Coach's Secret Code

Basketball is a fast-paced game, but successful plays don't happen by accident. Coaches design strategies, communicate instructions, and rely on players to carry out those instructions in the correct order. Computers work in a similar way! They follow instructions called algorithms to complete tasks. Instructions must be clear, organized, and precise. In this activity, students will discover how coding and computational thinking can help a basketball team turn a coach's strategy into a successful play.

Here's what you'll need:

- **Blank paper**
- **Pencil or writing utensil**
- **Projector/Display**
- ***Coach a Play!* and *Create Your Own Play!* activity sheets** (print double-sided, one per student)
- ***What's Wrong With the Play? Side A and Side B* activity sheets** (print double-sided, one per student)
- ***Optional*: tokens to represent basketball players on the grids**
- ***Optional*: painter's tape or sticky notes to create a larger grid**



Warm Up: *Think Like a Computer*

Computers are everywhere! We use them to play games, watch videos, communicate with friends, and follow our favorite basketball teams. But how does a computer know what to do? Unlike humans, computers don't understand just any language or any instruction. Instead, they follow very specific instructions that tell them what actions to perform.

In this Warm-Up, students will explore the importance of clear directions and the order in which those directions are given. Students will be introduced to the concept of an algorithm, which is a set of step-by-step instructions for accomplishing a task. These instructions must be specific enough to produce the intended result.

Teacher Note: Avoid revealing the connection to computational thinking before students complete both drawing challenges. Allow them to discover how the activity relates to computers and algorithms during the discussion afterward!

PART A: Draw a Basketball

Begin by providing students with a blank sheet of paper and a pencil or other drawing utensil. Tell students, *"I'm going to give you a task. Your job is to follow the instructions exactly as given."*

Then, give students just one task: *"Draw a basketball."*

Some students may ask for clarification or more specific instructions. **Encourage them to do their best using only the information provided.** The goal is to see how students interpret the same instruction in different ways.

After a few minutes, ask students to hold up their papers or share their drawings with a partner. Even though students received the same task, their drawings will likely look different! Some might have drawn a large basketball, while others drew a small one. Some may have included the lines on a basketball, while others drew a simple circle. Students might have even drawn an entire court.

Ask students:

- *"Why do our drawings look so different if all of you were given the same task?"*
  - Students might answer that they imagined basketball differently, added details, or maybe don't even know a basketball well enough to draw it.
  - Guide students to recognize that the instruction didn't specify how to draw the basketball - so there was room for everyone to make their own decisions.

PART B: Follow the Instructions

Ask students to turn their papers over, or use another blank sheet. Explain that this time, instead of completing a task based on one instruction, they will follow a series of instructions.

Remind students to follow each instruction exactly as it is given, without adding anything extra. Students should also lift their pencils between each instruction, then wait for the next one before continuing.

Read the following instructions aloud, pausing after each step:

1. *Draw a large circle in the CENTER of your paper.*
2. *Starting at the TOP of the circle, draw a STRAIGHT, vertical line down through the middle to the bottom of the circle.*
3. *Starting at the LEFT EDGE of the circle, draw a STRAIGHT, horizontal line across the middle to the right edge.*
4. *Starting at the UPPER RIGHT EDGE of the circle, draw a CURVED line that bends inward toward the center, crosses the horizontal line, and curves back outward to the lower right edge.*
5. *Starting at the UPPER LEFT EDGE of the circle, draw a CURVED line that bends inward toward the center, crosses the horizontal line, and curves back outward to the lower left edge.*

After students finish, ask them to compare their new drawings and their original drawings. Then, let them discuss with nearby classmates. Encourage them to notice similarities and differences without making any changes.

Students' drawings should resemble the example below, although some variation is expected:



Ask students:

- *"What was different about the instructions I gave you the second time?"*
  - Students may notice that instead of receiving one general instruction, they received several smaller, more specific instructions.
  - Students might also recognize that the instructions were given in a particular order.
  - Encourage students to identify which instructions could be made even more specific. Some students might have drawn a larger circle or curved their lines differently.

Explain that students just followed an **algorithm!** In computer science, an algorithm is a set of step-by-step instructions for completing a task. Algorithms help us break complicated tasks into smaller, more manageable steps.

Humans are good at using their experiences and knowledge to interpret instructions. When students heard "draw a basketball," they likely already knew what a basketball looked like, and could decide how to represent it. However, computers don't automatically have that same understanding.

Whether a computer is playing a video game, calculating math, or accomplishing other tasks, it relies on instructions written in a language that it can interpret and execute. That's why clear, precise instructions are so important in computer science!

