SAE STEM@Home: Thinking Computationally

EDUCATOR GUIDE

Resources from A World In Motion®
What Is Programming Each Other?

In the Programming Each Other challenge, children learn about the thought processes involved in computational thinking, which include breaking down problems and converting them into logical steps. Throughout the challenge, they investigate real-world problems that can be broken down into smaller subproblems. They explore complex programming concepts as they try to systematize activities that they do on a daily basis.

In this challenge, children also explore programming. An exciting and unique feature of the challenge is that it is “unplugged.” Many young people spend a large amount of time looking at screens (both in and out of school). The Programming Each Other Challenge allows students to explore the thought processes of programming while being active, whether they are working to learn a new card trick or learning new dance steps.

Throughout this module, students explore some of the basic challenges inherent to programming. They discuss and practice increasingly complex tasks with each successive activity. Each new level of difficulty is designed to introduce specific programming concepts, including:

• Loops (repetition of sections of code)
• Conditional statements (determination of which action is performed in a given scenario)
• Variables (temporary place holder for data that may take on more than one meaning in a program)
• Error handling (steps to take if a program encounters unexpected data)
STEM CONNECTION
What Do You Need to Know to Be a Good Programmer?

Computational Thinking
Computational thinking involves solving problems, designing systems, and understanding human behavior by drawing on the concepts fundamental to computer science. Not everyone agrees on an exact definition of computational thinking, but many people believe it includes these four major elements: problem decomposition, pattern recognition, abstraction, and algorithms.

Problem Decomposition
Decomposition refers to breaking down a complex problem into smaller, more manageable parts. For example, the food waste in a compost pile decomposes. In solving problems, decomposition involves breaking a problem down into parts that make it easier to solve.

Computer programmers use decomposition in the same way that you might when deciding how to solve a problem. A programmer considers what the program needs to do and then breaks that down into smaller chunks. If a programmer was working to create a new video game, she would first need to decide what the world of the game was like, what types of characters the game would include, and the plot of the game. From there, she could break down each of the parts even further.

Decomposition can save time for a programmer, too. It is much easier for them to find mistakes in small amounts of code than in longer code. Small chunks of code can also be reused more easily. For example, code that creates a walking motion might be reused for multiple characters, even though those characters’ arms may move differently.

Decomposition can help us think about problems logically as well.

Pattern Recognition
The word “pattern” means a regularity in the world. Recognizing patterns allows us to predict what might happen in the future. Learning to read and recognizing people that we know involve pattern recognition. Patterns can also be found in nature.

In computer programming, pattern recognition is extremely important. Identifying patterns allows programmers to develop short segments of code that can easily be repeated. For example, a video game programmer may need to create a program that allows a character to walk across the screen. She could create a program that gives the same set of instructions many times or she could develop code that can be repeated as desired.

One of the most famous codes in history is the Enigma code that was used by Germany in World War II. The Germans used a special machine to translate messages into code. The way that messages were encoded was changed every night and there were 158 quadrillion (158,000,000,000,000) ways to set up the machine. The British worked very hard to crack the Enigma code and they used pattern recognition to do it. The British noticed patterns in the messages that were sent by the Germans, including the fact that messages often began...
with the same word and ended with the phrase “Heil Hitler”. These patterns helped the British crack the Enigma code in only 5 and a half months. Alan Turing, a mathematician from the University of Cambridge, was one of the most important code breakers.

Abstraction
Abstraction is a way to filter out some details to make a problem easier to solve. It is important to figure out what details are not as important so that you can make the problem simpler. When you look for patterns, you might begin to notice that some details are important and others less important. It is important for children to learn what information is relevant and what information doesn’t really make a difference so that they can correctly and efficiently solve problems.

