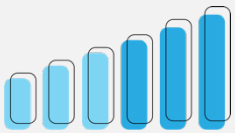
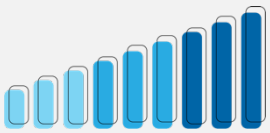
Core Concept: Computational Thinking

www.p12engineering.org

Engineering Literacy Dimension: Engineering Practices
Practice: Quantitative Analysis
Overview: *Computational Thinking* is the process of dissecting complex problems in a manner to generate solutions that are expressed as a series of computational steps in which a computer can perform (Aho, 2012). Typically, this process is separated into four elements: (1) decomposition (the method of dissecting a problem into smaller more manageable tasks), (2) pattern recognition (the method of searching for similarities within problems or solutions), (3) abstraction (the method of synthesizing important information and filtering out irrelevant data while generating a solution), and (4) algorithm design (the method of creating a step-by-step solution to be carried out by a computer program) (BBC, 2018). Computational Thinking also includes knowledge related to (a) *the formation of algorithms (including flowcharting),* (b) *the translation of algorithms using appropriate programming languages,* and (c) *software design, implementation, and testing*. Computational Thinking is important to the practice of Quantitative Analysis as engineering professionals systematically analyze and develop algorithms and programs to develop or optimize solutions to design problems. Furthermore, computational thinking is necessary to develop efficient and automated physical systems as well as visualizations of design concepts and computational scientific models (NRC, 2011).

Performance Goal for High School Learners
I can successfully design, develop, implement, and evaluate algorithms/programs that are used to visualize/control physical systems that address an engineering problem/task.

	 Basic	 Proficient	 Advanced
ALGORITHM FORMATION (including flowcharting)	I can interpret a flowchart of a designed system and describe how the system may work with what algorithms.	I can develop algorithms in order to develop a part of my solution and communicate them using flowcharts.	I can develop and implement a program that incorporates a series of algorithms in order to optimize my solution in its entirely.
PROGRAMMING LANGUAGES	I can develop basic programs using correct syntax and logical organization.	I can develop programs using more advanced programming techniques, such as loops, conditional structures, and variables.	I can develop programs using highly advanced techniques, such as writing external functions and calling them from a program.
SOFTWARE DESIGN, IMPLEMENTATION, & TESTING	I can develop a solution to an engineering design problem using industry-grade software.	I can develop and implement a solution to an engineering design problem using a variety of industry-grade software.	I can evaluate and justify which software package, among a variety of industry-grade software, is optimal for solving a specific engineering design problem.