T. M. Haladyna & S. M. Downing (Eds.), *Handbook of test development* (pp. 61–90). New York, NY: Routledge.

Mislevy, R. J., Riconscente, M. M., & Rutstein, D. W. (2009). *Design patterns for assessing model-based reasoning (large-scale assessment technical report 6).* Menlo Park, CA: SRI International.

Mislevy, R. J., Steinberg, L. S., & Almond, R. G. (2003). Focus article: On the structure of educational assessments. *Measurement: Interdisciplinary Research and Perspectives, 1(*1), 3–62.

Osterlind, S. J., & Everson, H. T. (2009). *Differential item functioning* (2nd ed., Vol. 161). Thousand Oaks, CA: SAGE.

Szalay, A., & Gray, J. (2006). 2020 Computing: Science in an exponential world. *Nature, 440*(7083), 413–414.

Appendix: Exploring Computer Science Design Patterns and Example Assessment Tasks

In this section, we present the design pattern elements for each of the first four foundational ECS units along with example assessment tasks aligned with those elements:

- Unit 1: Human Computer Interaction
- Unit 2: Problem Solving
- Unit 3: Web Design
- Unit 4: Introduction to Programming

Unit 1: Human Computer Interaction

Overview

This unit introduces students to the concepts of a computer and computing while investigating the major components of computers and the suitability of these components for particular applications. Students will experiment with Internet search techniques, explore a variety of websites and web applications and discuss issues of privacy and security. Fundamental notions of Human Computer Interaction (HCI) and ergonomics are introduced. Students will learn that "intelligent" machine behavior is not "magic" but is based on algorithms applied to useful representations of information, including large data sets. Students will learn the characteristics that make certain tasks easy or difficult for computers, and how these differ from those that humans characteristically find easy or difficult. Students will gain an appreciation for the many ways in which computing-enabled innovations have had an impact on society, as well as for the many different fields in which they are used. Connections among social, economic and cultural contexts will be discussed.

Focal KSAs

About Computers

1. Ability to explain why an object is or is not a computer
2. Ability to evaluate a computer for the suitability to a particular application
3. Ability to compare the components of two or more computers for the suitability to a particular application

About Computation

4. Ability to explain why an activity or task is or is not an example of a problem a computer can solve

Using the Web

5. Ability to use web resources to find information
6. Ability to evaluate whether a particular web-based resource meets the needs of a user or problem

7. Ability to explain the extent to which results of a web search are trustworthy
8. Ability to explain how the results of a web search meet the purpose of the search
9. Ability to compare the results from multiple web searches based on criteria

Impacts of Computing

10. Ability to evaluate how computing impacts different contexts
11. Ability to describe the benefits of computing innovations
12. Ability to explain how computing innovation has led to new types of legal, ethical and privacy concerns

Communication and Data Exchange

13. Ability to explain the relationship between communication, data exchange, and/or computing devices
14. Ability to evaluate the implications of a form (or various forms) of data exchange (communication) on social interactions (including security and privacy concerns)
15. Ability to evaluate/compare how specific representations of data are used to communicate information

Computer Programs and Machine Intelligence

16. Ability to describe or demonstrate the characteristics of a computer program
17. Ability to describe the difference between intelligence as it relates to humans and computers

Characteristic Features

About Computers

- The student must be presented with an object.
- The object must have clear characteristics that allow the evaluation of whether it is a computer.
- The task must provide the student with information about a computer and information about the application it will be used for.

About Computation

- The student must be presented with an activity or task.

Using the Web

- The student must be given a specific set of information to find.
- The task must include information about the web-based resource and the need.
- The task must involve one web search or the results of a web search.
- The task must involve one web search.
- The task must present a collection of web sites as the result of one search.
- The task must involve at least two web searches.

Impacts of Computing

- The task must involve one or more contexts outside of traditional computing, computer science or engineering.

Communication and Data Exchange

- The task must ask the students to relate forms of communication with computing devices.
- The task must provide the student with a particular form of data exchange.
- The task does not include specific representations of data.
- The task must provide a specific representation of data (for evaluate) and multiple representations of data (for compare).

Computer Programs and Machine Intelligence

- The task must provide an example of a computer program.

Variable Features

About Computers

- Whether the object could be considered a computer or not
- Whether students would be able to argue either way if the object is a computer or not
- The number and type of characteristics that would indicate whether the object is a computer
- The degree to which characteristics of the object consistently suggest a computer vs. not
- The degree to which the important characteristics are explicitly stated in the problem or must be inferred by the test taker
- Type of application
- Level of details specified about the application and the computer
- Knowledge of particular computer components required

About Computation

- Whether the activity or task could be considered computing or not
- Whether students would be able to argue either way about if the activity or task was an example of computing

Using the Web

- The format of the report of the student's search and resulting information
- The type of information wanted (along with which web resources would be most appropriate for finding this information)
- The web resources available to the students
- The number of searches the student is allowed to perform to find the information
- The web-based resource
- The needs of the user or problem
- The degree to which the web-based resource matches the needs of the user or problem
- The clarity with which the resources match the needs of the problem (i.e., level of inference required by the test taker)

- The degree to which the results of a web search are trustworthy
- Whether students are provided the web search results or generate the results
- The amount of evidence required in the explanation
- Whether students are given the web search results or are asked to generate the results (Note: the latter is not feasible for a unit test)
- The degree to which the results of the web search meet the purpose of the search
- The amount of explicit vs. implicit purposes in the problem; the degree to which different purposes have higher priority than others
- Whether students are given the web search results or are asked to generate the results
- Whether the student is allowed to determine the criteria or it is given to them
- The degree to which the results of the web searches match the criteria
- The degree to which the results of the web searches match each other

Impacts of Computing

- The context or contexts
- Whether or not the students are given the context(s)
- The level of detail required in the explanation/comparison
- The number of contexts in which the innovation might apply
- The degree to which the context is familiar
- Whether students are given a particular computing innovation or asked to discuss computing in general
- The level of detail required of the explanation
- The degree to which the task separates legal, ethical and privacy concerns

Communication and Data Exchange

- Whether the task involves a specific computing device, or a specific example of communication, or if the task is more general
- Whether the task explicitly asks students to address the concept of data exchange, or if this is an implicit part of the task
- The form of data exchange provided
- The level of detail required in the explanation
- The representation used
- The amount and type of information that can be found from the representation
- Comparing the degree to which the representations are similar

Computer Programs and Machine Intelligence

- The degree to which the presented object is like a computer program
- Varying types of objects that function according to programs

Potential Observations

About Computers

- Appropriateness of the explanation of why an object is or is not a computer. (i.e., Did the student correctly identify aspects of the object that relate to aspects of a computer? Did the student correctly identify aspects of a computer that the object lacks?)
- Appropriateness of the evaluation of a computer for a particular application (i.e., Did the student correctly identify the features of the computer that match (or do not match) the needs of the application? Did the student support their judgment on the degree to which the computer matches the needs with evidence?)
- Appropriateness of the comparison of the components of two or more computers for a particular application (i.e., Did the student correctly identify differences in the components of the computers that relate to the application? Did the student correctly explain how the differences in the components affect the suitability for the application?)

About Computation

- Appropriateness of the explanation of why an activity or task is or is not computing. (i.e., Did the student correctly identify aspects of the activity or task that relate to aspects of computing, Did the student correctly identify aspects of computing that the activity or task lacks?)