Ask students:

- *"Can you think of something you do every day that involves following a series of steps?"*
  - Answers may include brushing their teeth, making a sandwich, getting ready for school, or following a recipe.
  - Help students recognize that algorithms aren't limited to computers. We follow step-by-step instructions all the time.
- *"Why does the order of instructions matter? What might happen if we changed the order?"*
  - Students may recognize that certain steps must happen before others for a task to be completed successfully. For example, drawing the lines of a basketball before drawing the circle could make it difficult to position them correctly. Explain that in computer science, the order in which instructions are carried out is called **sequencing.**
  - Students may recognize that changing the order of instructions can change the outcome of a task. For example, a basketball player cannot shoot a ball they haven't received yet! Explain that algorithms use instructions in a logical sequence, which means arranging the steps in an order that makes sense. Changing that order can produce unexpected results.
- *"Have you ever followed directions that were confusing or incomplete?"*
  - Students may share experiences with recipes, games, school assignments, or building something. Encourage students to consider what made the directions confusing and how they could have been improved. Connect their responses to the importance of precision in computer science, where clear and complete instructions help produce the intended result.

Again, emphasize to students that computers use algorithms to accomplish tasks. Those algorithms need to contain clear instructions in a logical sequence.

Transition to "Game Time"

Now, imagine that you're part of the Oklahoma City Thunder coaching staff. Your coach has developed a special play, but instead of writing out every instruction, they've created a secret code using symbols! Can you figure out what each symbol means and follow the instructions to execute the play?



Game Time: *Crack the Playbook!*

During a basketball game, players must be able to understand and follow their coach's instructions, even when the arena is loud and the game is moving quickly. Coaches often use diagrams, symbols, and signals to communicate plays. But what if the Thunder coaching staff created a secret code that only their players could understand?

In this activity, students will explore how symbols can represent instructions, why those symbols must have clearly defined meanings, and how a sequence of commands can be used to complete a task. Students will then use a simple coding language to decode a basketball play and guide a Thunder player across the court.

PART A: Learn the Code

Explain that the Thunder coaching staff has developed a secret playbook. Instead of writing out every instruction, the coaches use *symbols* to represent different actions. Before the players can execute a play, they need to understand what each symbol means.

Introduce the following four symbols, without explaining what they mean:



Explain that each basketball play has a designated **shooting position**, which is the square on the court where the player is supposed to take a shot. It can be identified by *lightly shading the square* we want the player to end up at.

The goal is to follow the commands to reach that square, and then use the ☆ command to actually shoot the ball. *For this part of the activity, the direction the player is facing does not matter when shooting.* A play is successful when the player reaches the designated shooting position and executes the ☆ command.

Explain that computers also need a shared system for interpreting instructions. Programmers use **programming languages** which have defined rules for expressing instructions that computers can execute. Just as people communicate using different languages, programmers can use different programming languages, such as Python, JavaScript, or Scratch. These languages may look different, but they can often be used to accomplish the same tasks.

Teacher Note: The symbols used in this activity are a simplified, made-up coding language designed for the lesson. Real programming languages have their own rules for writing and interpreting instructions.

Ask students:

- *"Why might a coach use symbols instead of writing out every instruction?"*
  - Students may suggest that symbols are faster to recognize, take up less space, or make it easier to communicate during a game. Explain that computers also use systems of symbols and commands to represent instructions. The important thing is that each command has a specific meaning that can be understood consistently.
- *"Would these symbols work if every player thought they meant something different?"*
  - Students should recognize that players need to agree on the meaning of each symbol. Similarly, programming languages use defined rules so instructions can be interpreted correctly.

Now, display the following symbols and what they mean. Each of these symbols is a **command:**



move forward | turn right | turn left | shoot the ball

Explain that each symbol represents a **command**, or an instruction to perform a specific action. When commands are placed in a particular order, they can tell a computer (or, in this case, a basketball player!) how to complete a task.

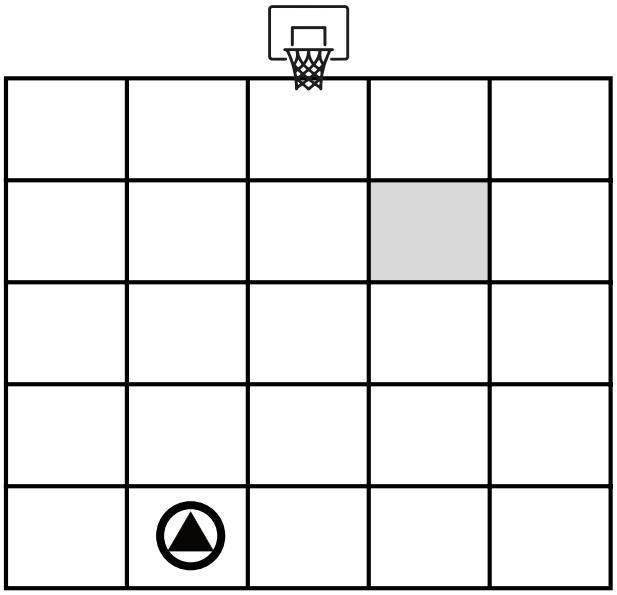
Remind students of the basketball drawing activity from the Warm-Up. They learned that an **algorithm** is a set of step-by-step instructions for accomplishing a task.