Writing Algorithms
An algorithm is a set of rules that precisely defines a sequence of steps for completing a task. Some people call algorithms programs. In fact, the biggest single difference between the word algorithm and the word program is that a program is most often written in a particular programming language and an algorithm is a detailed set of instructions for performing a particular task. Good programmers are able to clearly lay out the steps required to perform a task before they even begin to write code. Attention to detail and the ability to think through algorithms before writing computer code can save a programmer a lot of time and worry when creating their programs.

Programming
Programming is the act of writing a series of instructions that control the operation of a computer or other machine. Simple programs can look similar to instructions for building furniture, walking to school, or baking a cake. Programs can consist of lots of parts pieced together to solve fairly complex problems.

Loops
One of the first techniques budding programmers learn is the loop. Loops are key to writing easily read and understood code (the written text of a program), and they give programmers an efficient way to describe repeated steps.

A goal for any programmer is to write code that is efficient—it can perform the task correctly and as quickly as possible while using the fewest resources (which, for a programmer, would mean processing power and data space). Using loops in programs may seem to be more efficient for the computer, but it really is not: A computer does not care if it is given an instruction 100 times or given an instruction once and asked to repeat it 100 times—in the end, it does the same thing, whether the steps are all written out or put in a loop.

However, loops do provide efficiency for the programmer. Because humans write (and review and revise) code, it is easier to do so when the code is as short as possible. Programming with loops makes code easier for humans to read, interpret, and debug, and also helps avoid errors (e.g., mistyping a line or miscounting how many times a certain step is included). Some techniques that programmers use to shorten their code will not necessarily result in more efficiency when the program is run, but they make the code easier to read and fix.
Conditionals

A conditional is a programming statement that determines which action is performed based on a condition. For example, if it is raining, then I will put on my raincoat. Typically, an if-then statement is found at the end of a loop, with a condition that says to go back to a particular step in the program and then continue from there if a condition is or is not met.

In programming, several specialized looping concepts have evolved. There are three types of specialized conditional statements, do-until, while-do, and for-next.

Do-until creates a loop that continues until a certain condition is met. For example, if you were washing dishes:

```
DO {
    PICK UP A DISH
    SCRUB WITH SPONGE
    PUT DISH IN DRYING RACK
} UNTIL THE SINK IS EMPTY
```

A While-Do loop, on the other hand, has the condition at the beginning, so it is checked even before the steps in the loop are run. This is better to use if it is possible that the condition might not be met even the first time the loop is encountered in the program.

```
WHILE THE SINK STILL HAS DISHES IN IT, DO {
    PICK UP A DISH
    SCRUB WITH SPONGE
    PUT DISH IN DRYING RACK
}
```

Programmers use Do-Until and While-Do loops when they don’t know ahead of time how many times the loop needs to be performed, but they can state a particular condition that makes the loop stop.

A third common type of loop is the For (or For-Next) loop. In these loops, a counter is set that specifies exactly how many times the loop will run. In the example below, the program will run four times.

```
FOR INDEX = 1 TO 4 STEP 1
    PICK UP A DISH
    SCRUB WITH SPONGE
    PUT DISH IN DRYING RACK
NEXT INDEX
```
ACTIVITIES
1. Breaking Down a Problem
In this activity, children explore how problems are decomposed or broken down into smaller problems to make them easier to solve.

Getting Ready to Go to Sleep
Materials
• Interactive activity

Designing a Solution
Materials
• Paper
• Pencil
• Eggs (optional)
• Cardboard (optional)
• Variety of materials (optional)

2. Recognizing Patterns
In this activity, children explore how recognizing patterns can help them to play games and solve problems.

Guess Who?
Materials
• Sheet of Guess Who? game characters from Hasbro
• Kooky creatures activity sheet
• Paper
• Pencil

Using Patterns to Break Codes
Materials
• Build a Cipher wheel activity sheet
• Paper fastener or pushpin
• Pencil
3. Abstraction
In this activity, children practice the skill of abstraction—removing unnecessary details to make solving a problem easier.