Using the Web

- Appropriateness of the web resources to the topic searched (i.e., Did the student use a web resource that matches the problem/topic?)
- Degree to which the information found matches the information wanted
- Appropriateness of the evaluation of a web-based resource (i.e., Did the student describe aspects of the web-based resource in relation to the needs of the user or problem? Did the student identify aspects of the web-based resource that do not match the needs of the user or problem?)
- Appropriateness of the explanation of the extent to which results of a web search are trustworthy (i.e., Did the student include aspects of the web-page found from the search such as the source, the date, the professionalism of the web-page? Did the student explain the relationship between these aspects and the trustworthiness of the page)
- Appropriateness of the explanation of how the results of a web search meet the purpose of that search (i.e., Did the student describe how the results meet the initial question being asked? Did the student describe how the results do not meet the purpose of the search?)
- Appropriateness of the comparison of how the results of multiple web searches meet the purpose of that search (i.e., Did the student explain the similarities and differences between the multiple search results? Did the student relate those similarities and differences to set of criteria?)

Impacts of Computing

- Appropriateness of the explanation/comparison of how computing impacts different contexts (i.e., Did the student include aspects of computing in their explanation? Did the student describe a context outside of traditional computing, computer science, and engineering?)

- Appropriateness of the explanation (i.e., Did the student describe computing innovations? Did the student relate these innovations to legal, ethical and/or privacy concerns?)

Communication and Data Exchange

- Appropriateness of the explanation of the relationship between communication, data exchange, and computing devices (i.e., Did the student connect communication to data exchange? Did the student explain how computing devices could be used for communication (as well as data exchange?)
- The appropriateness of the evaluation of the implications of a particular form of data exchange (i.e., Did the student discuss privacy concerns? Did the student describe social aspects of the form of data exchange such as how personal the data exchange is? Did the student discuss benefits and drawbacks of this form of data exchange? Did the student discuss tradeoffs between the different forms of data exchange, including privacy concerns and social aspects? Did the student explain that different representations might communicate different information?)
- Appropriateness of the evaluation/comparison (i.e., Did the student describe the type of information that can be found from each representation? Did the student describe types of information that cannot be found? For compare--Did the student explain the tradeoffs between the different representations?)

Computer Programs and Machine Intelligence

- Appropriateness of the explanation of what a computer program is (i.e., Did the student explain that a computer program is stored in memory, fetched sequentially, and executed one by one?)
- Appropriateness of the description of intelligent behavior

Potential Work Products

About Computers

- An explanation of why an object is or is not a computer
- Evaluation of a computer for the suitability to a particular application
- Comparison of computers for the suitability to a particular application

About Computation

- An explanation of why an activity or task is or is not an example of computing

Using the Web

- The record of the student's search
- The information the student found
- The evaluation of a web-based resource
- An explanation of the extent to which results of a web search are trustworthy
- An explanation of how the results of a web search meet the purpose of the search
- A comparison of how the results of multiple web searches meet a set of criteria

Impacts of Computing

- An explanation/comparison of how computing impacts different contexts
- An explanation of how computing innovation has led to new types of legal, ethical, and privacy concerns

Communication and Data Exchange

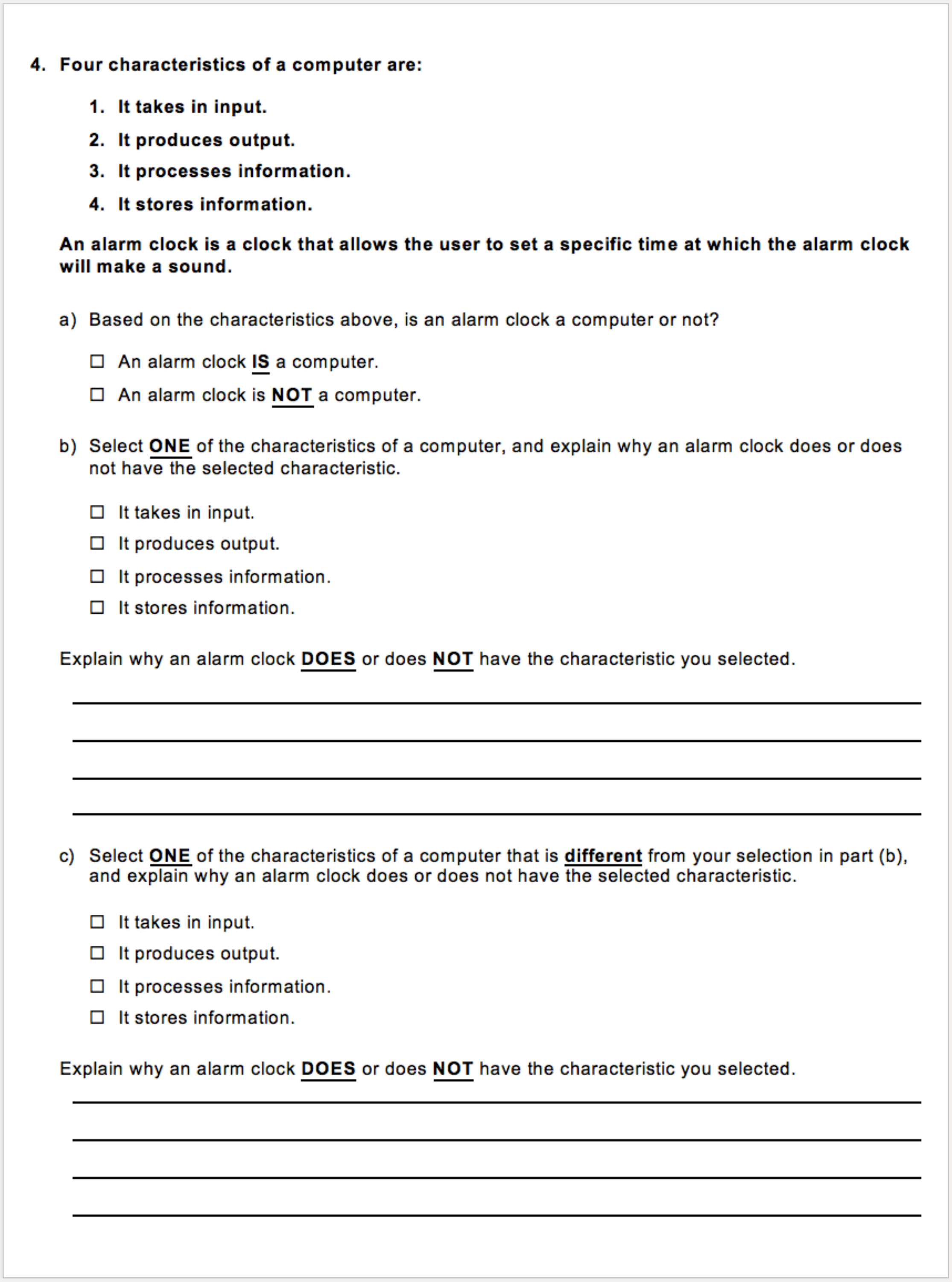
- An explanation of the relationship between communication, data exchange and computing devices
- The evaluation of the implications of a particular form of data exchange
- The comparison of the implications of various forms of data exchange
- An explanation of how representations of data are used to communicate information
- The evaluation/comparison of how specific representations of data are used to communicate information

Computer Programs and Machine Intelligence

- The explanation of a computer program
- A listing of the characteristics of a program
- A description of the problems a computer may have in interpreting instructions
- A description of intelligent and non-intelligent behaviors

Unit 1 Assessment Item Example: What is a computer?