Ask students:

- *"What is the difference between an algorithm and code?"*
  - An algorithm describes the steps needed to accomplish a task. Code expresses those instructions using the programming language being used.
  - The same algorithm can be expressed using different programming languages. What matters is consistently using the "rules" of whichever language is being used. For example, the same set of instructions can be communicated in English or Spanish without changing the steps themselves.

PART B: Coach a Play

Guide students through a practice challenge as a whole class or in small groups. You can display the *Coach a Play* activity sheet and distribute a copy to each student. The activity sheet includes a basketball court grid and a "basketball play" (algorithm). Explain that the player must follow commands to reach the shooting position and attempt a shot.



Basketball Play:

↑ ↑ ↷ ↑ ↑ ↶ ↑ ☆

Teacher Note: Important! The player's starting position AND the direction they are facing must be established before following any instructions:

- The turning symbols (↷ and ↶) change the direction the player faces - but will not move the player to another square.
- The ↑ command always moves the player one square forward in the direction they're facing, not necessarily toward the top of the page.

Looking at the grid, a basketball coach might create an algorithm that says: "*move forward two spaces, turn right, move forward two spaces, turn left, move forward one space, shoot the ball.*"

Using the Thunder's secret code, those same instructions could be written as:

↑ ↑ ↷ ↑ ↑ ↶ ↑ ☆

By using the command symbols, emphasize that the algorithm hasn't changed. The only thing that did change was the way it is represented! You can also explain that different programming languages can express the same algorithm in different ways.

Have students work with a partner to decode the new basketball play. Encourage students to predict where the player will end up before attempting to follow the commands. Also remind them that turning changes the direction the player faces but does not move the player to a new square.

After students have tested the algorithm, ask them to compare their results with another pair. Ask, *"Did everyone end up in the same position?"* If not, encourage students to retrace the commands together and identify where their paths began to differ.

Ask students:

- *"Why do we need to know which direction the player is facing?"*
  - Students should recognize that moving forward depends on the direction the player is facing. Turning changes the direction of future movements, even though the player remains in the same square.
- *"What would happen if we switched the order of two commands?"*
  - Students may recognize that the player could end up in a different position or fail to reach the shooting square. Reinforce that sequencing is important because the order of commands affects the outcome.
- *"Are we following an algorithm, or are we using code?"*
  - Students may initially choose one or the other. Explain that they're doing both! The algorithm is the sequence of steps needed to complete the basketball play, and the code is how those steps are represented - we are using the Thunder's symbols.

PART C: Create Your Own Play!

Now it's time for students to become a coach! Have students flip over their activity sheet to the *Create Your Own Play!* side, where they'll find a new, blank basketball court grid. Challenge students to create their own basketball play by writing a sequence of commands using the Thunder's secret code, and then trade with a partner.

Before exchanging papers, make sure both partners know the player's starting position and the direction they're facing - or you can establish this as a whole class. However, emphasize that each student can determine where they want their shooting position to be - but they need to keep it a secret! Their goal is to guide their "player" to a shooting position using only the commands they've learned.

Once students have written their code, have them exchange papers with a partner. Each partner should follow the commands exactly as written, even if they think the instructions are incorrect or won't lead to the intended position. Remind students that computers follow the instructions they're given, not necessarily what the programmer intended them to do!

Once both partners have tested their plays, have students reveal their intended shooting positions and compare them with where their players actually ended up. Did the code successfully guide the player to the correct square to execute the ☆ command?

Ask students:

- *"After creating and coding your own basketball play, did you need to change any of your instructions after your partner tested it? Why or why not?"*
  - Students may have needed to add, remove, or rearrange commands to achieve their goal.

Ask students (cont.):

- *"Let's pretend you want to go to the same shooting position on your grid as someone else does. Could you each write a different algorithm and still end up in the same spot?"*
  - Students should recognize that different algorithms can accomplish the same goal! Two players - or two coaches - might take different paths or follow different sequences of commands but still reach the same shooting position.
  - Explain that in computer science, there can be multiple algorithms for solving the same problem. Some algorithms may be more efficient than others, requiring fewer steps, less time, or fewer resources.
- *"Was there an action you wanted your player to perform that wasn't included in our secret code? What command would you add?"*
  - Students may suggest adding commands for passing, dribbling, jumping, moving backward, or moving more than one space at a time. Encourage students to consider how they would represent their new command with a symbol, and how they would explain its meaning so another player could interpret it correctly.
  - Students may recognize that the available commands limited the types of basketball plays they could create. Explain that programming languages provide different instructions and features, and programmers must work within the rules of the language they're using.

