THINKERSMITH™

Traveling Circuits

Lesson 3

MY ROBOTIC FRIENDS

a one hour
introductory activity
adapted for CSED Week 2013

COMPUTER SCIENCE
Education Week
WWW.CSEDWEEK.ORG

Disclaimer
Neither Thinkersmith nor any other party involved in the creation of this curriculum can be held responsible for damage, mishap, or injury incurred because of this lesson. Adult supervision and supervised caution is recommended at all times. When necessary, every effort has been made to locate copyright and permission information.

Lesson 3: My Robotic Friends

Main Goal: Highlight programming techniques and illustrate the need for functions.

Overview: Using a predefined "Robot Vocabulary" your students will figure out how to guide one another to accomplish specific tasks without discussing them first. This segment teaches students the connection between symbols and actions, as well as the valuable skill of ***debugging***.

If time allows, there is an option to introduce ***functions*** at the end of the lesson.

Objectives:

Students will

- Learn to convert real-world activities into instructions
- Gain practice ***coding*** instructions with symbols
- Gain understanding of the need for precision in coding
- Gain practice debugging malfunctioning code
- Understand the usefulness of functions and ***parameters*** (grades 7+)

Materials and Preparation:

Estimated lesson time: 1 hour
Estimated prep time: 10 min

Materials

- Symbol Key (1 per group)
- Cup Stack Pack (1 per group)
- Disposable Cups or Paper Trapezoids (6 or more per group)
- Blank paper or note cards (1 per person)
- Writing Instrument (1 per person)

Preparation

- Print out one Symbol Key for each group
- Print a Cup Stack Pack for each group
- Cut trapezoids from Paper Trapezoid template if not using cups
- Stack cups or trapezoids in designated area away from groups (Robot Library)

Key Lesson Vocabulary:

Algorithm - A series of instructions on how to accomplish a task

Coding - Transforming actions into a symbolic language

Debugging - Finding and fixing issues in code

Function - A piece of code that can be called over and over

Parameters - Extra bits of information that you can pass into a function to customize it

Lesson Plan

Introduce: Start by asking the class if anyone has heard of robotics. Has anyone seen a robot or touched one? Does a robot really "hear" you speak? Does it really "understand" what you say? The answer to the last question is:
"Not the same way that a person does."

Robots operate off of "instructions", specific sets of things that they have been preprogrammed to do. In order to accomplish a task, a robot needs to have a series of instructions (sometimes called an algorithm) that it can run. Today, we are going to learn what it takes to make that happen.

Kickstart: Pull out a copy of the Symbol Key (or write the symbols on the board). Step to the side and tell the class that these will be the only six symbols that they will be using for this exercise. For this task, they will instruct their "robot" to build a specific cup stack using only these arrows:

- ↑ - Pick Up Cup
- ↓ - Put Down Cup
- → - Move 1/2 Cup Width Forward
- ← - Move 1/2 Cup Width Backward
- ↻ - Turn Cup Right 90°
- ↺ - Turn Cup Left 90°

Adjustments:

Grades K-3

- Try this lesson all together as one class. Let the students shout directions for the teacher to write down.
- Have a class assistant leave the room during programming, then return to perform the finished code.
- If there is time, switch. Have the assistant write the instructions from the class and have the teacher perform them.

Grades 4-6

- Adjust group sizes between three and five, depending on personality of class.
- Expect each student to want a turn, this will likely use the entire hour.

Grades 7+

- Limit groups to four students, three is ideal.
- Students generally complete the full round of turns with plenty of time to include the supplement section on functions.

Steps:

1. Choose one "Robot" per team.
2. Send robot to "Robot Library" while the "programmers" code.
3. Choose one image from the Cup Stack Pack for each group.
4. Groups will create an algorithm for how the robot should build the selected stack.
5. Coders will translate their algorithm to arrows, as described in Symbol Key.
6. When programmers have finished coding their stack they can retrieve their robot.
7. Upon return, the robot reads the symbols from the cards and translates them back in to movements.
8. The group should watch for incorrect movements, then work together to debug their program before asking the robot to re-run it.

