Algorithms are Everywhere!

Unit Overview

Students will gain a basic understanding of what algorithms are, in both their everyday lives and computer science, and learn how to write simple algorithms. The unit includes interactive lesson plans that teach students conditionals and how to write algorithms to give to their peers to “run” and “debug.”

The unit culminates in students utilizing their knowledge to write the general algorithm for solving any maze. The unit does **not** require computers, but teachers seeking a computer component can teach Lesson 6 as a supplement to the rest of the lesson plans.

Lesson 1: Instruct Teacher to Tie Shoe - Students will be able to describe what an algorithm is and create a simple algorithm for tying shoe laces. (AP-01, AP-06, AP-10)

Lesson 2: If/Then Statements (Conditionals) - Students will be able to create if/then statements for situations from their own life. (AP-05, AP-06)

Lesson 3: Intro to Mazes - Students will be able to create their own algorithm for solving a basic maze. (AP-01, AP-06, AP-10)

Lessons 4/5: Theseus Myth/Algorithm to Solve Any Maze - Students will connect algorithms to literature by reading the Theseus Myth and then working together to construct an algorithm that can solve any maze. (AP-01, AP-06, AP-10, AP-11)

Note: Lessons 4/5 are effectively one lesson, but since it is longer, we describe it as being 2 lesson plans.

Lesson 6: Blockly Mazes - Students will be able to apply their knowledge of using algorithms to solve mazes on a computer. (AP-03, AP-04, AP-05, AP-06, AP-10)

Instruct Teacher to Tie Shoe

Algorithms are Everywhere: Lesson 1

Lesson Overview

Introduction
10 minutes

Shoe Tying Activity
15 minutes

Reflection
5 minutes

Standards

- AP-01 Compare two algorithms
- AP-06 Break down problems
- AP-10 Identify and debug errors

Materials

- 1 Shoe with laces

Introduction

The goal of this lesson is to introduce the topic of algorithms. This will be a new word for most students, so take a few seconds to say the word clearly and have the class repeat it back to you. Then, ask the students what they think an algorithm is. After hearing a few guesses, you can explain that an algorithm is **a set of steps that you use to complete a task**.

In order to help them internalize this definition, ask them to come up with a quick example: If they wanted to brush their teeth, what would be the steps needed? Call on students to give the steps they would use. Their answer should be something along the lines of:

1. Get toothbrush and toothpaste
2. Put toothpaste on toothbrush
3. Brush each tooth front, back, and top
4. Spit out toothpaste
5. Wash off toothbrush

After you get a few answers, commend the students for participating, and then ask them to think about how they would construct the algorithm for a more complicated problem. The task for today is to write an algorithm for tying your shoelaces, so this is a great way to get them thinking ahead before you start the next activity.

Shoe Tying Activity

To introduce the idea of an algorithm, students will collaborate to vocally instruct a teacher how to tie their shoelaces.

All you will need for this activity is a shoe with laces. Explain to the students that they must provide very specific **step by step** instructions to help you tie your shoe completely.

Ask for students to volunteer to give the first step in the instructions. You, the teacher, should begin tying the shoe, but follow the instructions given by the students **exactly**. Follow the students' "algorithm" (instructions) until you either tie the shoe or make a mistake. If the algorithm is incorrect (it likely will be) ask the class to fix it (this is the equivalent of debugging in computer science) by starting the process from the beginning and giving more specific instructions. The teacher should emphasize that specificity and order matters by leading students to explain the process of tying a shoe in words so exact that they could not be misinterpreted.

Reflection

Allow five minutes for the students to reflect on what they learned through the lesson's activities. Questions might include:

- In your own words, what is an algorithm?
- Why is working as a team on an algorithm helpful?