Transition to "Analyze the Replay"

After students have decoded their plays, explain that even well-planned basketball plays don't always work exactly as expected, and the same thing can happen when writing computer programs!

In the next activity, students will become Thunder assistant coaches, reviewing a play that didn't go as planned. Their challenge will be to find the mistake, correct the instructions, and help the team successfully execute the play.



Analyze the Replay: *What's Wrong With the Play?*

Even the best basketball teams sometimes make mistakes. A player might move in the wrong direction, miss a pass, or misunderstand a signal. That's why coaches review game footage to figure out what went wrong and how to improve the team's performance.

Computer programmers do something similar! When a program doesn't work as expected, programmers examine their code to identify and correct errors.

In this activity, students will become Thunder assistant coaches tasked with reviewing a basketball play that didn't go as planned. Using their knowledge of algorithms and coding, they'll investigate what went wrong and determine how to fix it.

PART A: Find the Bug

Explain that the Thunder coaching staff has designed a basketball play, but something isn't working correctly. The players are following the instructions exactly as written, yet they aren't reaching the intended shooting position!

Display the *What's Wrong With the Play? (Side A)* activity sheet, and distribute one to each student. Side A includes a starting position, facing direction, shooting position, and basketball play in code. Have students examine the code and predict where the player will end up.

Then, as a whole group, guide students through the commands one at a time and trace the player's movement across the court.

When the player fails to reach the intended position, **ask**:

- *"Did the player make a mistake, or did the instructions contain a mistake?"*
  - Students may initially suggest that the player moved incorrectly. Remind them that the player followed each command exactly as written. Help students recognize that the error was in the instructions (you could also use the words "algorithm," "code," or "commands") rather than how they were executed.
- Then ask, *"So, if the player followed the instructions correctly, why didn't the play work?"*
  - Students should recognize that an algorithm can be followed correctly but still produce an unintended result if the instructions contain an error. Reinforce that computers execute the commands they're given, even when those commands don't accomplish the programmer's goal.

Explain that an error in a computer program is often called a **bug**. Finding and correcting these errors is called **debugging**.

PART B: Fix the Play

Ask students to flip over to Side B of the *What's Wrong With the Play?* activity sheet. They can work in pairs or groups.

Challenge students to identify what is missing or incorrect in the code. Then, have them revise the code so the player reaches the intended shooting position. Have students test their revised code using the basketball court grid, reminding them to begin at the original starting position and facing direction each time they test their instructions.

After students have tested their corrections, bring the class back together to compare solutions. Invite a few students to share what they changed or added, why they changed it, and whether their revised code successfully guided the player to the shooting position.

Explain that **debugging doesn't end when we find a possible mistake!** Programmers must test their revised code to determine whether their changes actually solved the problem.

Ask students:

- *"How did you figure out what was wrong or missing in the code?"*
  - Students may explain that they traced the player's path, compared it with the intended shooting position, or tested different commands. Explain that programmers use similar strategies to locate errors in their code.
- *"How did your edit change the outcome of the play?"*
  - Students may recognize that changing or adding a single command can change the outcome of an entire sequence. Reinforce that even small errors in code can affect how a program works.
- *"Why is it important to test the code again after making a correction?"*
  - Students should recognize that changing an instruction doesn't guarantee the problem has been solved. Programmers test their code after making changes to confirm that it produces the intended result.



Overtime: *Sideline Signals*

During a basketball game, the arena can get incredibly loud! With thousands of fans cheering, players may not always be able to hear their coach's instructions. That's why coaches use diagrams, symbols, hand signals, and other nonverbal cues to communicate plays with their teams. Now it's the students' turn to create a secret basketball code of their own!

Challenge students to develop a new **coding language** using hand signals instead of written symbols. Explain that, just like a programming language, their new system will need clearly defined commands and consistent rules so that others can understand and follow their instructions.

Have students work with a partner or small group to create a hand signal for each of the four commands they learned during Game Time: *move forward, turn left, turn right, and shoot the ball.*

Students may also invent additional commands if they'd like to expand their language, or as a possible extension. If they do, students will create a new written symbol for each additional command and record its meaning. Remind students that they must agree on what each signal means before they begin.

Once students have established their signals, challenge one student to silently communicate a short basketball play using only their new hand signals. Another student must interpret the signals and translate the play into written code, using the Thunder's original symbols and any additional symbols their group created. Have students record each command in the order it was communicated.

Have students compare the intended play with the translated code. Did the commands match? If not, encourage students to identify which signals caused confusion and discuss how they could make their coding language clearer.

Optional Hands-On Challenge: Bring the basketball plays to life! Using painter's tape, floor tiles, or sticky notes, create a simple grid where each square represents one movement. Have one student communicate the play using hand signals while another follows the instructions by moving across the grid. Remind students that turning changes the direction they're facing but does not move them to a new square.