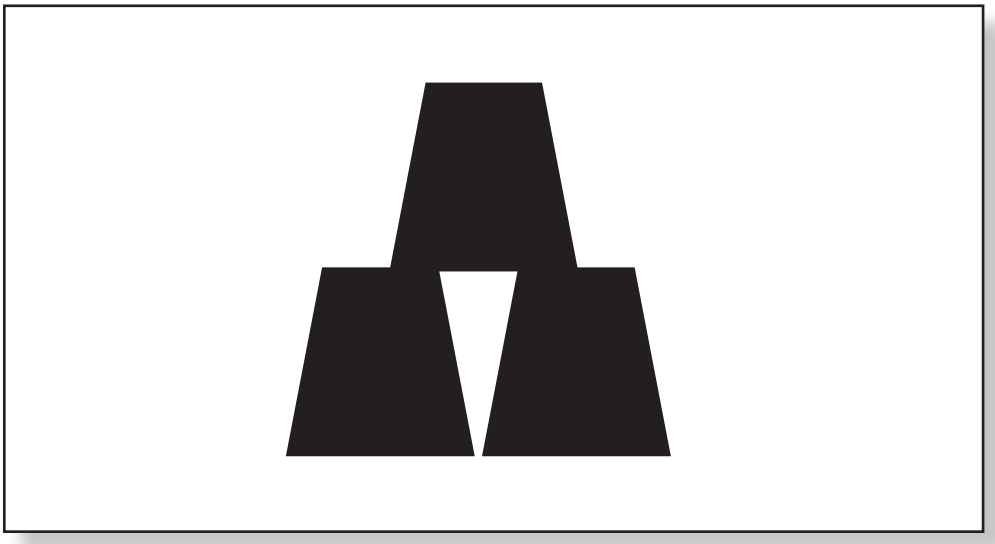
Rules:

1. Coders should translate all moves using *only* the six arrows suggested.
2. Cups should remain with the robot, not provided to programmers during coding.
3. Once robots are back with their groups, there should be no talking out loud.

If your student asks about rules that haven't been defined above, you can either define them according to your exercise, or ask them to define that rule within their own group.

Example

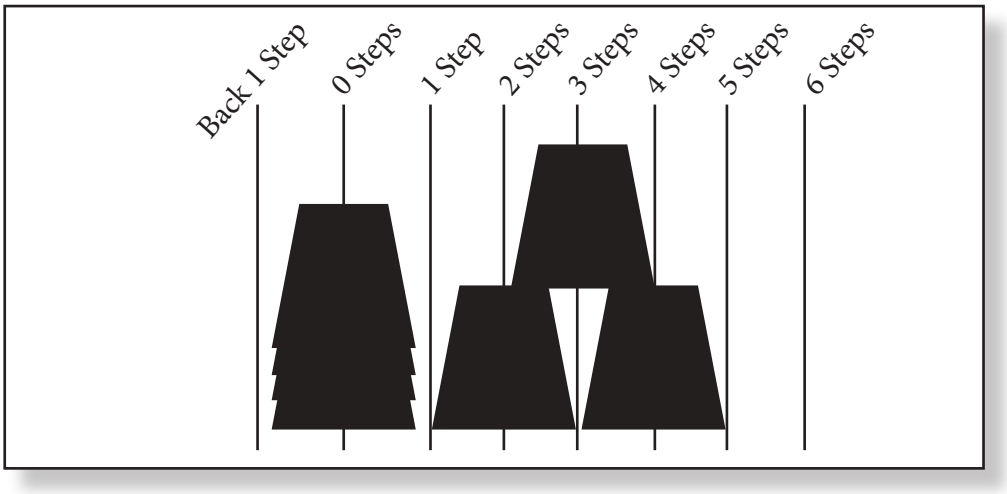
Beginning: It can be helpful to go over an example as a class. There is one cup stack in the pack that includes only three cups, that is the sample card. Hold it up for the class and walk them through the exercise.