- *FKSA 1: Ability to explain why an object is or is not a computer*



4. Four characteristics of a computer are:

1. **It takes in input.**
2. **It produces output.**
3. **It processes information.**
4. **It stores information.**

An alarm clock is a clock that allows the user to set a specific time at which the alarm clock will make a sound.

a) Based on the characteristics above, is an alarm clock a computer or not?

☐ An alarm clock **IS** a computer.

☐ An alarm clock is **NOT** a computer.

b) Select **ONE** of the characteristics of a computer, and explain why an alarm clock does or does not have the selected characteristic.

☐ It takes in input.

☐ It produces output.

☐ It processes information.

☐ It stores information.

Explain why an alarm clock **DOES** or does **NOT** have the characteristic you selected.

c) Select **ONE** of the characteristics of a computer that is **different** from your selection in part (b), and explain why an alarm clock does or does not have the selected characteristic.

☐ It takes in input.

☐ It produces output.

☐ It processes information.

☐ It stores information.

Explain why an alarm clock **DOES** or does **NOT** have the characteristic you selected.

Unit 2: Problem Solving

Overview

This unit provides students with opportunities to become "computational thinkers" by applying a variety of problem-solving techniques as they create solutions to problems that are situated in a variety of computational contexts. The range of contexts motivates the need for students to think abstractly and apply known algorithms where appropriate, but also create new algorithms. Analysis of various solutions and algorithms will highlight problems that are not easily solved by computer and for which there are no known solutions. This unit also focuses on the connections between mathematics and computer science. Students will be introduced to selected topics in discrete mathematics including Boolean logic, functions, graphs and the binary number system. Students are also introduced to searching and sorting algorithms and graphs.

Focal KSAs

Algorithms

1. Ability to state what an algorithm would output given a set of inputs
2. Ability to explain the inputs of an algorithm, how it operates on that input, and what the outputs are
3. Ability to evaluate the extent to which an algorithm solves a stated problem
4. Ability to compare the tradeoffs between different algorithms for solving the same problem
5. Ability to create (using a narrative description or a representation) an algorithm that addresses a set of specifications

Problem Solving

1. Ability to describe the steps to solving a problem
2. Ability to evaluate how an approach/strategy solves a problem
3. Ability to enact the steps of a problem-solving process in order to address a need

Problem Solving with Computing

4. Ability to evaluate features of a task that made it appropriate as a computing problem

Math and Computer Science

5. Ability to explain connections between elements of mathematics and computer science
6. Ability to recognize patterns or describe patterns

Collaborative Problem Solving and Communication

7. Ability to solve a problem as a group by distributing the workflow among group members and then combining the results
8. Ability to use multiple sources of feedback to develop a solution to a problem
9. Ability to communicate a problem, the solution process, and the final solution

10. Ability to collaboratively communicate a problem, the solution process, and the final solution

Characteristic Features

Algorithms

- The task must provide the student with an algorithm and a set of inputs.
- The task must provide the student with an algorithm.
- The task must provide students with information about the algorithm and the problem the algorithm is addressing.
- The task must provide students with information about the algorithm and the problem the algorithm is addressing.
- The task must provide a set of specifications.

Problem Solving

- The task must involve asking the students to explicitly describe steps in the problem solving process.
- The task must provide a problem to the student.
- The task must involve a problem and documentation of the problem solving process.

Problem Solving with Computing

- The task must provide a task to the student.

Math and Computer Science

- The task must explicitly ask students to discuss connections with mathematics.

Collaborative Problem Solving and Communication

- The task must have 2 or more people creating a problem solution.
- The task involves an already created problem solution.
- The task must include feedback on that problem solution.
- The task must specify the problem solution to be used in the communication.
- The task must provide guidelines for the format and length of the communication.
- The task must specify the problem solution to be used in the communication.
- The task must provide guidelines for the format and length of the communication.

Variable Features

Algorithms

- The algorithms being used
- The inputs to the algorithm
- The complexity of the algorithm and the inputs

- The number of outputs
- The outputs of the algorithm
- The complexity of the algorithm
- The information provided about the problem the algorithm will be used to solve
- The degree to which the algorithm solves the stated problem
- The degree to which each algorithm solves the problem
- The degree to which the two algorithms are similar to one another
- The amount and type of information provided about the problem and the algorithms
- The complexity of the set of specifications
- The representation required of the student

Problem Solving

- Whether students are asked to describe the steps in general or in the context of a specific problem
- The problem (and complexity of the problem) that is to be solved
- Whether the approach/strategy is provided to the student or generated by the student
- The degree to which the approach/strategy solves a problem
- The degree to which the approaches/strategies are similar
- The complexity of the problem
- The degree to which the steps in the process are documented
- The format of the documentation (e.g., oral accounts, written documentation)

Problem Solving with Computing

- The complexity of the task
- The degree to which the task is appropriate for a computing problem

Math and Computer Science

- Whether students are asked to explain connections in general or connections as related to a specific problem
- The knowledge of mathematical concepts required
- The knowledge of computer science concepts required

Collaborative Problem Solving and Communication

- The number of members in a group
- The complexity of the problem solution to be designed
- The level of specifications provided to the group
- The complexity of the feedback
- The amount of disagreement of the feedback
- The source of the feedback

- The complexity of the initial problem solution
- The level of detail asked for in the communication
- The format of the communication (power point, poster, oral, written…)
- The length of the communication
- Audience for the communication
- The number of participants in the group

Potential Observations

Algorithms

- Correctness of the identified outputs
- Accuracy of the explanation of the parts of an algorithm (i.e., Did the student correctly identify the inputs of an algorithm? Did the student correctly and completely describe the operations performed on those inputs? Did the student correctly describe the outputs?)
- Appropriateness of the evaluation (i.e., Did the student evaluate the algorithm for the boundary cases as well as for the general cases? Did the student make a correct judgment on whether or not the algorithm solves the problem for these cases? Did the student support their judgments with an explanation?)
- Appropriateness of the comparison (i.e., Did the student compare the algorithms at the boundary cases as well as at the general case, did the student make a correct judgment on whether or not each algorithm solves the problem for these cases, did the student support their judgments with an explanation, did the student state the trade-offs between the two algorithms)
- Appropriateness of the algorithm created (i.e., Does the algorithm meets the set of specifications? Does the algorithm works at the boundary cases? Is the algorithm high level?)

Problem Solving

- Appropriateness of the description of the steps (i.e., Did the student clearly separate each of the steps in their description: Understanding the problem, creating a plan, implementing the plan, review and reflect on the solution? Did the student provide a clear description of each of these steps?)
- Appropriateness of the evaluation (i.e., Does the student discuss how the approach/strategy addresses the problem? Does the student include the limitations of the approach/strategy? Does the student include the tradeoffs between the different approaches/strategies?)
- The degree to which the problem solving process was followed when solving a problem (i.e., Do the students demonstrate that they understood the problem? Did they create a plan and follow that plan? Did they review their work?)