Additional Challenge: Have students exchange their hand-signal systems with another pair. Can the new pair understand the signals without being taught what they mean? What information would they need to interpret the code correctly?

Ask students:

- *"Why was it important for everyone to agree on the meaning of each hand signal before beginning?"*
  - Students should recognize that everyone using the hand signals needs to interpret them consistently. Just like programming languages, a communication system needs clearly defined rules so that instructions can be understood and followed correctly.*"Did changing from written symbols to hand signals change the algorithm itself?"*
  - Students should recognize that the algorithm can remain the same even when the instructions are represented differently. The sequence of actions hasn't changed; only the way those instructions are communicated.
- *"What happened when a hand signal was unclear or misunderstood?"*
  - Students may recognize that the wrong command was recorded or the intended play was changed. Reinforce that precise communication is important when giving instructions, whether using code, symbols, or gestures.



Coach's Corner: *Computer Science Fundamentals*

Students don't need a computer to learn about computer science! Throughout this lesson, students explore foundational concepts through basketball-inspired challenges, symbolic commands, and hands-on activities.

The following definitions can help you reinforce these concepts throughout the lesson:

- **Algorithm**: A set of step-by-step instructions for accomplishing a task. Algorithms can be written in everyday language, represented by symbols, or expressed using a programming language.
- **Sequencing**: The order in which instructions are carried out. Changing the order of commands can change the outcome of an algorithm.
- **Command**: An individual instruction that tells a computer or person to perform a specific action.
- **Code**: Instructions expressed using a particular coding or programming language. In this lesson, students use a simplified symbolic code to represent basketball plays.
- **Programming Language**: A system of defined rules and instructions used to write programs that computers can execute. Examples include Python, JavaScript, and Scratch.
- **Debugging**: The process of identifying, investigating, and correcting errors in a program or algorithm.
- **Efficiency**: How effectively an algorithm accomplishes a task using resources such as time, steps, or memory. In this lesson, students can compare the number of commands or movements required to reach the same shooting position.

There are plenty of common misconceptions when it comes to computer science:

- *"There's only one correct algorithm for a problem."*
  - Different algorithms can accomplish the same goal! Students may discover several routes to the same shooting position. Encourage them to compare their approaches and consider which might be more efficient.
- *"If the computer follows the instructions correctly, the program will work."*
  - Computers execute instructions according to their programming, but those instructions may contain errors or produce unintended results. That's why programmers test and debug their code.
- *"Turning left or right means moving to a different square."*
  - In the Thunder's secret code, turning changes the direction the player faces but does not move the player to another square. Reinforce this distinction throughout the activities.
- *"An algorithm and code are the same thing."*
  - An algorithm describes the steps for accomplishing a task, while code represents those instructions using a particular language. The same algorithm can be expressed in different ways without changing the underlying steps.
- *"Programming always requires a computer."*
  - Computer science involves problem-solving, logic, communication, and designing instructions. Students can explore these concepts through unplugged activities, like the ones in this lesson.

Tips for Success:

- **Start with the goal.** Before students follow or create a coded play, make sure they know the player's starting position, facing direction, and intended shooting position.
- **Encourage students to think aloud.** Ask them to explain what each command means and predict what will happen before executing it.

- **Celebrate different solutions.** When students develop different algorithms that reach the same destination, use the opportunity to discuss how multiple approaches can solve the same problem.
- **Make mistakes part of learning.** Debugging is a normal and essential part of programming. Encourage students to treat unexpected results as opportunities to investigate and improve their code.
- **Keep the rules consistent.** Students may invent new symbols or hand signals during Overtime, but each command must have a clearly defined meaning shared by everyone using that code.

Programming isn't just about writing code. It's about developing clear instructions, testing ideas, identifying mistakes, and improving solutions. Whether students are drawing a basketball, decoding a Thunder play, or inventing their own sideline signals, they're practicing the same logical thinking and problem-solving skills that computer programmers use every day!

Standards & Skills Alignment

This activity can be used to support students with developing the following skills and/or content.