3 Cup Stack from Cup Stack Pack

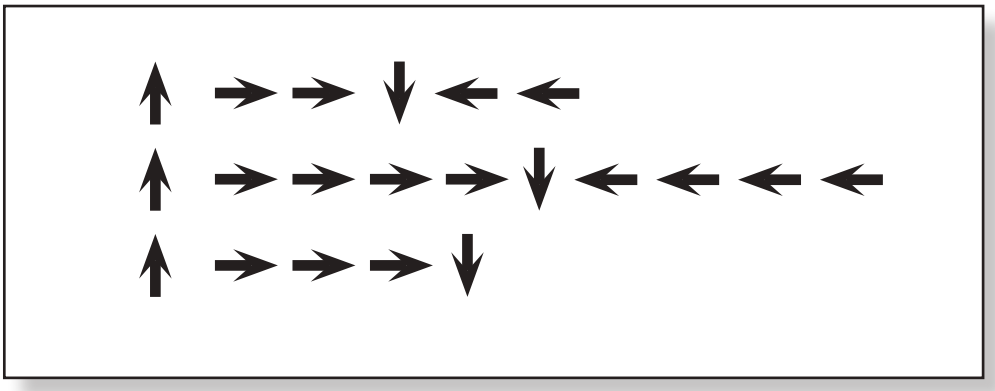
Place your stack of cups on the table where everyone can see them. Ask the class to instruct you on the first thing to do. The correct answer is "pick up cup". When you pick up each cup, note that the cup should automatically rise above the highest cup already in the stack.

With your hand still in the air, ask for the next move. You may have to remind the class a time or two that one step forward is only half the width of a cup.



Step Guide

Middle: Once you've placed a single cup, transition back to the blackboard (or document camera) and challenge the class to help you write the symbols on the board so that you can "run the program" later. One possible solution looks like this:



One Solution for 3 Cup Stack

Completion: With the program written down for the class to see, you can call a volunteer to "run" it, or you can run the program yourself. Say the arrows out loud as you move the cups into place. For example, the program above would be pronounced:

"Pick up cup", "Step forward", "Step forward", "Put down cup"
"Step backward", "Step backward"

"Pick up cup", "Step forward", "Step forward", "Step forward",
"Step forward", "Put down cup", "Step backward", "Step backward",
"Step backward", "Step backward"

"Pick up cup", "Step forward", "Step forward", "Step forward",
"Put down cup"

The Exercise

Group Up: Group the students appropriately for their age as described on page 2. The goal is to have enough programmers in each group that the group is never entirely lost.

Robot: Choose one "robot" in each group to go hang out in the "robot library". This should be a location far enough away from the groups that no robot can find out what Cup Stack Card their programmers are working with. Robots can use their time in the library to practice cup stacking and to ask for clarification on rules.

Program: Each group of programmers should be handed one Cup Stack Card at a time. They can then begin to figure out the algorithm for their stack. How many cups will they need? How many steps for the first cup? The second? Are any cups upside down? How do you get the robot to flip a cup?

Once these questions are answered, the programmers can use the symbols to write their code on the blank paper or a note card. The programmers should review their code to see if it makes sense for the stack before checking their robot out of the robot library.

Run Code: Now that the robot is back with the group, everyone should be silent. The groups should not attempt to use words or gestures to influence their robot's behavior. The robot should only operate according to what the arrows tell them to do.

If the group finds a mistake, they are allowed to halt the program, check the robot back into the library, and fix the error before bringing the robot back to complete the challenge.

Repeat: Each time a group solves a challenge, they should choose a new robot to head to the library, and the group should be given a new (preferably more difficult Cup Stack Card.)

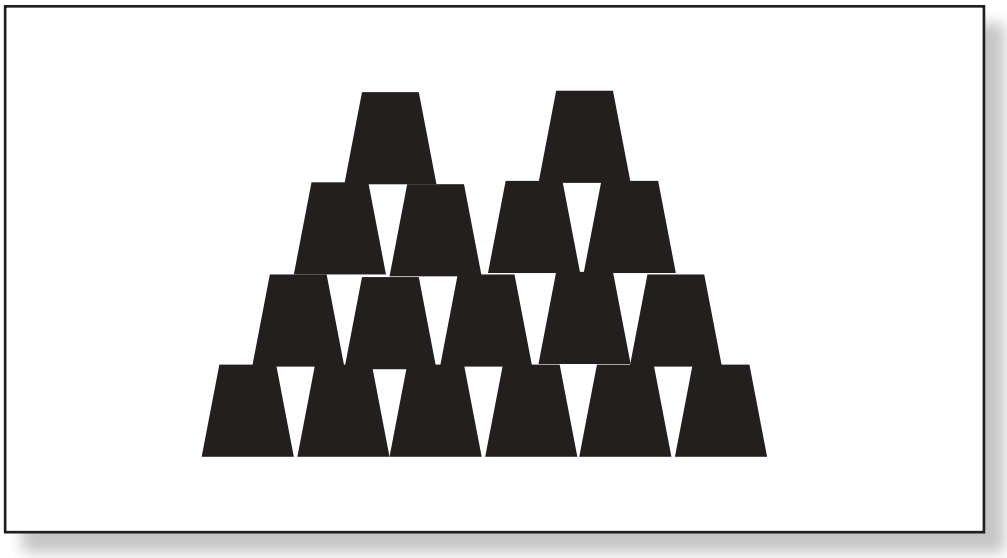
This can continue either until time is done, all group members have been robots, or the cards have become difficult enough to warrant a discussion about functions.

Tip: If the lesson is still going strong, but the groups begin to run out of Cup Stack Cards, challenge them to create their own stack drawings.

Function Supplement

Opportunity: If you find that you have extra time, this can be a great opportunity to introduce the usefulness of functions!

Introduce: Gather the class back together. Tell them that you are going to give them a special stack that you want them to code in record time...then show them this:



17 Cup Stack

You will almost certainly hear moans, giggles, or even refusals. Ask the class what the problem is. What makes this stack so difficult?

This stack is daunting, because every additional cup added to the width adds two more arrows to the line of code for steps forward, and two for steps backward. To properly code this, you would inevitably get a line of symbols that looks like this:

Many times, students begin to write a shortened version of the instructions during the main game that includes numbers. For example:

During the main game, that method should be noticed, and discouraged. Remind them to instead stick only to the six symbols they are allowed. In this portion, however, recognize the brilliance and foresight of those who tried the trick, and acknowledge that they independently discovered the need for functions!

Explain: Point out to the class that the arrow with numbers is a clever way of indicating that we want to repeat the arrow a specific number of times. By allowing that, we are essentially creating a new symbol that we can call in order to avoid repeating code unnecessarily. This is exactly the idea behind functions.

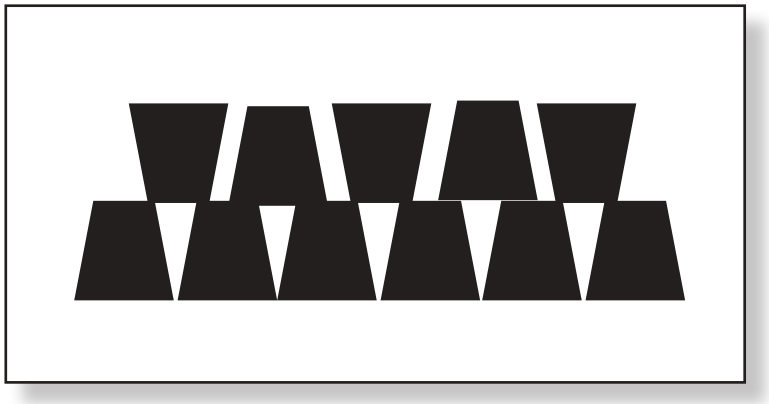
Challenge the class to find the biggest bundles of repeating code for each cup placement. As the instructor, you can settle on any grouping that makes sense, but the series might look something like what we discussed earlier:

That's a good intermediate step, but let's simplify it even further. If we position the symbols in a clockwise manner (starting at the top) and morph them into one structure, we might end up with something like:

where "x" is the number of steps you will need to move forward, then backward. That "x" becomes the parameter to our function.

Parameters: Above, we have an arrow symbol that somewhat resembles the rest of our vocabulary, but we've also included a way to "pass" information about how many times we want to apply the forward and backward arrows. In the computer science world, that extra passed information is called a parameter. Parameters can further customize an already helpful function.

Remix: Now that the class has this new symbology, let them tackle one of the more intense Cup Stack Cards. Groups may combine if they need more cups to work with.



17 Cup Stack

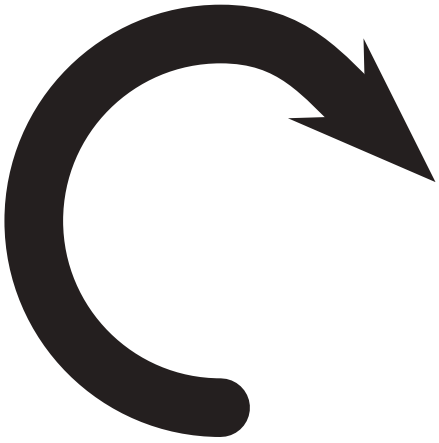
Symbol Key

Pick Up Cup

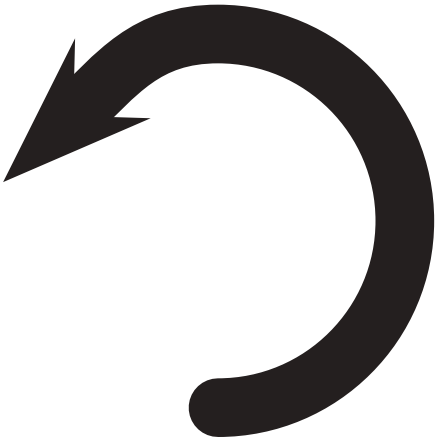
Put Down Cup

Step Forward

Step Backward



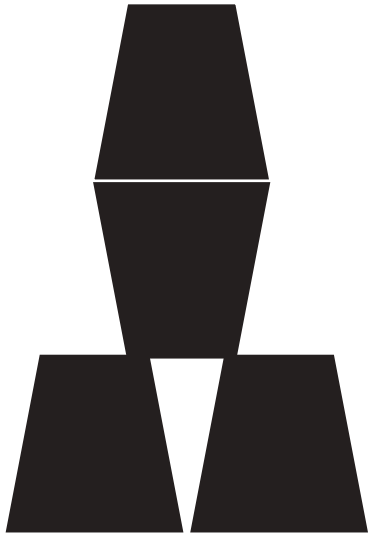
Turn Cup Right 90°

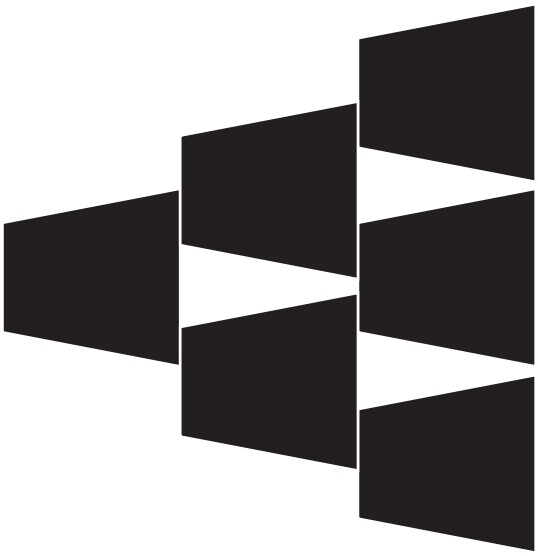


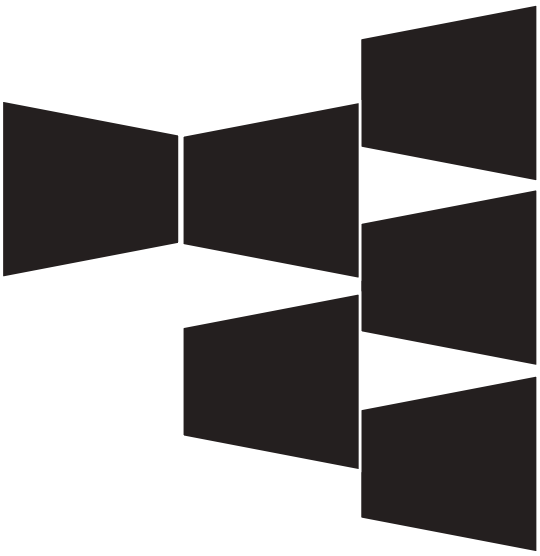
Turn Cup Left 90°

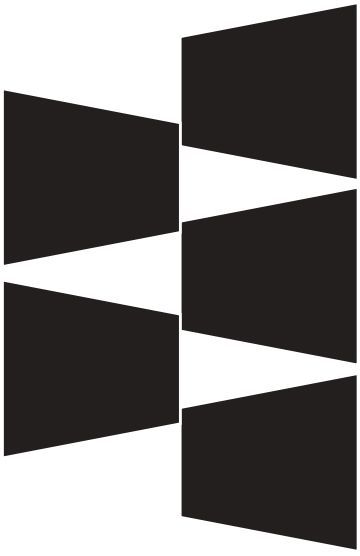
Cup Stack Pack™

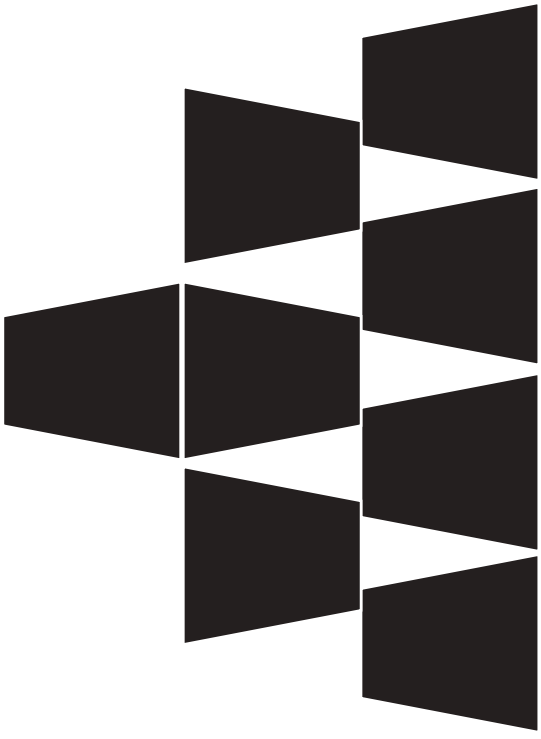
THINKERSMITH™

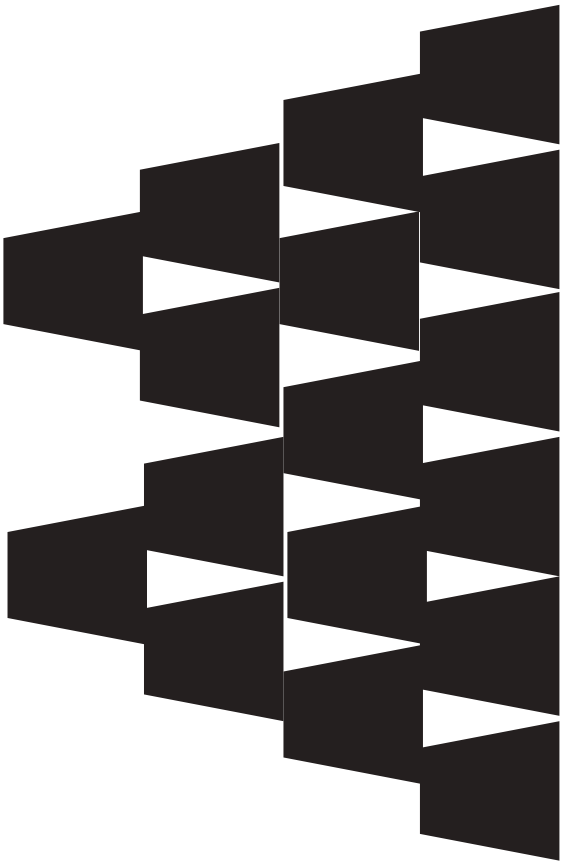


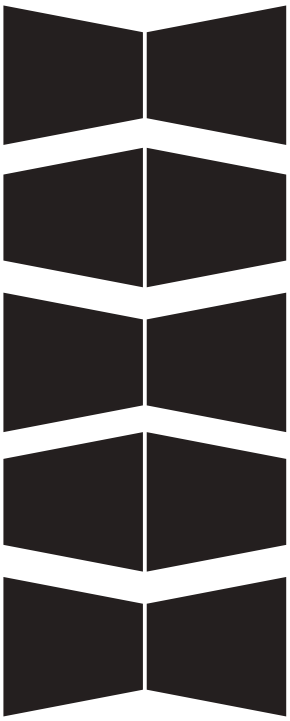