Funny Fill In The Blank
Materials
• Funny Fill In The Blank activity sheet
• Pencil

4. Writing Algorithms
In this activity, children develop their own algorithms (detailed instructions) for their own secret handshake and for navigating a maze.

My Secret Handshake
Materials
• Paper
• Pencil
• Navigating a Maze

5. Programming Unplugged
In this activity, children learn about common programming commands and explore how they work.

Looping
Materials
• Looping Dance Moves activity sheet

Game Conditionals
Materials
• Deck of cards
• Dice
• Pencil
• Paper
SAMPLE SOLUTIONS
1. Breaking Down a Problem

Think About It!

1. Now that you’ve thought about it, were you already decomposing problems to help you solve them?
   *Children’s answers to this question will vary. If possible, suggest some ways in which you use problem decomposition and try to draw out ways that children might. You could suggest that you use decomposition when planning a party or a meal or preparing for a task that you do at work.*

2. How do you think problem decomposition might help a computer programmer?
   *Programmers need to break down large, complex problems into smaller subunits so that the task of programming becomes less complex. Computer programs often have many different elements to them, so programmers create solutions to each of those elements and then combine to create the final program.*

2. Recognizing Patterns

Think About It!

1. Based on the patterns that you found in the Kooky creatures, what do you think are the best first questions to ask?
   *Based on the creature characteristics, it is probably best to ask, “Does your creature have 2 eyes?” There are 9 creatures with 2 eyes and 15 with 1, 3, or 4 eyes, so this characteristic eliminates as close as possible to half of the creatures in the first guess. (Another question that would eliminate nearly half is “Does your creature have an even number of eyes?”)*

2. Have you ever used a dichotomous key to identify an organism? (If you haven’t, find a leaf outside and use the Arbor Day Foundation’s Identification Field Guide to identify it.) These tools help you to identify plants and animals by answering a series of questions. How do you think that identifying patterns helps people develop dichotomous keys?
   *By identifying patterns in leaves, the people who created this field guide could develop a tree that would allow users to identify trees based on their leaves. These patterns allow the scientists to develop a decision tree that eliminates various species as each characteristic is explored. Field guides are very similar to the game of Guess Who?*
3. Abstraction

Think About It!

1. How could you use your abstraction skills to make your Fill in The Blank Story funnier?
   You could abstract the words to ensure that words that are most appropriate (and funny) are chosen. For example, simply choosing a noun might not be nearly as funny as choosing a “piece of furniture” or a “part of your face” or “an animal you might see on safari”.

2. What do you think the most important features of a dog are? How can you be sure that someone will know that you are talking about a dog and not a cat?
   *Children might say things like a dog’s ears, tail, or snout are important. Challenge them to think about what makes a dog a dog. You might say something like, “If I told you the animal I was thinking of has a snout, pointy ears, and a tail, what animal do you think I might be talking about?” Then, ask them to think of the characteristics that are clearly related to dogs.*

4. Writing Algorithms

Think About It!

1. What is your algorithm for tying your shoes? Do you think everyone uses the same algorithm? Watch the video below to see some different shoe-tying algorithms!
   *Children will have lots of different shoe-tying algorithms. They should understand that different people use different algorithms to accomplish the same tasks.*

2. What other types of algorithms can you think of? When do you use these algorithms? How do they help?
   *A recipe is a type of algorithm, as are the instructions for games or for building toys, such as Lego.*

5. Programming Unplugged

Think About It!

1. Have you ever thought about how machines know what to do? Imagine that you need to program a robot to navigate any maze. What kinds of commands might you use?
   *You might need the robot to turn left and right, move forward, to stop, and to recognize when a wall is in front of it.*

2. If you could design an app, what would you want that app to do? How could you start to create your app?
   *Children will have various ideas about apps they might like to create. If they don’t, explore the app store on a phone or other digital device to see the types of apps available and then talk about what might be fun or interesting to create.*