Oklahoma Academic Standards for Computer Science			
Content/Skills	**Grade 4**	**Grade 5**	**Grade 6**
Algorithms (AP.A) • Develop and follow algorithms consisting of sequenced instructions. • Model, compare, and refine multiple algorithms that accomplish the same task. • Evaluate algorithms for efficiency and effectiveness.	X	X	X
Program Development (AP.PD) • Test solutions to determine whether they produce the intended results. • Identify and correct errors through debugging. • Revise and retest solutions to improve their accuracy.	X	X	X
Program Development (AP.PD) • Collaborate with others to develop and refine solutions. • Communicate and explain decisions made during the development process. • Incorporate feedback to improve solutions.	X	X	X

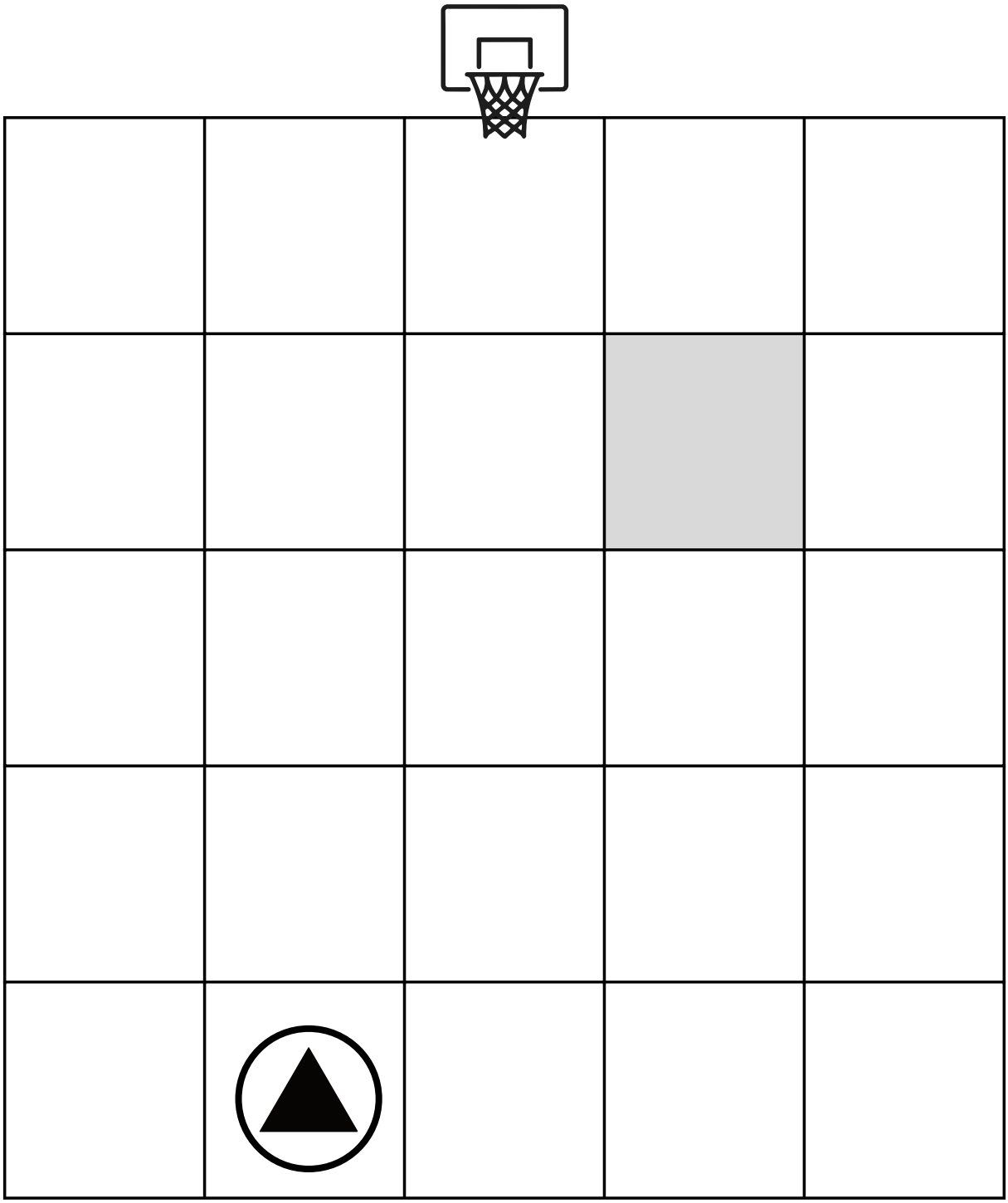
Standards & Skills Alignment (cont.)

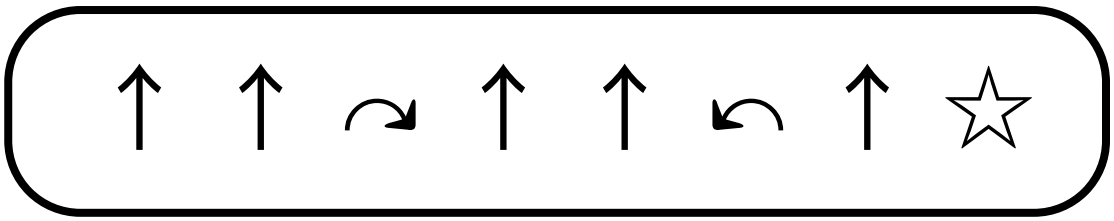
This activity can be used to support students with developing the following skills and/or content.