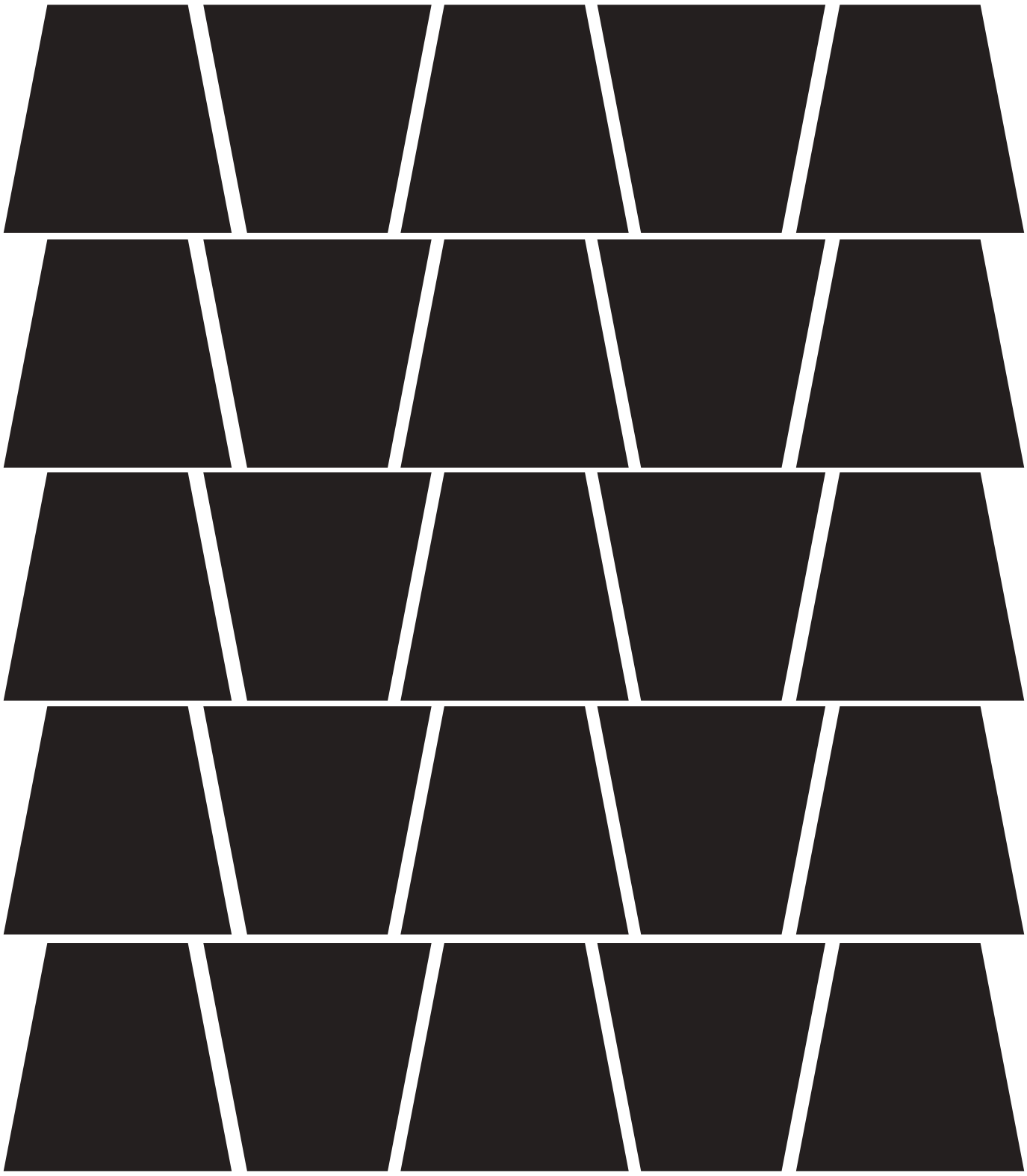








Paper Trapezoids



To cut quickly:

First cut in horizontal strips, then snip along lines to make trapezoids.

COMPUTER SCIENCE
Education Week
WWW.CSEDWEEK.ORG

For more lessons, please visit
www.thinkersmith.org