Problem Solving with Computing

- Appropriateness of the evaluation (i.e., Does the student correctly match features of the task to features of a computing problem? Does the student identify limitations of computing when solving this task? Does the student identify what features of the task make it unsuitable for computing, if any?)

Math and Computer Science

- Appropriateness of the explanation (i.e., Did the student identify elements of mathematics that are related to computer science? Does the student include an explanation of the set of elements along with specific examples?)

Collaborative Problem Solving and Communication

- Appropriateness of the division of workflow (i.e., Did all members have work to do? Was the work that was assigned to each member appropriate and matches to that member's skills?)
- Quality of the produced problem solution (i.e., Does the problem solution include all of the required pieces? Does it solve the problem?)
- Appropriateness of the revisions to a problem solution based on provided feedback (i.e., Were all comments addressed in some way? Were disagreements in feedback resolved?)
- Clarity of the communication (i.e., Was the description understandable? Was the communication organized?)
- Completeness of the communication (i.e., Did the communication address all of the sections of the problem solution? Was the description of the design process comprehensive?)"
- Degree of collaboration (i.e., Did each member of the group contribute to the communication?)
- Clarity of the communication (i.e., Was the description understandable? Was the communication organized?)
- Completeness of the communication (i.e., Did the communication address all of the sections of the problem solution? Was the description of the design process comprehensive?)

Potential Work Products

Algorithms

- The outputs of an algorithm
- The explanation of the algorithm
- The evaluation of the extent to which an algorithm solves a stated problem
- The comparison of the tradeoffs between different algorithms for solving a stated problem
- A representation of an algorithm

Problem Solving

- A description of the steps to solving a problem
- The explanation of how an approach/strategy solves a problem
- The comparison of how multiple approaches/strategies solves a problem
- Documentation of the problem solving process such as a description of the problem, a plan, a description of how students followed the plan, and a review of the solution.

Problem Solving with Computing

- The evaluation of a task as a computing problem

Math and Computer Science

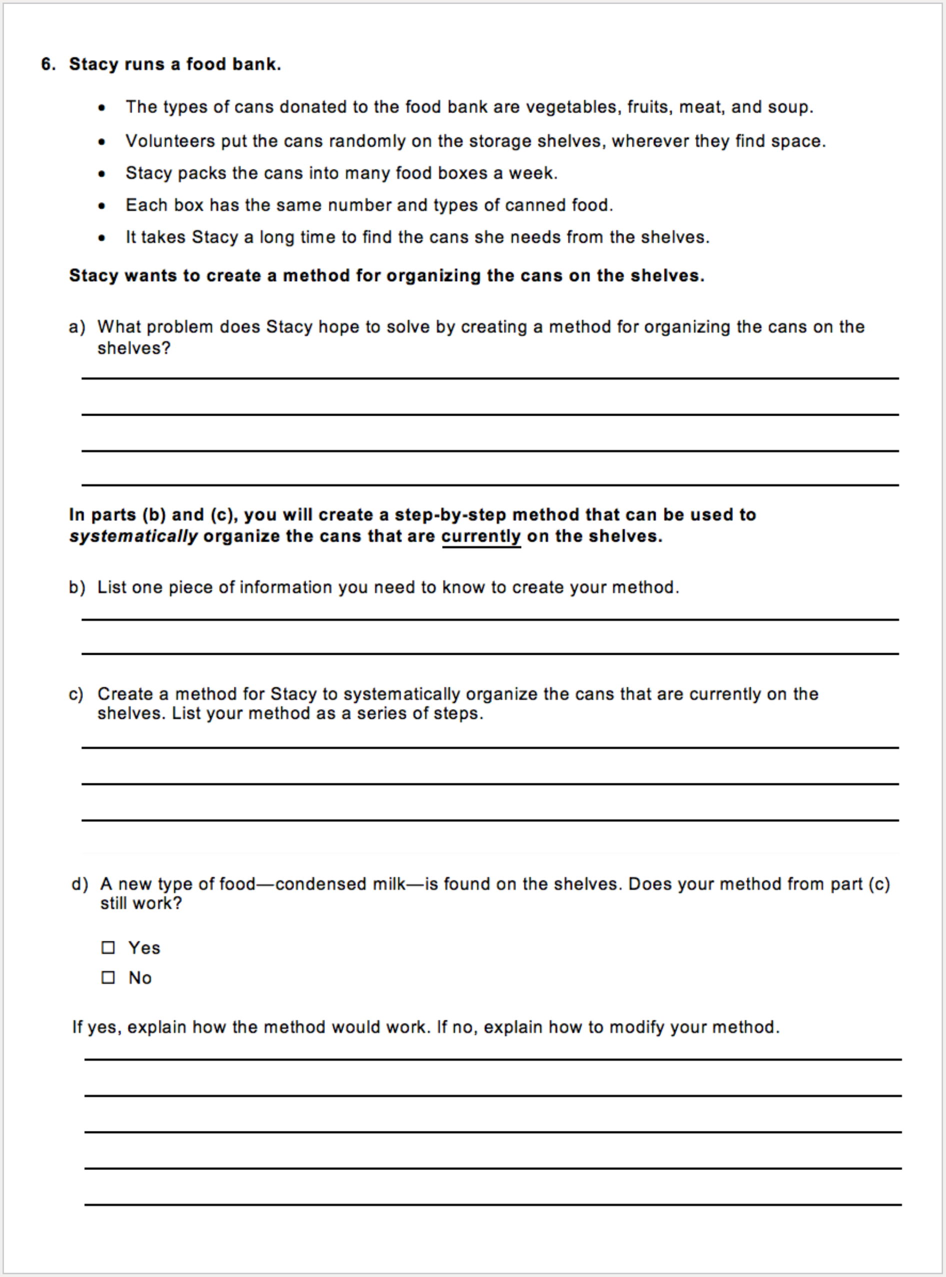
- The explanation of the connection between elements of mathematics and computer science

Collaborative Problem Solving and Communication

- The resulting problem solution
- A description or list of the assignment of the workflow
- The resulting problem solution
- A description of the comments and changes that were made and/or a description of how the comments were addressed
- A communication (could be power point, could be oral, could be written) about a problem solution
- A documentation of how each member of the group contributed to the communication

Unit 2 Assessment Item Example: Food Bank

- *FKSA 2: Ability to explain the inputs of an algorithm, how it operates on that input, and what the outputs are*
- *FKSA 5: Ability to create (using a narrative description or a representation) an algorithm that addresses a set of specifications*
- *FKSA 8: Ability to follow the steps in a problem solving process in order to address a need*



6. Stacy runs a food bank.

- The types of cans donated to the food bank are vegetables, fruits, meat, and soup.
- Volunteers put the cans randomly on the storage shelves, wherever they find space.
- Stacy packs the cans into many food boxes a week.
- Each box has the same number and types of canned food.
- It takes Stacy a long time to find the cans she needs from the shelves.

Stacy wants to create a method for organizing the cans on the shelves.

a) What problem does Stacy hope to solve by creating a method for organizing the cans on the shelves?

In parts (b) and (c), you will create a step-by-step method that can be used to *systematically* organize the cans that are currently on the shelves.

b) List one piece of information you need to know to create your method.

c) Create a method for Stacy to systematically organize the cans that are currently on the shelves. List your method as a series of steps.

d) A new type of food—condensed milk—is found on the shelves. Does your method from part (c) still work?