Oklahoma Academic Standards for Science			
Content/Skills	**Grade 4**	**Grade 5**	**Grade 6**
Information Transfer and Patterns (PS4.C) • Develop and compare solutions that use patterns to transfer information. • Recognize that information can be communicated through different systems of symbols and signals.	X		
Designing Solutions • Develop and compare possible solutions to a problem. • Test and refine solutions based on their effectiveness.	X	X	X
Patterns • Identify and use patterns to organize and communicate information. • Recognize patterns to make predictions and solve problems.	X	X	X
Cause and Effect • Identify relationships between actions and outcomes. • Use cause-and-effect relationships to predict and explain results.	X	X	X

Coach a Play

Name: _____





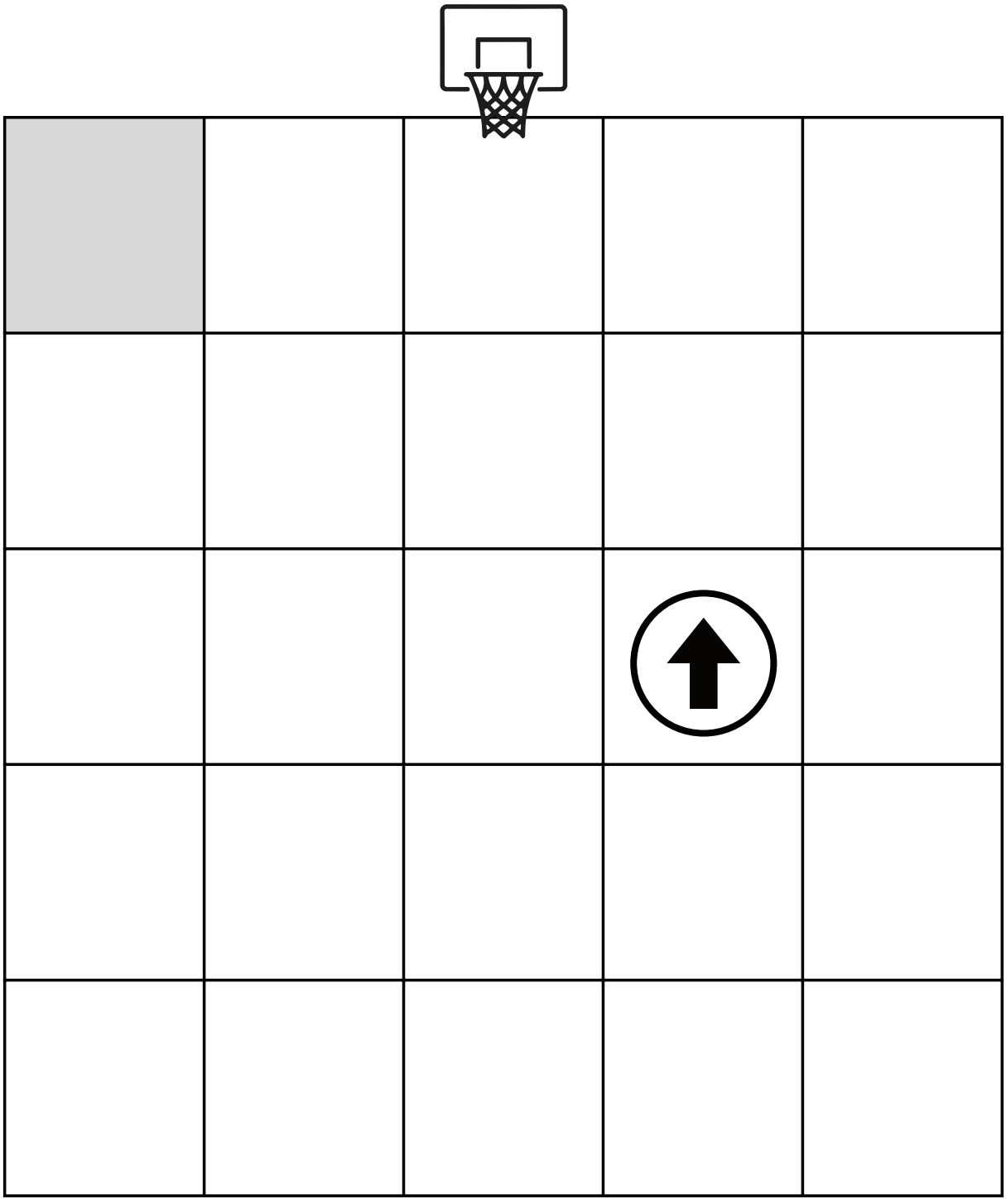
Create Your Own Play!

