



TURING TUMBLE™

BUILD MARBLE-POWERED COMPUTERS

Educator Guide

Version 1.0 - Covers challenges 1 to 30

Welcome Educators!

We're excited that you're using Turing Tumble in your classroom! This kit is easy to use. There are no batteries to charge, no apps to install or update, and no cords needed...just show the students how to follow the puzzle book and your class will be ready to go. You can use it as part of math stations, unit studies, learning engineering concepts, independent study, introducing computer science principles, free choice time, or in a library learning center.



This educator guide is a free companion to Turing Tumble. Everything is black and white to make it easy for you to print or copy. This version of the guide covers the first 30 challenges in the book. If it's useful, we'll do the same for the rest of the challenges.

What's in this guide?

Inside are two types of resources:

1. Lesson curricula: As students progress through the puzzles, there are certain times where you can take a break for a lesson. The lessons anchor the concepts learned through Turing Tumble into real life applications. You'll find materials for each of these lessons spread out among the puzzles.
2. Help with individual puzzles: We show the puzzle, the solution, an explanation of the solution, common pitfalls, and the underlying concept that the puzzle is intended to teach.

As you walk around a classroom, you can glance at this guide to quickly learn how to coach a student trying to solve a puzzle and help them understand the underlying concepts. We know there are things that you will find work well... or not at all. Please reach out to us with an email or on our Turing Tumble

Community page to share your ideas and ask questions.

What is Turing Tumble?

Turing Tumble is a game where players build mechanical computers powered by marbles to solve logic puzzles. While they play, they learn key computer science concepts and build skills essential to coding. It also helps students learn how computers work at a basic level: how simple switches, connected together in clever ways, can do incredibly smart things. It can be used as a standalone tool to teach how computers work, as a precursor or supplement to coding lessons, or as a helpful manipulative to reinforce programming and mathematical concepts.

What does Turing Tumble teach?

Turing Tumble teaches a number of concepts and skills that are fundamental to computers, programming, and digital electronics. The concepts include logic gates, truth tables, conditionals, binary, binary operations, and digital circuit design. It also builds skill in computational thinking, logic, algorithm design, critical thinking, debugging/troubleshooting, fine motor, spatial reasoning, and persistence.

Is this guide useful to you?

Please tell us what you think! We're always looking for ways to make Turing Tumble and this guide better. Just send us an email at hello@turingtumble.com.

Did you find an error?

Please send a quick email to hello@turingtumble.com and we'll fix it right away. We don't like bugs.



| | |
|--|-----------|
| Basics for Classroom Use..... | 1 |
| How it Works | 2 |
| How to Use this Guide | 5 |
| Computer Logic Lesson #1: How is Turing Tumble a Computer?..... | 7 |
| Computer Logic Lesson #2: Ramps..... | 11 |
| Challenge #1: Gravity | 13 |
| Challenge #2: Re-entry..... | 15 |
| Challenge #3: Ignition | 17 |
| Challenge #4: Fusion..... | 19 |
| Computer Logic Lesson #3: Crossovers | 21 |
| Challenge #5: Entropy | 23 |
| Challenge #6: Total Internal Reflection | 25 |
| Challenge #7: Path of Least Resistance | 27 |
| Computer Logic Lesson #4: Bits | 29 |
| Challenge #8: Depolarization | 31 |
| Challenge #9: Dimers | 33 |
| Challenge #10: Double Bond | 35 |
| Challenge #11: Selectivity..... | 37 |
| Computer Logic Lesson #5: Interceptors..... | 39 |
| Challenge #12: Duality - Part 1 | 41 |
| Challenge #13: Duality - Part 2 | 43 |
| Computer Logic Lesson #6: Conditional Statements..... | 45 |
| Challenge #14: Duality - Part 3..... | 49 |
| Challenge #15: Inversion | 51 |
| Challenge #16: Termination | 53 |
| Challenge #17: Fixed Ratio | 55 |
| Computer Logic Lesson #7: Logic Gates | 57 |
| Challenge #18: Entanglement..... | 59 |
| Computer Logic Lesson #8: Truth Tables..... | 61 |
| Challenge #19: Entanglement..... | 65 |
| Challenge #20: Symbiosis | 67 |

| | |
|---|-----------|
| Computer Logic Lesson #9: Registers..... | 69 |
| Challenge #21: Quantum Number | 73 |
| Challenge #22: Depletion | 77 |
| Challenge #23: Tetrad..... | 81 |
| Challenge #24: Ennead | 83 |
| Challenge #25: Regular Expression | 85 |
| Challenge #26: Nucleus | 87 |
| Challenge #27: Reflection | 89 |
| Computer Logic Lesson #10: Gears and Gear Bits | 91 |
| Challenge #28: Latch..... | 97 |
| Challenge #29: One-Shot Switch | 99 |
| Challenge #30: Overflow | 101 |



Recommended Age

Turing Tumble is best for 3rd grade and up. The puzzle book is laid out so that a student can independently get started and progress at his/her own pace. Even college students can gain a fresh perspective on computers.

Single Player or Partners

We recommend only 1-2 students per board. With this game, true understanding happens when students are physically working through the puzzle. You will see your students add pieces and then drag their finger down the board to test their prediction of where the balls will fall. When you see this happening, you will know they are getting it, but this kind of hands-on learning can't happen very well in a group of three or more.

Independent Learning and Intentional Scaffolding

Students must keep track of their own progress – it is *crucial* that they understand each puzzle and don't try to skip ahead, even if the first puzzles are too easy for them. Each puzzle introduces a new concept, rule, or trick and they'll find themselves quickly lost if they don't complete each of the puzzles in turn. We often see students try the first 3 puzzles, flip to the last puzzle, and think they've solved it immediately because they don't yet know all the rules.

Practice Guide

Don't forget about the Turing Tumble Practice Guide! You can download it for free at edu.turingtumble.com. Besides offering all the puzzles in an easy-to-print/copy black and white form, it also contains 30 extra “practice puzzles” placed in between the regular ones. The practice puzzles lower the learning curve by easing players into new concepts more gradually.

Turing Tumble on a Screen

Online Turing Tumble simulators are great for demonstrating how to use Turing Tumble to a class. You can project a simulator on a screen, build machines on it quickly, and run it right there on screen for your class to watch. Currently, we recommend using either [this simulator](#) or [this simulator](#).

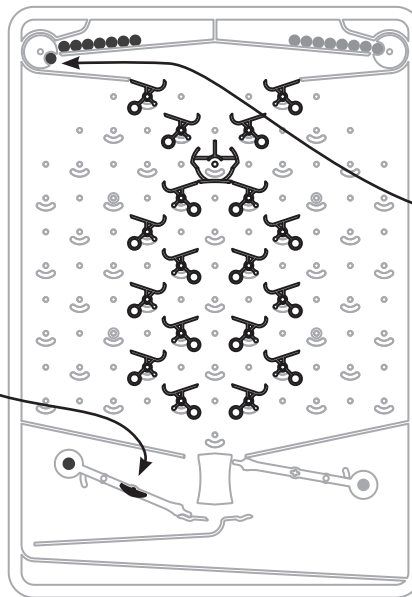
How it Works



The game board releases one ball at a time from the top:

Press here to start the machine...

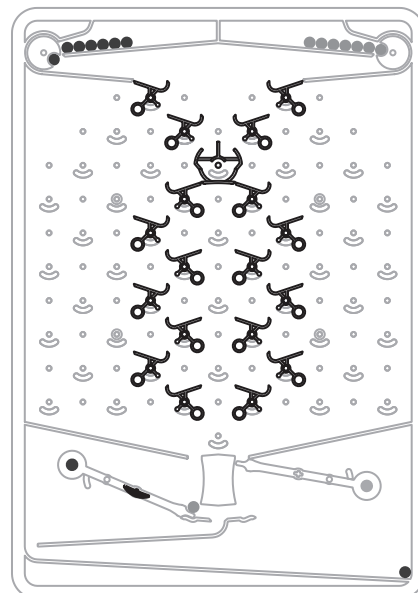
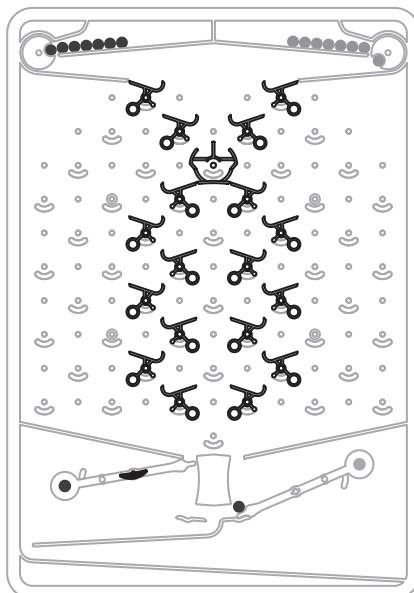
...and a ball is released from the top.



Each ball falls down the board and when it reaches the bottom, it pushes down one of two black flippers at the bottom that release another ball.

If it pushes down the **right** flipper, a **red** ball is released.

If it pushes down the **left** flipper, a **blue** ball is released.



Players add logic by putting **six different types of parts** onto the board:



RAMP

The **ramp** directs balls to the right or to the left. It is a reversible piece.



CROSSOVER

The **crossover** lets ball paths cross over one another. Balls come in one side and exit on the opposite side.



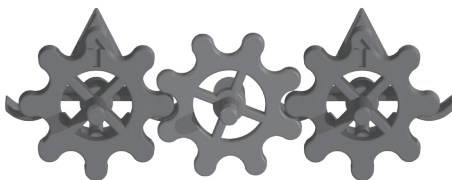
BIT

The **bit** adds logic. It stores information by pointing to the right or to the left, like a 1 or 0. It becomes more and more important as the puzzles progress.



INTERCEPTOR

When the computer's objective is complete, the **interceptor** can be used to stop the computer from releasing any more balls.