☐ Yes
☐ No

If yes, explain how the method would work. If no, explain how to modify your method.

Unit 3: Web Design

Overview

This unit prepares students to take the role of a developer by expanding their knowledge of algorithms, abstraction, and web page design and applying it to the creation of web pages and documentation for users and equipment. Students will explore issues of social responsibility in web use. They will learn to plan and code their web pages using a variety of techniques and check their sites for usability. Students learn to create user-friendly websites. Students will apply fundamental notions of Human Computer Interaction (HCI) and ergonomics.

Focal KSAs

Design and Implement Web Pages

1. Ability to create a set of specifications for a web page given the intent of the web page
2. Ability to design a web page based on specified objectives
3. Ability to implement a web page based on specified objectives
4. Ability to compare two web pages based on provided criteria
5. Ability to describe techniques used when designing and implementing a web page
6. Ability to compare different techniques used in designing and implementing a web page
7. Ability to use techniques when designing and implementing a web page
8. Ability to apply abstraction to separate style from content when designing and implementing a web page

Evaluate and Debug Web Page Implementation

9. Ability to evaluate the extent to which a web page meets specified objectives
10. Ability to compare design decisions in relation to the user experience
11. Ability to identify errors in given web page code
12. Ability to explain why specific errors have occurred in web pages and how to correct them
13. Ability to correct errors in web page code

Collaborative Web Design and Communication

14. Ability to create a web page as a group by distributing the workflow among group members then combining the results
15. Ability to use multiple sources of feedback to develop a web page
16. Ability to communicate the objectives of a web page, the design process and the web page
17. Ability to collaboratively communicate the objectives of a web-page, the design process, and the web page

Characteristic Features

Design and Implement Web Pages

- Each task must provide students with an overall intent for the web page.
- Each task must ask students to generate specifications.
- The task must specify the objectives of the web page.
- The task must provide information on the format requirements of the design.
- The task must provide students with the web-page and the objectives.
- The task must provide students with two web-pages and the criteria for the comparison.
- The task must specify a technique that can be used when designing and/or implementing a web page.
- The task must be about techniques used during web-page development.
- The task is about the implementation of a technique.
- The task must explicitly indicate to the students that they are separating style from content.

Evaluate and Debug Web Page Implementation

- The task explicitly indicates to the student that they must discuss the usability (but they don't have to use that term) of the web page.
- The task must include a web page with an error in it (the error should be straightforward to find for this unit).
- The task must provide the error to the student.
- The task must ask the student to fix an error in web page code.
- The task must have 2 or more people creating a web page.

Collaborative Web Design and Communication

- The task involves an already created web page.
- The task must include feedback on that web page.
- The task must specify the web page to be used in the communication.
- The task must provide guidelines for the format and length of the communication.

Variable Features

Design and Implement Web Pages

- The intent of the web page, beyond the intent itself, one might vary the level of specificity or ambiguity in the intent, the degree to which the intent could be achieved by multiple types of specifications, the degree to which students must infer aspects of the intent (i.e., something important, but implicit).
- Can include practical, personal, and/or societal intents
- The level of detail required from the student in the set of specifications
- The format required of the specifications

- The amount of scaffolding provided for the development of the specifications
- The format the students should use for their design
- The level of detail needed for the design
- The complexity of the objectives
- The format of the response (i.e., The response should be more than just a checklist where students check off if the web page meets an objective. The response could be a checklist with an explanation of how each of the specifications was met or not met.)
- Complexity of the criteria
- Degree of similarity between the web pages
- Whether the technique is given to the student or the student is able to pick the technique
- Whether the techniques are given to the student or generated by the student
- The degree of similarity between the techniques
- Are students told which techniques to use, or are they allowed to pick (if they pick they need to identify which technique they are using)
- The degree to which the documentation of the design of the web page is required
- The purpose of the web page
- The representation of the web page required
- The degree to which the documentation of the design of the web page is required
- The purpose of the web page

Evaluate and Debug Web Page Implementation

- The design decision(s) being compared
- The comparability of the design decision(s)
- The amount of context given for the task (how much does the student know about the web-page)
- The representation of the web-page code (are they given the web-page, or the html code or both?)
- The representation of the identification of the errors or the fix for the errors
- The level of complexity of the error
- The level of complexity of the fix for the error
- The number of members in a group
- The complexity of the web page to be designed
- The level of specifications provided to the group

Collaborative Web Design and Communication

- The complexity of the feedback
- The amount of disagreement of the feedback
- The source of the feedback
- The complexity of the initial web page

- The level of detail asked for in the communication
- The format of the communication (PowerPoint, poster, oral, written…)
- The length of the communication
- Audience for the communication
- The number of participants in the group

Potential Observations

Design and Implement Web Pages

- Completeness of the description of the specifications (i.e., Did the student include a discussion of the relevant parts of a web page such as headings and menus?)
- Appropriateness of the set of specifications (i.e., How well do the specifications match the intent? Is the student providing space for all of the relevant content information?)
- Appropriateness of the organization of the specifications (i.e., Did the student specify multiple pages? If so, are there specifications for how the pages should be linked and/or navigated? Do the specifications clearly state where to put different types of information?).
- Degree to which the design of the web page matches the specified objectives (i.e., Did the student address and satisfy all of the objectives?)
- Completeness of the evaluation (i.e., Did the student include information on every objective?)
- Appropriateness of the evaluation (i.e., Did the student provide a reasonable explanation for their evaluation? Did the student accurately reflect the web page in their evaluation?)
- Completeness of the comparison (i.e., Did the student include information on every objective?)
- Appropriateness of the comparison (i.e., Did the student provide a reasonable explanation of the similarities and differences between the web-pages? Did the student accurately reflect the web pages in their comparison?)
- Accuracy of the description of a technique used when designing and implementing a web-page (i.e., Did the student cover the main points of the technique? Does the student make it clear how to use a technique and/or when this technique would be applied?)
- Completeness of the comparison (i.e., Did the student include information on the main similarities and differences between the techniques?)
- Appropriateness of the comparison (i.e., Did the student provide a reasonable explanation of the similarities and differences between the techniques? Did the student accurately reflect the techniques in their comparison?)
- Correctness of the use of a technique in the design and implementation of a web page. (i.e., Did the student design the web-page taking the technique into account? Did the student use the technique? If so, did they implement it correctly?)
- Degree to which style is separated from the content in the design and/or implementation of a web-page (i.e., Did the student use style sheets? Were there places on the web page where style was integrated with the content that could have been separated?)

Evaluate and Debug Web Page Implementation

- Appropriateness of the comparison of the design decisions (i.e., Did the student discuss trade-offs of the decisions? Does the explanation accurately reflect the design decisions?)
- Correctness of the identification of the errors
- Accuracy of the explanation for an error and the fix for that error (i.e., Did the student give a plausible cause for the error? Would the fix they describe work?)
- Appropriateness of the fix (i.e., Did the fix address the error, did the fix correct the error? Did the fix introduce new errors?)
- Appropriateness of the division of workflow (i.e., Did all members have work to do? Was the work that was assigned to each member appropriate and matched to that member's skills?)
- Quality of the produced web-page (i.e., Does the web-page include all of the required pieces? Is the style of the web page consistent throughout?)

