TECH TIP: Computational Thinking

Computational thinking (CT) at its core is a problem-solving process that can be used by everyone, in a variety of content areas and everyday contexts. Computational thinking is an approach in which you break down problems into distinct parts, look for similarities, identify the relevant information and opportunities for simplification, and create a plan for a solution. This broad problem-solving technique includes four elements: decomposition, pattern recognition, abstraction and algorithms.

HOW TO USE COMPUTATIONAL THINKING

Decomposition	Pattern Recognition
Breaking down problems into smaller sections. • Breaking down problems into smaller parts can make complicated challenges more manageable. This enables other computational thinking elements to be applied more effectively to complex challenges. The solutions to the smaller problems are then combined to solve the original, larger problem. • Real-world Examples: For instance, when you clean your room, you may put together a to-do list. Identifying the individual tasks (making your bed, hanging up your clothes, etc.) allows you to see the smaller steps before you start cleaning.	*Recognizing if there is a pattern and determining the sequence.* • Examining the problem for patterns, or similarities to previously solved problems, can simplify the solution. Pattern recognition can lead to grouping, organizing, or streamlining problems for more efficient outcomes. Conversely, a lack of patterns is also useful because it means there is no more simplification to be done. • Real-world Examples: You have likely used pattern recognition in games like UNO, checkers, mancala and SET. Sports like football and basketball also use pattern recognition to identify the opponent's strategy.
Abstraction	**Algorithms**
Generalization of a problem — focus on the big picture and what's important. • Taking a step back from the specific details of a given problem allows you to create a more generic solution. This requires analyzing the problem to remove extra detail and highlight the basic parts. Once completed, begin brainstorming a solution to the problem. • Real-world Examples: Public transportation maps are examples of abstraction that you may encounter often. The maps show only the important information (the stops, the general direction that you are heading) and leave out the finer details.	*Step by step instructions to solve a problem.* • When solving a problem, it is important to create a plan for your solution. Algorithms are a strategy that can be used to determine the step-by-step instructions on how to solve the problem. Algorithms can be written in plain language, with flowcharts, or pseudocode. • Real-world Examples: We use algorithms daily, normally in the form of step-by-step instructions. Recipes, instructions for making furniture or building blocks sets, plays in sports, and online map directions are all examples of algorithms.

The Bowers Institute

FACILITATIVE TIPS

There are a variety of ways that students can practice and hone their computational thinking, well before they try computer programming.

Integrate computational thinking into other subjects to make it concrete and relevant for students. Find the ways your classroom already practices computational thinking and call it out!

You may naturally find opportunities to explicitly highlight CT elements during activities like:

- Multi-part project assignments (decomposition).
- Recurring sequences, like routines within a school day: circle times, food breaks, small group work, individual reflections (pattern recognition).
- Document analysis to develop a synopsis or summary (abstraction).
- Daily practices, such as classroom procedures to line up or exit the classroom (algorithms).

Focus on one CT element at a time. Finding opportunities to practice each individual element may be easier than developing activities with a combination of skills.

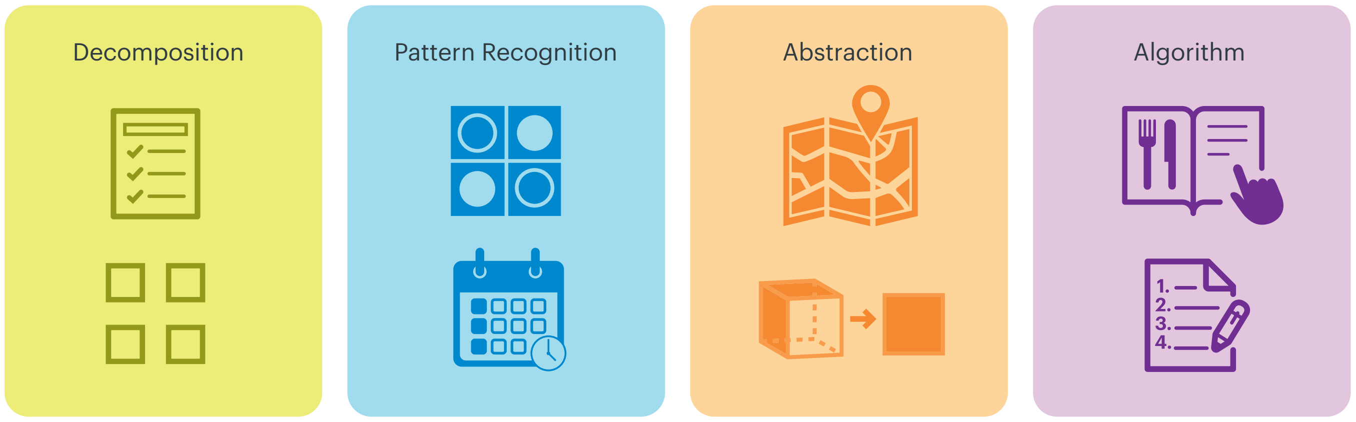
When possible, long-term projects give learners a chance to use all four computational elements. The order of the CT elements will vary depending on the project; however, many projects follow a similar process:

- Break the task into smaller pieces (decomposition)
- Recognize prior knowledge that they can apply to the task (pattern recognition)
- Sift through to find the relevant details (abstraction)
- Create a timeline and plan for execution (algorithms)

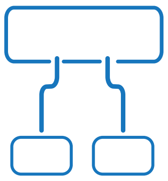
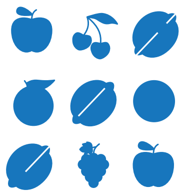
Bridge the connections to computer science by using a combination of “plugged” and “unplugged” activities. While computational thinking is necessary for computer programming, applying these elements doesn’t need to happen on a computer. This varied approach reinforces student confidence with these skills, better preparing them to write a computer program in the future.

COMPUTATIONAL THINKING

Real-world Examples:



FACILITATIVE QUESTIONS

Computational Thinking Element	Facilitative Questions
Decomposition 	• What are the different parts of the problem you are trying to solve? • Describe the main sections/parts of the problem you are solving. • How could this problem be divided into smaller parts?
Pattern recognition 	• Are there any patterns that you observe? • Do you notice any similarities between this problem and something else you've already solved? • Do any of the parts of this problem share qualities? • Does anything repeat?
Abstraction	• What are you trying to solve? • Which details are important in solving this problem? • What can you leave out? What information is unnecessary? • Can you describe this problem as something more basic?
Algorithms 1 2 3	• What's the first step you can take in solving this problem? • What are the steps that you need to do to solve this problem? • In what order should you complete those steps?

TECH TIP: Computational Thinking

EXAMPLES OF COMPUTATIONAL THINKING

You already use computational thinking all the time, both in your classroom and in everyday life. Here are some examples of specific ways to incorporate these different elements into your teaching.

	Decomposition	Pattern Recognition	Algorithms	Abstraction
Math	**Tangrams** are a fun example of decomposition. Ask students to analyze a shape and break it down into geometric parts. (K-5) It helps to break down large number problems into smaller, more digestible parts through strategies like **factoring**. (4-5)	**Sequencing** problems ask students to find similarities between the provided information and then use that information to determine what comes next. (1-3) In geometry, identifying **symmetry** requires pattern recognition. (1-5) Noticing that you can use the two-**variable strategies** when solving an equation with three variables is also an example of pattern recognition. (6-8)	Students use **processes and formulas** that allow them to learn addition, subtraction, multiplication, and division with single digit integers and increase the complexity of the problems over time. (K-5) **PEMDAS** and **FOIL** are two examples of generic algorithms that ensure that problems are solved in the correct order. (6-8)	Solving **word problems** requires students to comb through the information to pull out what's necessary. (3-5) **Data analysis** involves looking at large sample sets and teasing out trends. (9-12)
Science	Using **logical inference**, one can break down a large scientific question into smaller, tractable experiments that, when combined together, yield insight into the larger question. For example, students may infer the ecosystem of a given animal based on its qualities. (K-12)	**Sorting and classification** in biology and astronomy rely on pattern recognition. (6-12) Neurons in the brain do pattern recognition in order to **process data**. (9-12)	The "**methods**" or "**procedure**" of a classroom lab experiment is a set of instructions. (6-12)	**Scientific and mathematical models** distill complex systems into only the pertinent information. (K-12) **Scientific laws and theorems** are succinct statements that describe natural phenomena developed from a wide body of experiments. (K-12)
ELA	When students approach an **essay**, they likely break the essay into the central argument, the introduction, and the conclusion. They write the sections separately and then combine them for a full argument. (3-12)	An understanding of **phonics** helps us pronounce new words correctly. We understand that a letter or group of letters creates a certain sound and then apply that to unusual words. (K-12)	When writing poetry, **traditional poetic forms** can be defined like algorithms, with a sense of meter, rhyme structure, and the order in which the lines need to be written. (1-12)	A **book report** is meant to be an analysis of ideas and themes in literature. A student writing a book report will tease out the relevant information and omit small details. The report should revolve around defending their central thesis and not retell the whole story. (4-8)
Social Sciences	**Causal inference**, interpreting the elements that precipitated historical events, is a way of looking at history and breaking down the societal and cultural factors that influenced it. (9-12) Our federal government is broken into **three major branches**, each of which has different sets of responsibilities and powers, creating a system of checks and balances. (4-12).	Throughout history, **empires** across continents have followed similar trajectories: rise, development of a culture, a period of peace, and then a fall. (6-10)	Formalized processes in the **legislative system** can be viewed as algorithms. For instance, the process of creating a law can be broken down into: Drafting the bill, debate, voting in Congress, and attaining presidential approval through signature. If the president vetoes the bill, then we need a supermajority of the Senate to vote to override the veto. (3-12)	A compelling **analysis of historical events** focuses not on the minute details but rather the thematic trends and sociological forces of the period. (6-12) **Maps** are often examples of abstraction — you seek different types of maps depending on the level and category of detail needed. (3-8)