If/Then Statements

Algorithms are Everywhere: Lesson 2

Lesson Overview

Introduction

5 minutes

If/then Daily Examples

10 minutes

If/then Physical Activity

10 minutes

Reflection

5 minutes

Standards

- AP-05 Practice implementing conditionals
- AP-06 Break down problems

Materials

- 1 A die or coin

Introduction

To start the lesson, you will test out some if then statements with the students. Ask the students to listen carefully and follow the directions as you read out the following statements:

1. If your name starts with the letter M, then stand up. (Have students then sit down)
2. If you have a pet at home, then stand up. (Have students then sit down)
3. If you wear glasses, then stand up. (Have students then sit down)

After the students are seated, explain that we use if then statements to tell the computer to do something - but only in specific situations. Just as they only stood up IF they met the condition you gave, if / then statements are only followed by the computer IF the condition given is satisfied.

What is an If/then?

Give a concrete example such as “if it is raining outside, I will use my umbrella. If it is not raining, I will not use my umbrella.” Emphasize that you’re checking some condition (in this case, whether or not it is raining) and taking an action based on that (to use an umbrella or not). Examples of if/then statements can be really simple (if I am tired, then I will go to sleep, if I am hungry, then I will eat, etc.). Have students try and give examples of if/then decisions they make in their daily lives. If they are struggling, then give them some examples of if statements and have them complete the then portion (if you are bored at home, what will you do?). Try and have most students give an example, and it’s okay if you need to help them come up with one.

A good example to show how if/then statements are used in computer science is by showing students some website that requires you to log in. What the website shows you depends on **if** you are logged in or not! If you are not logged in, then you will see a login page. If you are logged in, then you will see your own information.

If/then Physical Activity

For this next activity, you will be using a way to generate a random outcome (a die or coin or even a piece of paper that you divide up into 6 numbered sections and have a student volunteer to close their eyes and move their finger randomly through the different sections until you tell them to stop). This will work better with the more possible outcomes you have so we recommend a die or the paper idea. For the rest of this activity, we will be describing the activity as though we are using a die, but it works the same with something other than a die.

Explain to the students that you will roll the die (display the die over a projector if you can, otherwise just tell the students what the outcome is when you roll it on your desk) and they will then do an action depending on the result of the roll. List out the possible outcomes and actions on the board. An example would be:

1 - Do 5 jumping jacks
2 - Wave at the person closest to you
3 - Spin in a circle
4 - Pat your head
5 - Rub your belly
6 - Clap your hands

Phrase it as “if a 1 is rolled, then everyone does 5 jumping jacks!” Make sure the students have enough space to do whatever actions you choose, then start the activity by rolling the die and continuing to roll it. After each roll, students should say together “if it’s a [number that was rolled] then we [do action for that number]” and then do that action. You can also change outcomes during the activity, make 2 or more numbers have the same outcome, or any other modifications. The point of this activity is to just really emphasize an if/then statement is all about checking what a condition is and taking an action based on that condition. Stop the activity after about 10 minutes.

Reflection

Students should use five minutes to reflect on what they learned in the lesson. Some questions that might help them do so include:

- What was an example of an if then statement we created?
- Why do we use if then statements in computer science?

Intro to Mazes

Algorithms are Everywhere: Lesson 3

Lesson Overview

Introduction
5 minutes

Physical Maze Activity
20 minutes

Reflection
5 minutes

Standards

- AP-01 Compare two algorithms
- AP-06 Break down problems
- AP-10 Identify and debug errors

Materials

1. Easy Maze (prvd.)
2. Hard Maze (prvd.)
3. Pens/pencils

Introduction

To begin the lesson, let the students know that today they will continue to learn about algorithms and even try to write their own. Take a second (just as you did earlier in the unit) to say the word algorithm and have them repeat it back to you. Ask if anyone can remind the class of the definition you all came up with earlier in the unit. Thank the students who answered and commend them for their memory! Then, share that today we are going to write an algorithm for solving a maze. Ask the class what the goal in solving a maze is, and once they answer correctly (navigate to the end or goal), you can move on to the first activity.

Physical Maze Activity

After passing out Maze 1 to the class, give students around 2 minutes to complete the activity.

Once they are all done, congratulate them on their work, but share that doing the maze was the easy part. Now, you will write an algorithm for solving the maze. Ask for someone to remind you what an algorithm is (a set of steps). Once you get this answer, expand on it by telling the students they are going to write out a set of steps for solving the maze. Share that this process - writing algorithms - is what Computer Science is all about.