Name: _____

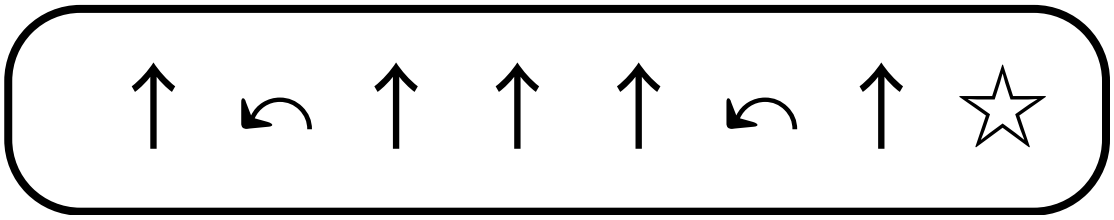


What's Wrong With the Play? (Side A)

Name: _____

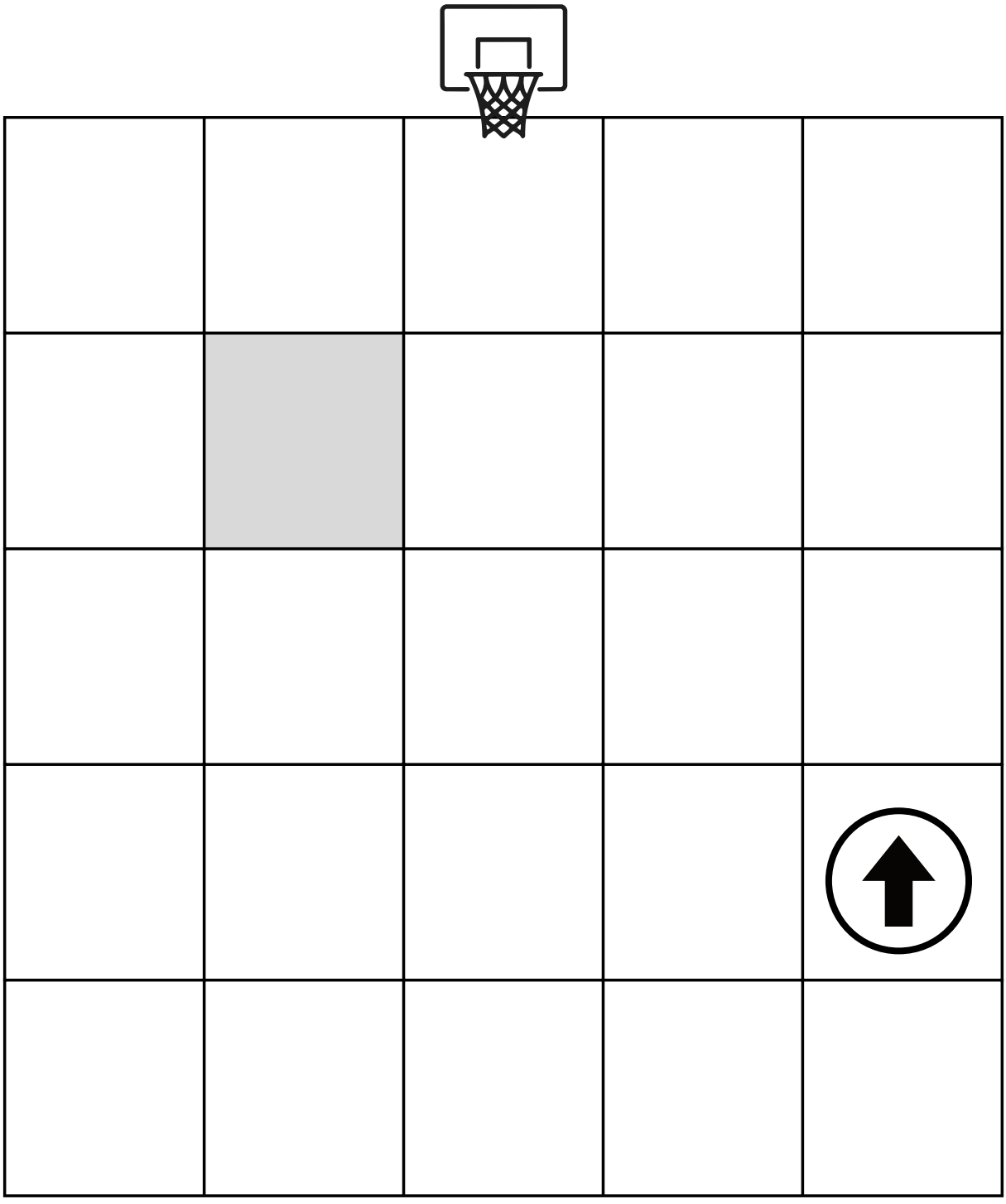


Help **debug** the code below to get the player to the shooting position:



What's Wrong With the Play? (Side B)

Name: _____



Help **debug** the code below to get the player to the shooting position:

