This page is for

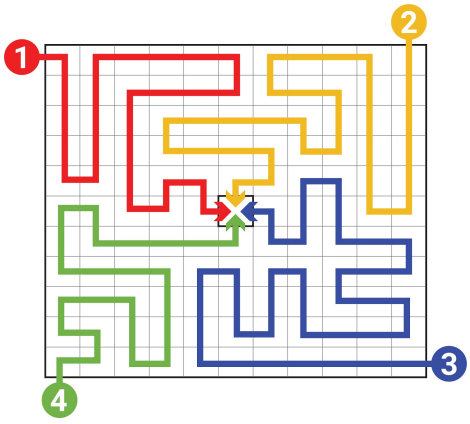
Teachers

Parents and carers

Digital Technologies @ Home
Unplugged activities for students

This activity is for: Years 5-8

Maze escape!



Many thanks to the Girls Programming Network who created this activity.

This activity teaches…

In this activity students create an algorithm, or set of instructions, to navigate through a maze. To create the algorithm, students choose from a finite set of commands. We want students to create the **best** set of instructions to get through the maze - in this activity best means the fewest number of instructions.

Creating good sets of instructions is a very important concept in computing. Solving problems takes a computer's time and energy: as we ask computers to solve bigger and bigger problems (like climate modelling and searching for life beyond earth) it's important to find the most efficient solution available.

The commands students can use to get through the maze include two key programming ideas: **branching**: ie IF something is true THEN do something) and **iteration:** keep doing something a fixed number of times or until a specified condition is no longer true.

This activity is targeted towards students in years **5 to 8** and will take **1 to 2 hours**.

Getting started (read this with your student):

We're going to write a program to move a character to the center of the maze! Use the printed commands to create an algorithm, or set of instructions, to get to the centre of the maze, following one of the lines. Your goal is to use as few commands as possible.

You can do this activity in a few ways: you can use a printed copy of the maze, or you can use masking tape (if you have space) on the floor, creating a room-sized maze following the picture shown. You can either do this with a partner, taking turns coming up with instructions and then following them, or you can do it alone and just use a coin or other item on the print out to follow the instructions, making sure you follow them exactly.

You can print off and cut out all the commands for this activity, or you can write instructions out on a piece of paper.

See a demonstration

cmp.ac/mazevid

This page is for

Students

Maze escape!

Can you get to the centre of the maze with the fewest instructions?

Step 1

Either use the maze on the next page for this activity, or if you have space, you can create this maze on your floor using masking tape.

Step 2

Cut out the commands on pages 4-9.

Step 3

Using only the printed commands, create an algorithm to get you, a partner or a counter to the centre of the maze, following one of the lines.

Step 4

With a partner, or alone, follow your Step 3 instructions exactly. Do they work? Do you need to change them a bit? Can you improve them at all?

Step 5

If your first set of instructions gets you to the middle, work out how many points you earn (see step 6), then make a new set of instructions for the next path. Keep going until you have done all 4 paths.

Step 6

You earn points for each set of instructions. There are 4 paths to the centre of the maze. For each path, the less commands you use the more points you get!

- You get 10 points per path if your instructions have less than 12 commands.
- You get 8 points per path if your instructions have between 12 and 20 commands.
- You get 5 points per path if your code has more than 20 commands.

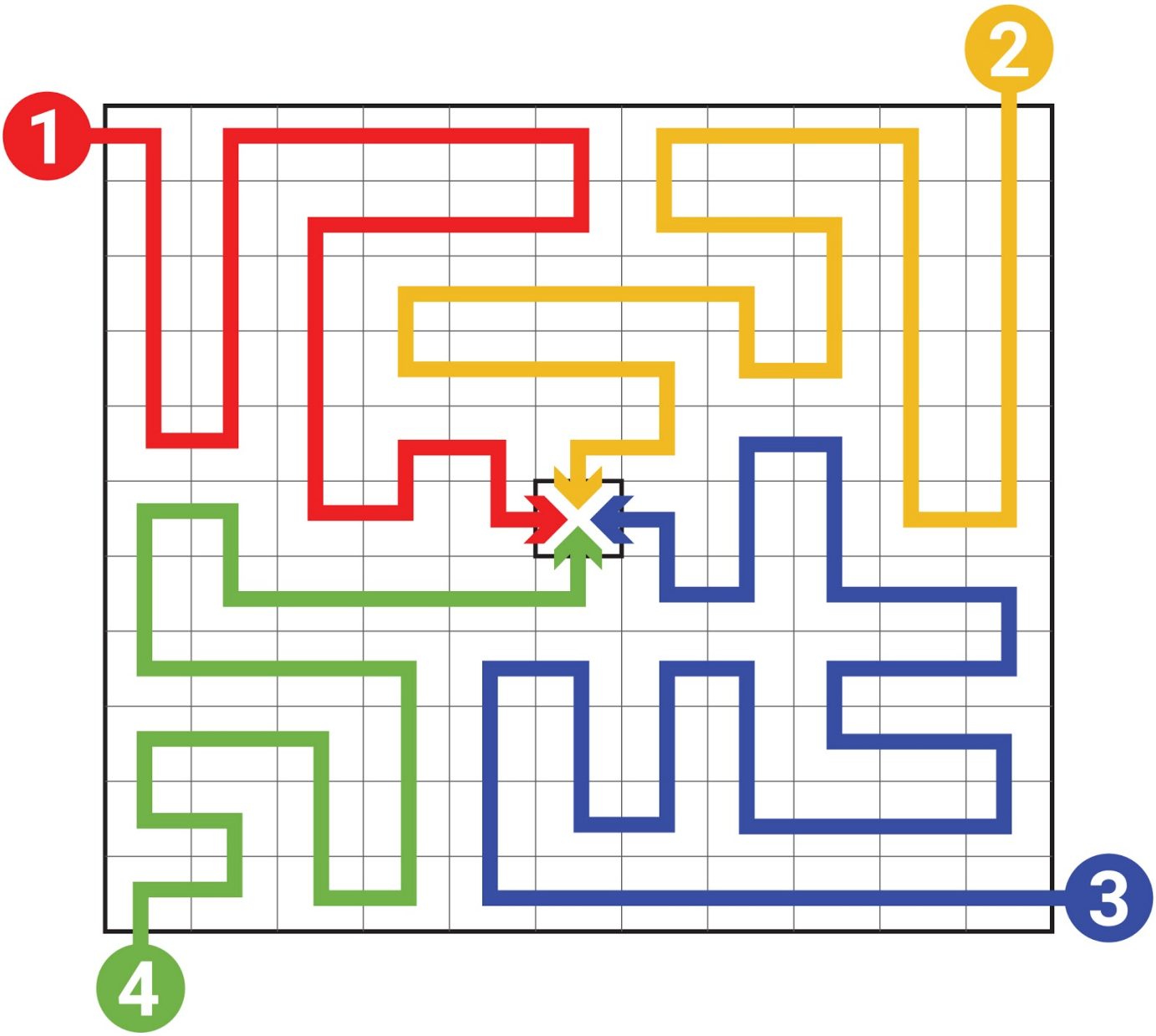
If you write a program that works on any of the paths and will get you (or your partner) to the centre, you get a **bonus 10 points**. Get **another bonus 10 points** if your instructions solve all the puzzles in **less than 7 lines**.

This page is for

Students

Maze escape!

Can you get to the centre of the maze with the fewest instructions?



This page is for

Students

Maze escape!

Can you get to the centre of the maze with the fewest instructions?

Print off these commands

For ____ counts: →
For ____ counts: →
For ____ counts: →
For ____ counts: →

This page is for

Students

Maze escape!

Can you get to the centre of the maze with the fewest instructions?

Code snippets - while loop

While not at the end of maze: →
While not at the end of maze: →
While not at the end of maze: →
While not at the end of maze: →

This page is for

Students

Maze escape!

Can you get to the centre of the maze with the fewest instructions?

Code snippets - while loop

While there is path ahead: →
While there is path ahead: →
While there is path ahead: →
While there is path ahead: →

This page is for

Students

Maze escape!

Can you get to the centre of the maze with the fewest instructions?

Code snippets - if

If there is a path to the left: →
If there is a path to the left: →
If there is a path to the right: →
If there is a path to the right: →

This page is for

Students

Maze escape!

Can you get to the centre of the maze with the fewest instructions?

Code snippets

If there is a path in front: →
If there is a path in front: →
If there is a path in front: →
If there is a path in front: →

This page is for

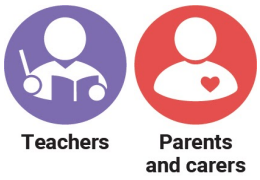
Maze escape!

Can you get to the centre of the maze with the fewest instructions?

Turn to the right	Turn to the left
Turn to the right	Turn to the left
Turn to the right	Turn to the left
Turn to the right	Turn to the left
Turn to the right	Turn to the left
Step forward	Step forward
Step forward	Step forward
Step forward	Step forward
Step forward	Step forward

Answer key

Choose if you want to print this for your kids or keep it to yourself!



There are many possible solutions to this problem. The most efficient solution uses **6 commands** that solves all mazes:

```
While not at the end of maze:
→ If there is a path to the left:
    → Turn to the left
    If there is a path to the right:
    → Turn to the right
    Step forward
```

The reason we don't need the "If there is a path in front" before stepping forward - is because we've checked if the line turns already, it will *always* be forward.

Here is an example to follow line 1 that would only earn 5 points:

```
Step forward
Turn to the right
For 4 counts:
→ Step forward
Turn to the left
Step forward
Turn to the left
For 4 counts:
→ Step forward
Turn to the right
For 4 counts:
→ Step forward
Turn to the right
Step forward
Turn to the right
```

```
(continued):
For 3 counts:
→ Step forward
Turn to the left
For 4 counts:
→ Step forward
Turn to the left
Step forward
Turn to the left
Step forward
Turn to the right
Step forward
Turn to the right
Step forward
Turn to the left
Step forward
```

There are many possible solutions! The best way is to run through someone's program to see if it gets you to the end!

This page is for

Teachers

Want more?

Here are some further activities, online resources, assessment ideas and curriculum references.

Adapting this activity

Once students understand how to complete this activity, ask them to perform similar steps using a map - or the steps to get to their bedroom.

Keep the conversation going

- What were some strategies that you used to solve the problem?
- How does it relate to other types of instructions like directions or navigation?
- Did you see any repeated patterns within the solution? In programming we have functions that could repeat a certain pattern - even if they don't come directly after each other.

Keep learning

To move this type of thinking into the computer, students can write their own programs that follow these types of steps! We recommend trying the Blockly Turtle course:
cmp.ac/blocklyturtle

For younger students who would like to create a maze using the Scratch programming language, there is a printable step by step guide available here:
cmp.ac/scratchmaze

For teachers creating a portfolio of learning or considering this task for assessment

Ask students to submit their best algorithm for the 4 paths in this worksheet.
You could also ask them to design their own maze and best algorithm.

Linking it back to the Australian Curriculum: Digital Technologies

Algorithms
Design, modify and follow simple algorithms involving sequences of steps, branching, and iteration (repetition) (ACTDIP019 - see cmp.ac/algorithms)

Refer to aca.edu.au/curriculum for more curriculum information.