Physical Maze Activity (cont.)

Give students 5-10 minutes to write their set of steps (algorithms) individually. Then, ask for a student to volunteer to have their algorithm tested. You, the teacher, should go to the start of the maze with one of the students' algorithms, and follow it **exactly** reading each step as you go along. You can do so by either projecting the maze to the class, or by setting up the maze for yourself physically by either using obstacles or tape on the ground. If the algorithm is incorrect (it likely will be) give the algorithm back to the student and ask them to fix it (this is the equivalent of debugging in computer science). If another student is ready to have their algorithm tested, repeat the process. If all of the students realize their algorithms are not correct after the first example, give them a couple minutes to try and fix it. Hopefully, after 3 tries, someone will have gotten a correct solution. If not, explain to the students what a correct algorithm would have been.

Once this activity is completed, congratulate the students for writing their first algorithm! Then pass out Maze 2 and instruct them to repeat the process of solving the maze and then writing out their list of steps. This second maze is much more complicated, so you as the teacher will not test it in the same way as you did the first.

Reflection

Discuss as a group the difference in wording between the students' instructions. Point out that some students said "go up" and others said "go forward" when they meant the same thing. Then, explain that this is the purpose of a computer programming language - to create consistency between code written by people from all over the world.

Theseus & Maze Algorithm

Algorithms are Everywhere: Lesson 4 & 5

Lesson Overview

Introduction
10 minutes

Maze Activity
20 minutes

Maze Debugging
20 minutes

Reflection
5 minutes

Standards

- AP-01 Compare two algorithms
- AP-06 Break down problems
- AP-10 Identify and debug errors
- AP-11 Program development

Materials

1. Handout (prvd.)
2. Lesson 3 Mazes
3. Maze 3 (prvd.)
4. Paper and pencil

Introduction

To get started, let the students know they will be reading a Greek myth for the first activity. They may be confused as to why you are reading mythology in a Computer Science class, to which you can share that Computer Science is about solving problems, and reading this myth will help us define our problem for today.

Ask the students if they know what a myth is, and depending on if they get the answer right or not, you can tell them a myth is an old story that usually involves some magical characters or events. Then, you can begin to pass out the text and let them know that today they will be reading a myth about Theseus and his maze.

You can begin reading the text to the students, but ask for volunteers to let them read the story aloud to the class themselves. After reading the text together, these are some questions you can ask to get a discussion going about the story:

1. What was the problem that Theseus faced?
2. How did Theseus solve the problem?
3. Think back to one of our earlier lessons, what is the connection you see between Theseus and that class?

Once the students have answered all the questions and seem to understand the story, you can share that they are going to help Theseus solve the maze by writing an algorithm that could work for ANY maze. Just as Theseus used a piece of string, we as programmers will use an algorithm!

Maze Activity

Start the activity by pointing out how the class had two very different mazes in the last task. Ask the students: If you tried to use your instructions/algorithm from the first maze, would you be able to solve the second maze? (Assumption is that they won't be able to solve their second maze with the instructions from the first).

Let the students begin to brainstorm by asking if there is any way we can come up with an algorithm that will work with ANY maze. It would be helpful to reiterate what an algorithm should be: An algorithm is a set of steps - we need to think of steps that you would take to solve any maze.

Give students 5 minutes to see if they can come up with a solution.

If they cannot by this time, which is a likely scenario, you can give them hints - in a maze you will always walk forward until you can't walk any further. Ask students what you would do if you came to a dead end - they would turn around. Repeat this as an if/then statement to make the connection - IF you come to a dead end/point where you can't keep going forward, THEN you should try to turn (left). (IF there is no left turn, THEN turn right.) Give them 5 more minutes in groups to think of an algorithm to solve the maze in terms of IF/THEN statements. They should ALL write this algorithm down in steps on their own piece of paper.

After they have done this, bring up Maze 3 on the board, and use the students' algorithms to see if they can solve this generic maze.

If they find a solution, get them to test it on their original 2 mazes to see if their solutions work.

This is a good place to end the first part of this lesson plan then pick it up again here next class.

Maze Debugging

Give the students 10 minutes to draw their own mazes on paper.

Ask the students to pass their algorithms from the last task to the person next to them. Each student should use their partners algorithm to see if it will solve the maze they have drawn. This will allow the students to practice debugging each others algorithms.

Reflection

For the last 5 minutes, ask the students if their partners algorithm worked, and if it did, what was good about it. If it didn't work, ask the student what was wrong with the algorithm, and how they can fix it.

Blockly Mazes

Algorithms are Everywhere: Lesson 6

Lesson Overview

Introduction
5 minutes

Blockly Mazes Activity
50 minutes

Reflection
5 minutes

Standards

- AP-03 Create programs with sequences
- AP-04 Create programs using loops
- AP-05 Programs that use conditionals
- AP-06 Break down problems
- AP-10 Identify and debug errors

Materials

1. Computer
2. Projector
3. Blockly Link (prvd.)

Introduction

Note: This is an optional lesson plan that incorporates the online resource Blockly games, which allows students to make algorithms to guide a character through 10 different mazes.

In this lesson, the class is going to end the unit by creating algorithms on computers to help a character get through a maze! Again, ask the students if someone can refresh the class on what the definition of an algorithm is. Once you have reminded the students that an algorithm consists of "specific steps that accomplish a goal," you can pull up the website:

Blockly can be found HERE. (https://blockly.games/maze?lang=en)

Demonstrate how they can move the purple blocks from the gray area to the white area and put them together to make an algorithm, and share that once they create their code, they can hit the "Run Program" button. This will test their algorithm and they will have one of two outcomes. The first possibility is that their character makes it to the flag. In this case, they succeeded and can move on to the next level. The other possible outcome is their character does not make it to the flag, meaning their code doesn't work. Let them know this is a great learning opportunity and encourage them to try and figure out why it didn't work and **debug** their algorithm.

Blockly Mazes Activity

Allow everyone to get on a computer (or pair up if needed), and instruct them to go to Blockly (https://blockly.games/maze?lang=en) and get coding! Let them know you will be here to help if they get stuck. They can also ask each other for help debugging, but they shouldn't share their answers.

Starting at level 3, there is a "repeat until goal" block. If asked, tell students that any code they put within this block will be repeated until the character reaches the goal. This is very useful when we need to repeat a certain pattern of steps over and over again.

If/then statements will also be introduced, but students should already be familiar with this concept from the previous lesson plan on if/then statements. The general algorithm to solve any maze that students developed in lessons 4 and 5 will be needed to solve the final maze.

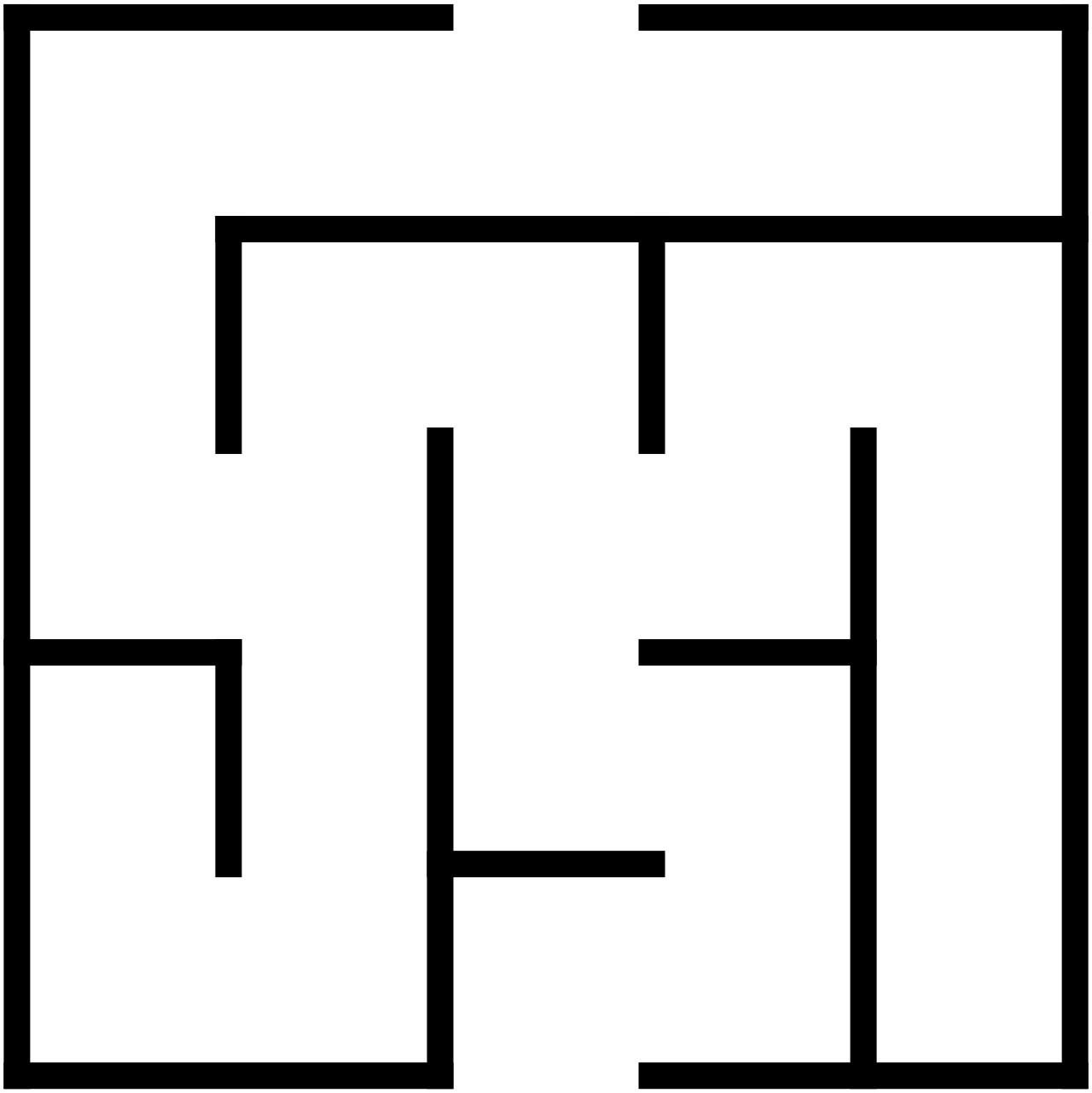
Reflection

Take these five minutes to allow the students to reflect on what they learned over the course of this lesson. A couple questions to help guide them are:

- Did you think using algorithms on a computer was a lot different than doing it on paper? If so, why?
- Would you be confident that you could get out of any maze using the general algorithm we discussed?

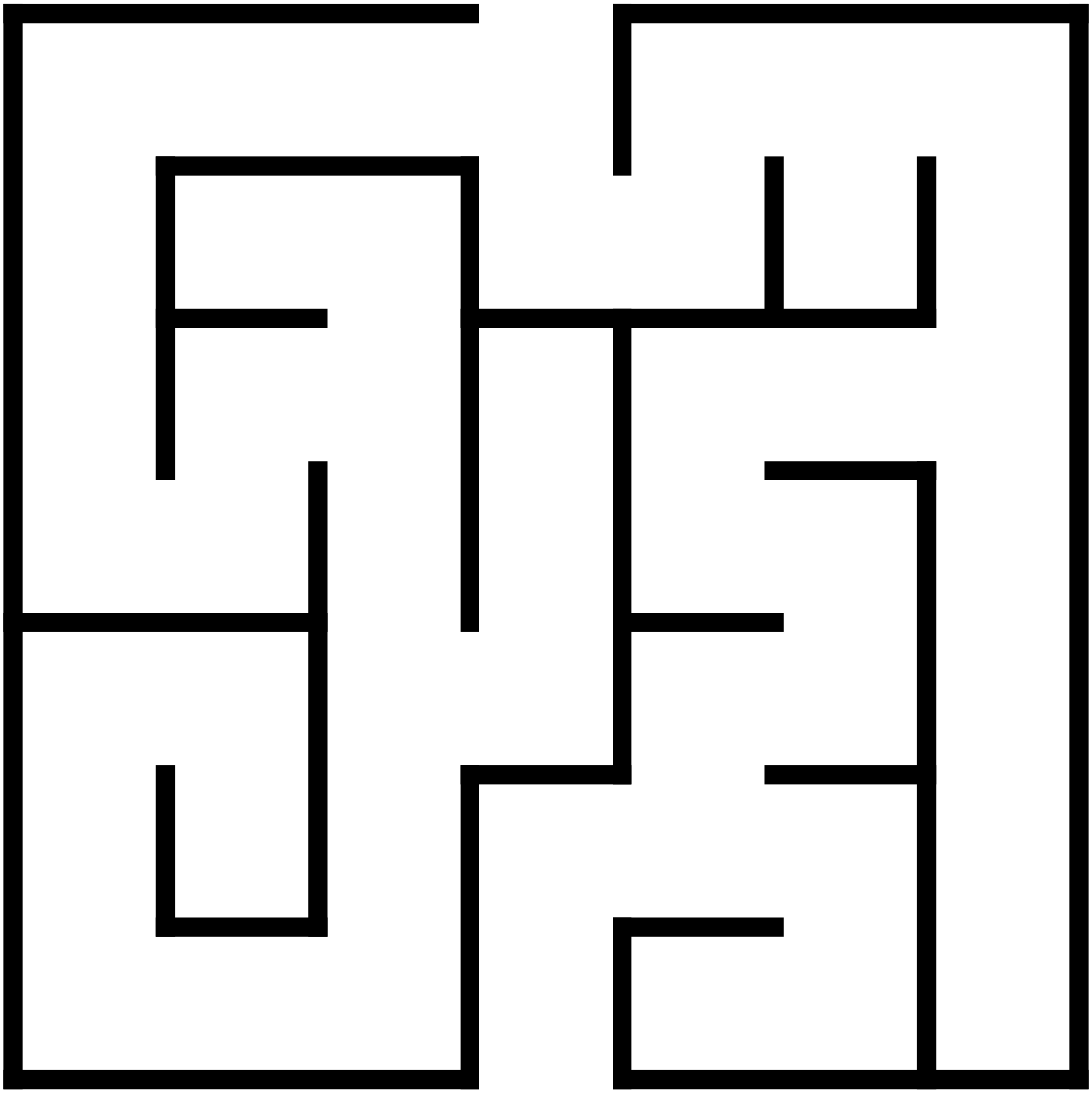
Resources

End



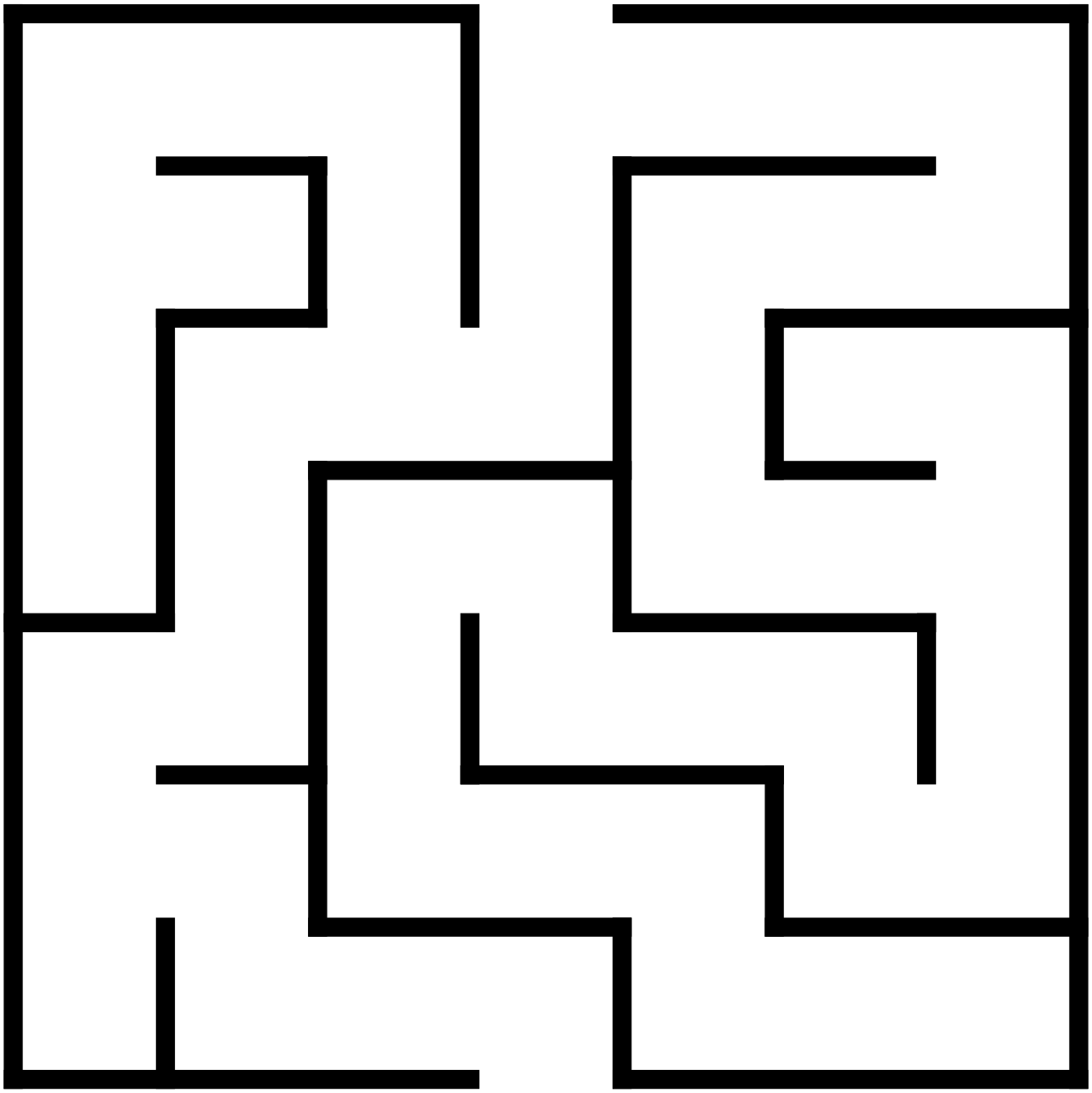
Start

End



Start

End



Start

Theseus and The Minotaur Myth

A long time ago, on an island called Crete, there lived a monster, called the **Minotaur**. It had the body of a man and the head of a bull. The Minotaur ate children. **King Minos of Crete** ordered a special home to be built for the Minotaur. It was a **labyrinth** (a type of maze).

King Minos had defeated the **King of Athens** in a war. To stop King Minos destroying his country, the King of Athens agreed that every 9 years, he would send seven boys and seven girls to King Minos as a sacrifice. King Minos would then send the fourteen young people from Athens into the labyrinth. No one ever found their way out of the labyrinth alive.

The King of Athens had a son called **Theseus**. Theseus was horrified when he heard what was happening to the children, so he travelled to Crete to kill the Minotaur. He decided to pretend to be a prisoner so he could enter the labyrinth.

King Minos had a daughter named **Ariadne**, who fell in love with Theseus, and wanted to help him escape the labyrinth. She gave Theseus a **ball of string** and told him to tie it to the entrance of the Labyrinth. He unwound it as he went through the passages, so that he could follow it back out again.

When Theseus found the Minotaur, he killed it with one blow of his sword. Then he quickly wound up the string and led all the young people out of the labyrinth. They sailed back to Athens. They were the first people to have survived going into the Minotaur's labyrinth!