COMPUTATIONAL THINKING RUBRIC

for more information, please contact CTAL-info@UDel.edu

This rubric was created and refined by faculty and administrators at the University of Delaware (UD). It is explicitly modeled after the VALUE rubrics developed by the American Association of Colleges and Universities.[1] This rubric was initially developed by faculty in Computer & Information Science and revised through participation in NSF Award 1611959 ("Infusing Computational Thinking into General Education"). It addresses a new General Education objective for all UD undergraduate students. Like the original VALUE rubrics, this rubric "articulates fundamental criteria for each learning outcome with performance descriptors demonstrating progressively more sophisticated levels of attainment." It is intended to guide faculty in developing computational thinking assignments, activities, and objectives in courses in many different disciplines. It is also intended to aid faculty in assessing student learning.

Definition

Computational thinking[2] is a problem solving approach that is systematic and can be automated and used to solve many kinds of problems often (but not always) with the help of a computer. Students employing computational thinking use a set of concepts, such as decomposition, data, algorithms, and abstraction, to process and analyze data, and to create real and virtual artifacts. (Adapted from Barr & Stephenson, 2011. http://csta.acm.org/Curriculum/sub/CurrFiles/BarrStephensonInroadsArticle.pdf)

Framing Language

Computational thinking is regularly employed by scholars, students, and practitioners in many different disciplines. The skills that are used in computational thinking are not unique to this problem-solving approach despite their origins in mathematics and computer science. What sets computational thinking apart from other problem-solving approaches is a focus on automating the approach. Although the cutting edge of computation thinking focuses on quickly solving problems using immense sets of data, the approach is also used in other contexts where efficiency, transparency, and correctness are critical.

The rubric for this learning outcome includes four dimensions: decomposition, algorithms, data, and abstraction. They are listed in order beginning with the least challenging and simplest (decomposition) and ending with the most challenging and complex (abstraction). Although they differ in terms of challenge and complexity, decomposition and abstraction are conceptually related with decomposition a necessary skill for abstraction.

Computational thinking is closely related to quantitative reasoning. Many scholars and practitioners employ computational thinking to create data sets that are well organized and tools that enable analysis of those data sets. That is why the data dimension of this rubric includes criteria such as "can be analyzed to discover meaningful patterns and relationships." In practice, however, creating a new data set that meets all of the capstone criteria is very complex and challenging so some faculty teaching computational thinking may find it more practical to focus on analyzing pre-existing data sets. Moreover, faculty frequently require students to analyze data sets but this goes beyond computational thinking into other skills, primarily quantitative reasoning. However, faculty often find it helpful to have students analyze data as a way to provide concrete context to an abstract and unfamiliar computational thinking skills a practical and meaningful context for students.

A strict adaptation of the scholarship related to algorithm dimension requires the inclusion of "efficiency" as a core concept, particularly at the capstone level of that dimension. However, that concept is particularly challenging and is often a difficult and advanced skill even for upperclass computer science students who explicitly focus on it. Therefore, this concept is simplified in this rubric as it is unrealistic to expect that it is addressed at an advanced level in the vast majority of computational thinking courses and students outside of the specific discipline of computer science. In particular, the idea of efficiency included in this rubric at the capstone level of the algorithm dimension is simplified: it only focuses on simple internal characteristics of the algorithm (e.g., optimal sequencing of steps, steps that address all parts of the problem) without expecting students to make comparisons to other algorithms or perform complex analysis of algorithmic efficiency.

This material is based upon work supported by the National Science Foundation under Grant No. 1611959

[1] See https://www.aacu.org/value/rubrics for more information about the VALUE rubrics.

[2] Although UD's GenEd objectives use the phrase "computational reasoning," throughout the literature this concept is referred to as "computational thinking" so we use that phrase in this document..

COMPUTATIONAL THINKING RUBRIC

for more information, please contact CTAL-info@UDel.edu

Definition

Computational thinking is a problem-solving methodology that can be automated and transferred and applied across subjects and is often implemented with a computer. It is explicitly modeled after the VALUE rubrics developed by the American Association of Colleges and Universities.[3] Students employing computational thinking use a set of concepts, such as abstraction, recursion, and iteration, to process and analyze data, and to create real and virtual artifacts. (Adapted from Barr & Stephenson, 2011. http://csta.acm.org/Curriculum/sub/CurrFiles/BarrStephensonInroadsArticle.pdf)

Evaluators are encouraged to assign a zero to any work sample or collection of work that does not meet benchmark (cell one) level performance.

	Capstone	**Milestones**		**Benchmark**
	4	3	2	1
Decomposition *Breaks a problem into its constituent subproblems*	Breaks a complex problem into clearly described, well-defined, and distinct-but-related subproblems that are easier to solve than the original problem but when combined efficiently solves the original problem.	Breaks a complex problem into clearly described subproblems that are distinct-but-related but lack efficiency, although they solve the original problem.	Breaks a complex problem into subproblems that lack efficiency, fail to have sufficient descriptions, and overlap, although they solve the original problem.	Breaks a complex problem into subproblems that are inefficient, described poorly, overlap or closely related, and fail to completely solve the original problem.
Algorithms *Creates a series of ordered steps to solve a problem or achieve a goal*	Creates a logical, efficient, and well-described sequence of steps or instructions to solve a problem or achieve a goal.	Creates logical sequence of steps that are well-described (e.g., unambiguous, precise) and solve a problem or achieve a goal but the steps are inefficient e.g., not in an optimal sequence, overlapping, duplicative, or unnecessary.	Creates logical sequence of steps that solve a problem or achieve a goal but the steps are poorly described (e.g., ambiguous, vague).	Creates a sequence of steps that do not solve a problem or achieve a goal. The steps lack efficiency, sufficient descriptions, and are not described or documented.
Data *Evaluate[4] a data set to ensure that it facilitates discovery of patterns and relationships*	Evaluates[4] a data set to ensure it is sufficiently comprehensive, efficiently organized, meaningfully labeled, and thoroughly described so that it can be analyzed to discover meaningful patterns and relationships.	Evaluates[4] a data set to ensure it is sufficiently comprehensive, meaningfully labeled, and thoroughly described but fails to ensure that it is efficiently organized.	Evaluates[4] a data set to ensure it is sufficiently comprehensive and meaningfully labeled but fails to ensure that it is thoroughly described and efficiently organized.	Evaluates[4] a data set but fails to ensure that it is sufficiently comprehensive, efficiently organized, meaningfully labelled, and thoroughly described so patterns and relationships are not obscured.
Abstraction *Reduces complexity to create a general representation of a process or group of objects so it is not only appropriate for the immediate purpose or goal but can also be used in different contexts*	Creates an accurate-but-simplified representation of a process or group of objects to solve the problem or meet the goal. Selects essential characteristics by filtering out unnecessary information. Can be used to solve other problems or goals.	Creates an accurate-but-simplified representation of a process or group of objects to solve the problem or meet the goal. Selects essential characteristics by filtering out unnecessary information. Cannot be used to solve other problems or goals.	Creates an accurate-but-simplified representation of a process or group of objects to solve the problem or meet the goal. Fails to select all essential characteristics by filtering out unnecessary information. Cannot be used to solve other problems or goals.	Creates a representation of a process or group of objects that is not accurate, not sufficiently simplified, or fails to solve the problem or meet the goal.

This material is based upon work supported by the National Science Foundation under Grant No. 1611959

[3] See https://www.aacu.org/value/rubrics for more information about the VALUE rubrics.

[4] Advanced courses or disciplines with a strong focus on computational thinking may instead require students to create a data set instead of evaluating an existing one.