Collaborative Web Design and Communication

- Appropriateness of the revisions to a web-page based on provided feedback (i.e., Were all comments addressed in some way? Were disagreements in feedback resolved?)
- Clarity of the communication (i.e., Was the description understandable? Was the communication organized?)
- Completeness of the communication (i.e., Did the communication address all of the sections of the web-page? Was the description of the design process comprehensive?)
- Degree of collaboration (i.e., Did each member of the group contribute to the communication?)

Potential Work Products

Design and Implement Web Pages

- The set of specifications for a web page
- Specifications could be sketches as other diagrams as well as statements
- A design of a web page
- The evaluation of the extent to which a web page meets specified objectives
- The comparison of two web pages
- The description of a technique
- The comparison of different techniques
- The design and implementation of a web page

Evaluate and Debug Web Page Implementation

- The explanation of how different design decisions might affect the users of the web page
- Identification of errors in web page code
- The explanation for an error and the explanation for the fix for that error

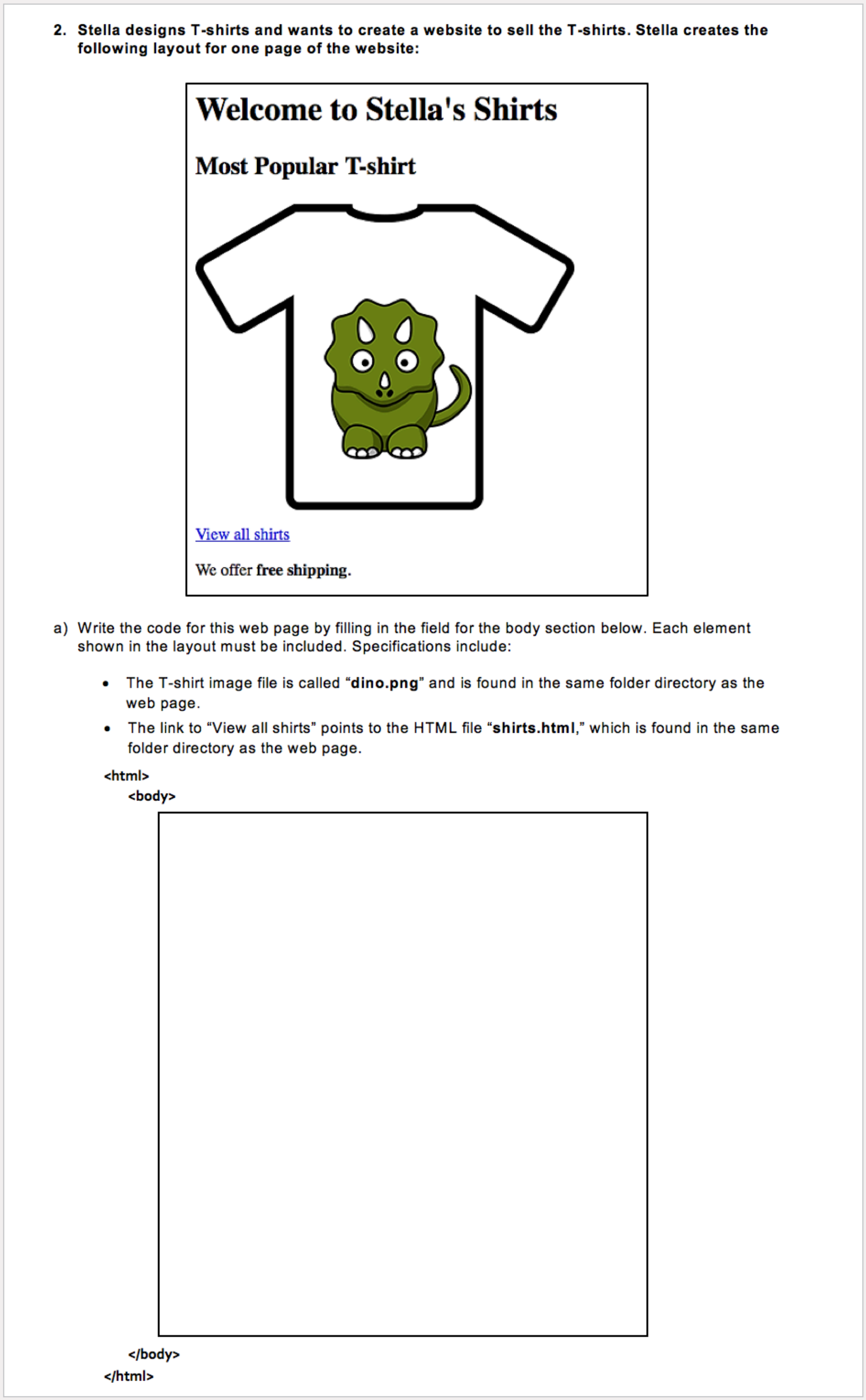
- Web page code with the fix implemented
- The resulting web page
- A description or list of the assignment of the workflow

Collaborative Web Design and Communication

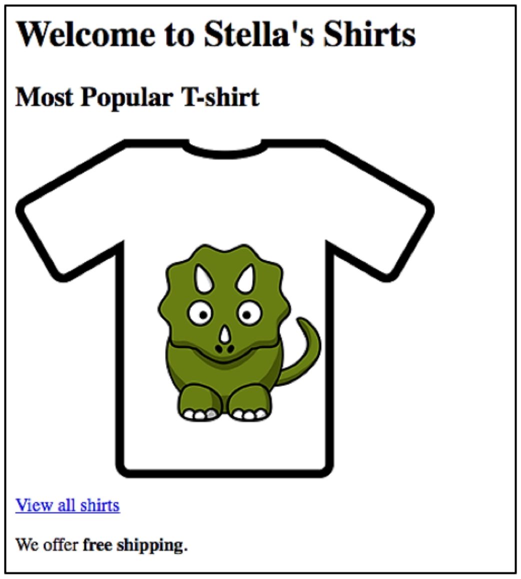
- The resulting web page
- A description of the changes that were made and/or a description of how the comments were addressed
- A communication (could be power point, could be oral, could be written) about a web page
- A documentation of how each member of the group contributed to the communication

Unit 3 Assessment Item Example: Stella's Shirts

• *FKSA 7: Ability to use techniques when designing and implementing a web-page*



2. **Stella designs T-shirts and wants to create a website to sell the T-shirts. Stella creates the following layout for one page of the website:**



Welcome to Stella's Shirts

Most Popular T-shirt

View all shirts

We offer **free shipping.**

a) Write the code for this web page by filling in the field for the body section below. Each element shown in the layout must be included. Specifications include:

- The T-shirt image file is called "**dino.png**" and is found in the same folder directory as the web page.
- The link to "View all shirts" points to the HTML file "**shirts.html**," which is found in the same folder directory as the web page.

```
<html>
    <body>



    </body>
</html>
```

Unit 4: Introduction to Programming

Overview

This unit introduces students to some basic issues associated with program design and development. Students design algorithms and create programming solutions to a variety of computational problems using an iterative development process in a blocks-based language such as Scratch. Programming problems include mathematical and logical concepts and a variety of programming constructs.

Focal KSAs

Programming Fundamentals

1. Ability to describe what is programming
2. Ability to create a set of specifications for a program given the intent of the program
3. Ability to explain the concept of a variable
4. Knowledge of the concept of a variable
5. Knowledge of Boolean logic

Algorithms in Programming

6. Ability to compare the tradeoffs between different algorithms for solving the same problem
7. Ability to create an algorithm that addresses a set of specifications

Programming Structures

8. Ability to describe features of a programming structure
9. Ability to evaluate the relationship between features of a programming structure and features of a problem or algorithm
10. Ability to compare the tradeoffs between different programming structures for solving the same problem

Programming Process

11. Ability to describe debugging and testing methods
12. Ability to evaluate debugging and testing methods in terms of how they relate to the problem or program
13. Ability to use debugging and testing methods
14. Ability to generate test cases for a program

Evaluating Programs

15. Ability to state what a program would output given a set of inputs
16. Ability to explain the inputs of a program, command or object, how it operates on those inputs, and

what the outputs are

17. Ability to evaluate the extent/degree to which a program solves a stated problem
18. Ability to compare the tradeoffs between different programs for solving the same problem

Collaborative Programming and Communication

19. Ability to create a program as a group by distributing the workflow among group members then combining the results
20. Ability to use multiple sources of feedback to develop a program
21. Ability to communicate the objectives of a program, the design process and the program
22. Ability to collaboratively communicate the objectives of a program, the design process and the program

Characteristic Features

Programming Fundamentals

- The task must make it clear to the student that they should include a description of the programming cycle.
- The task must include information on the intent of the program.
- The task must provide students with information about the algorithms and information about the problem the algorithms are addressing.
- The task must include a set of specifications that can be used to generate an algorithm.

Algorithms in Programming

- The task must include a set of specifications that can be used to generate an algorithm.

Programming Structures

- The task must provide students with a programming structure.
- The task must provide students with a programming structure and information on a problem or algorithm.
- The task must provide the student with a problem or algorithm.

Programming Process

- The task must specify the problem or program that is to be tested.
- The task must provide the student with a program to debug and test.
- The task should provide a program that would allow for different test cases to be generated for it.

Evaluating Programs

- The task should provide a program and a set of inputs.
- The task should provide a program.
- The task should provide a program as well as a problem.
- The task must include information on the programs being compared.

Collaborative Programming and Communication

- The task must have 2 or more people creating a program.
- The task involves an already created program.
- The task must include feedback on that program.
- The task must specify the program to be used in the communication.
- The task must provide guidelines for the format and length of the communication.

Variable Features

Programming Fundamentals

- Format of the description
- Amount of detail required in the description
- Level of detail required of the specifications
- The intent of the program
- The format of the response
- Format of the explanation
- Amount of detail required of the explanation
- Whether the student is describing variables in general, or describing a variable in relation to a specific program or problem
- The degree to which each algorithm solves the problem
- The degree to which the two algorithms are similar to one another
- The amount and type of information provided about the problem and the algorithms
- The required format of the description of the algorithm
- The level and amount of details of the provided specifications

Algorithms in Programming

- The required format of the generated algorithm
- The level and amount of details of the provided specifications

Programming Structures

- The programming structure that is being described
- The format of the description
- The level of detail required of the evaluation
- The degree to which the programming structure matches the features of the problem or algorithm
- The complexity of the problem or algorithm
- The level of detail required of the comparison
- The degree to which the programming structures match the features of the problem or algorithm

- Whether the student determines the programming structures to be compared or this is provided to them
- The degree to which the programming structures match each other

Programming Process

- The context for which students are describing these methods (generally, or in relation to a specific problem)
- The level of details required of the description
- The debugging and testing methods that are being evaluated
- The problem or program used, and how well the methods relate to the problem or program
- The number and type of errors in the program
- Whether a description of the methods used is required
- The complexity of the program
- The required number of test cases to be generated

Evaluating Programs

- The complexity of the program
- The number of inputs
- The number of outputs
- The complexity of the program
- The complexity of the problem
- The degree to with the program addresses the problem
- The programs being compared
- The degree to which the programs differ

Collaborative Programming and Communication

- The number of members in a group
- The complexity of the program to be designed
- The level of specifications provided to the group
- The complexity of the feedback
- The amount of disagreement of the feedback
- The source of the feedback
- The complexity of the initial program
- The length of the communication
- Audience for the communication
- The level of detail asked for in the communication
- The format of the communication (PowerPoint, poster, oral, written…)
- The number of participants in the group

- Audience for the communication
- The length of the communication

Potential Observations

Programming Fundamentals

- Accuracy of the description of programming (i.e., Does the student explain the programming process as an iterative process that includes designing, implementing and debugging?)
- Degree to which the specifications meet the intent of the program. (i.e., Are the inputs and outputs identified in the specifications appropriate for the intent of the program?)
- Completeness of the specifications (i.e., Does the specifications include information on the boundary statements? Is the level of details in the specifications enough to start generating pseudo code?)
- Accuracy of the explanation of the concept of a variable (i.e., Does the student identify when a variable could be used? Does the student describe the purpose of a variable? Does the student relate variables to programming?)
- Appropriateness of the comparison (i.e., Does the student include similarities and differences in the algorithms? Does the student include an explanation with their comparison? Does the comparison include information on relevant aspects of both algorithms? Does the comparison accurately reflect the algorithms?)
- The correctness of the algorithm (i.e., Does the algorithm follow the specifications? Does the algorithm produce the correct outputs for any set of inputs?)
- Completeness of the description (i.e., Does the description include information about all of the main parts of the algorithm)

Algorithms in Programming

- The correctness of the algorithm (i.e., Does the algorithm follow the specifications? Does the algorithm work in the boundary cases?)

Programming Structures

- Description of the features of a programming structure (i.e., Given a programming structure can the student describe the purpose of the structure?)
- Appropriateness of the evaluation (i.e., Does the student include an explanation of how the features of the programming structure match the problem or algorithm? Does the explanation accurately reflect the features of the problem or algorithm and the features of the programming structure?)
- Appropriateness of the comparison (i.e., Does the student include an explanation for the benefits and drawbacks of each of the programming structures which accurately reflects the features of the problem or algorithm and the features of the programming structures?)

Programming Process

- Accuracy of the description of debugging and testing methods (i.e., Does the description cover multiple aspects of debugging and testing? Is the description of particular methods correct?)

- Accuracy of the evaluation of debugging and testing methods (i.e., Does the student accurately describe the type of information they will learn from the debugging and testing methods? Do the test cases cover common errors and boundary cases?)
- Correctness of the resulting program (i.e., Does the program run correctly for all cases?)
- Accuracy of the description of which debugging and testing methods were used
- Appropriateness of the test cases (i.e., Do the test cases cover the boundary cases as well as the general cases?)

Evaluating Programs

- Correctness of the output given a set of inputs
- Accuracy of the description of a program (i.e., Does the student correctly identify the inputs and outputs? Does the student explain how the program operates on the inputs?)
- Appropriateness of the evaluation of the extent/degree to which a program solves a stated problem (i.e., Did the student consider the boundary cases as well as the general cases?)
- Appropriateness of the comparison of the tradeoffs (i.e., Does the student describe how each program approaches the problem? Does the student describe cases for which the approaches differ and the costs and/or benefits of these differences?)

Collaborative Programming and Communication

- Appropriateness of the division of workflow (i.e., Did all members have work to do? Was the work that was assigned to each member appropriate and matches to that member's skills?)
- Quality of the produced program (i.e., Does the program include all of the required pieces? Does it run correctly?).
- Appropriateness of the revisions to a program based on provided feedback (i.e., Were all comments addressed in some way? Were disagreements in feedback resolved?)
- Degree of collaboration (i.e., Did each member of the group contribute to the communication?)
- Clarity of the communication (i.e., Was the description understandable? Was the communication organized?)
- Completeness of the communication (e.g., Did the communication address all of the sections of the program? Was the description of the design process comprehensive?)

Potential Work Products

Programming Fundamentals

- Description of programming
- A set of specifications for a program
- The explanation of the concept of a variable
- The comparison of two algorithms
- The description of an algorithm

Algorithms in Programming

- An algorithm

Programming Structures

- The description of a programming structure
- The evaluation of the relationship between features of a programming structure and features of a problem or algorithm
- The comparison of the programming structures

Programming Process

- Description of debugging and testing methods
- The evaluation of the debugging and testing methods
- A (corrected) program
- A description of the debugging and testing methods used
- Generated test cases

Evaluating Programs

- The list of outputs
- A description of a program
- The evaluation of the program
- Comparison of the tradeoffs between different programs

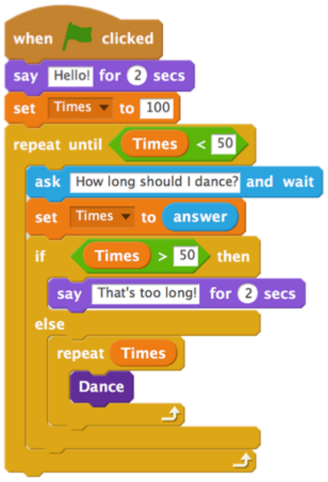
Collaborative Programming and Communication

- The resulting program
- A description or list of the assignment of the workflow
- A description of the comments and changes that were made and/or a description of how the comments were addressed
- A communication (could be power point, could be oral, could be written) about a program
- Documentation of how each member of the group contributed to the communication

Unit 4 Assessment Item Example: Scratch Dancer

- *FKSA 6 - Ability to describe features of a programming structure*
- *FKSA 10 – Ability to describe debugging and testing methods*
- *FKSA 13 - Ability to generate test cases for a program*
- *FKSA 14 - Ability to state what a program would output given a set of inputs*

6. Below is the initial stage and Scratch code for programming a dance performance.



```
when ⚑ clicked
say Hello! for 2 secs
set Times to 100
repeat until Times < 50
    ask How long should I dance? and wait
    set Times to answer
    if Times > 50 then
        say That's too long! for 2 secs
    else
        repeat Times
            Dance
```

The code uses a custom block called Dance (Dance). The Dance block makes the ballerina do a dance. The number of times the ballerina does the dance is based on the value of Times (Times).

The following is what happens when the green flag is clicked:

1. The ballerina says "Hello!"
2. The ballerina asks "How long should I dance?"

a) You enter the number 100 when asked "How long should I dance?" Describe what would

____

____

____

b) You enter the number 50 when asked "How long should I dance?" The following happens:

- The ballerina does the dance.
- Then the ballerina asks "How long should I dance?" again.

Based on the code, explain why the ballerina does the dance instead of saying "That's too long!"

____

____

____

____

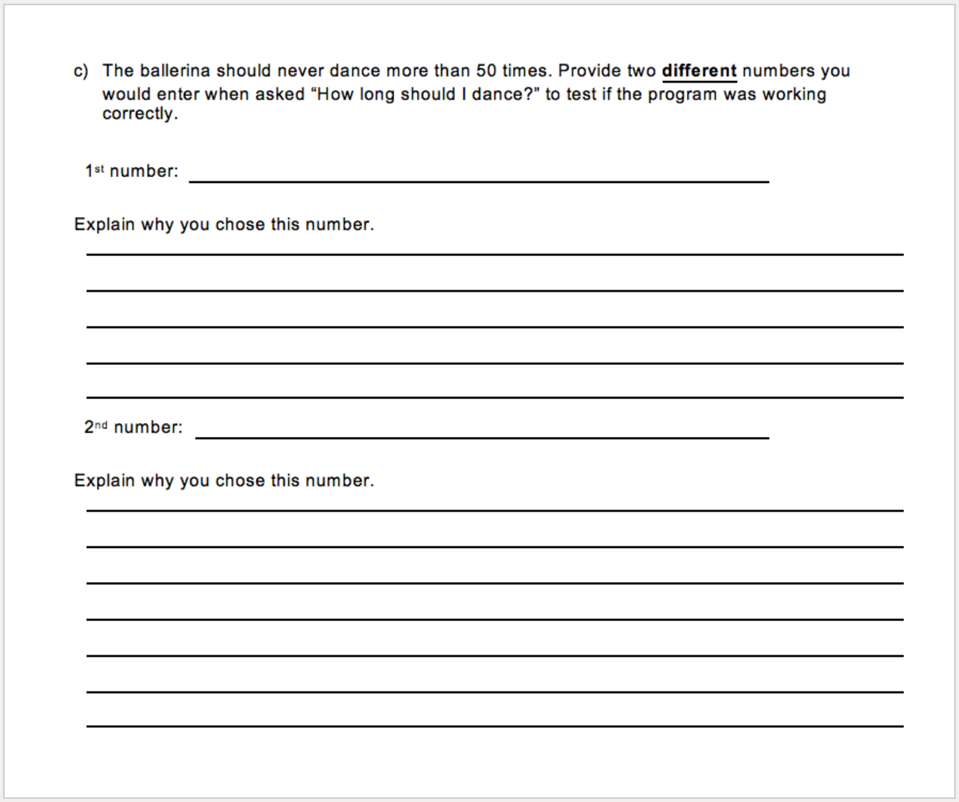
Based on the code, explain why the ballerina asks "How long should I dance?" again instead of stopping.

____

____

Unit 4 Assessment Item Example: Scratch Dancer *(continued)*

- *FKSA 6 - Ability to describe features of a programming structure*
- *FKSA 10 – Ability to describe debugging and testing methods*
- *FKSA 13 - Ability to generate test cases for a program*
- *FKSA 14 - Ability to state what a program would output given a set of inputs*



c) The ballerina should never dance more than 50 times. Provide two **different** numbers you would enter when asked "How long should I dance?" to test if the program was working correctly.

1st number: ______________________

Explain why you chose this number.

____

2nd number: ______________________

Explain why you chose this number.

____

SRI Education™

SRI Education, a division of SRI International, is tackling the most complex issues in education to identify trends, understand outcomes, and guide policy and practice. We work with federal and state agencies, school districts, foundations, nonprofit organizations, and businesses to provide research-based solutions to challenges posed by rapid social, technological and economic change. SRI International is a nonprofit research institute whose innovations have created new industries, extraordinary marketplace value, and lasting benefits to society.

Silicon Valley
(SRI International headquarters)
333 Ravenswood Avenue
Menlo Park, CA 94025
+1.650.859.2000
education@sri.com

Washington, D.C.
1100 Wilson Boulevard, Suite 2800
Arlington, VA 22209
+1.703.524.2053

www.sri.com/education

Stay Connected