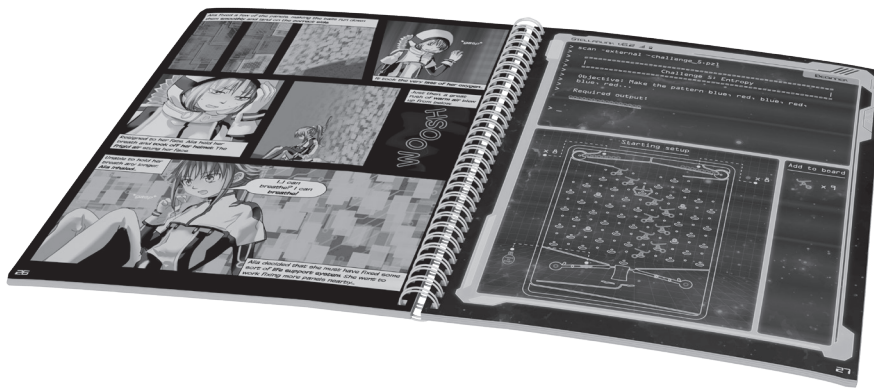
GEARS AND GEAR BITS

Like the bit, the **gear bit** stores information by pointing right or left, but when a gear bit is flipped, it can also flip other gear bits connected to it by **gears**. Every kit comes with a small bag of black washers. These should be added behind the gear bits when only two gear bits are connected together to increase friction. However, if *more* than two gear bits connected, the washers shouldn't be used.

The gears and gear bits are mind-bending, but they add a whole new level of functionality to the board. They also make the computer "Turing-complete", which means that if the board was big enough, it could do anything a regular computer could do!

Turing Tumble comes with a book of 60 puzzles. They start out easy and become steadily more challenging. As you move through the puzzle book, new types of parts are unlocked. For example, you start by just using the ramp pieces, but after 4 puzzles, the crossover is unlocked. Each puzzle leads the player to discover new concepts that can be applied to more complicated puzzles later on.

There is also a story woven into the puzzles to give them context and to hopefully make them more interesting for some students. Each puzzle brings Alia the space engineer closer to rescue from a seemingly deserted planet.





Each challenge is listed with the page number for quick reference. The puzzle page is shown at the top with the solution next to it. Some players' questions can be answered with a quick glance at the solution.

Computer logic lessons

In this guide, “Computer Logic Lessons” are positioned before students begin using a new part or before they learn a new concept. It is organized this way so that educators can anticipate what will be learned. However, you may find it easier to teach the lesson after the students have had some practice with the parts.

Puzzle-specific educator resources

Each challenge has its own teaching and learning concepts listed below the visuals of the challenge and solution. An educator can flip to any challenge and find guidance for helping students with the specific skills and hang-ups that might be encountered.

You'll quickly notice that there is repetition in the educator resources for each puzzle. For example, challenges five, six, and seven all teach students how to use the crossover. Therefore, in the educator resources for each challenge, you'll see many of the same points discussed. This allows you, as a teacher, to consult only the educator resources for the relevant puzzle, without having to look back at the educator resources for previous puzzles to understand it.

- **What players learn about computer logic**

In this section, you'll find quick bullets about the connection between the puzzle and what's going on inside a computer. The Computer Logic Lessons give a more thorough description of the connection.

- **What players learn about the game**

This section shows what players learn about the game through this puzzle. It could be a functional aspect of the game (like how the board or parts work), something to notice about the puzzle book or the challenges them-

selves, or a trick the player is learning that will need to be reused later on.

- **Possible hang-ups**

In this section, hints and tricks are included to help students complete the challenges. These are not exhaustive, but highlight the logic steps we've seen players struggle with.

As you use this Educator Guide, we would appreciate your input so that we can improve this guide and add content for future use.

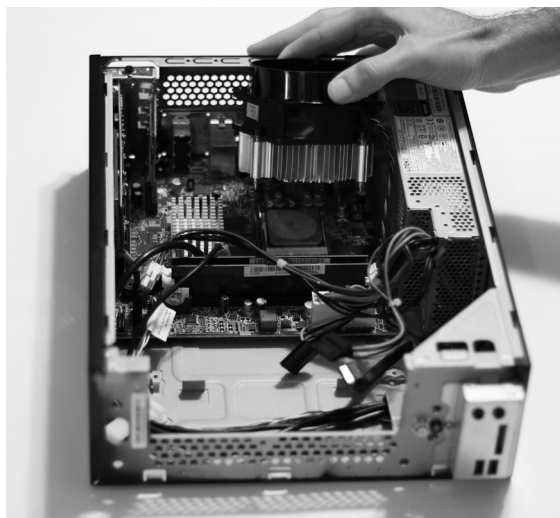
Computer Logic Lesson #1: How is Turing Tumble a Computer?

If you're like most people, you're probably wondering how on Earth this marble contraption could possibly be a computer. It has no screen, no keyboard, and no electronics. How is it anything like a computer?

To start, let's look at what's inside a regular desktop computer.

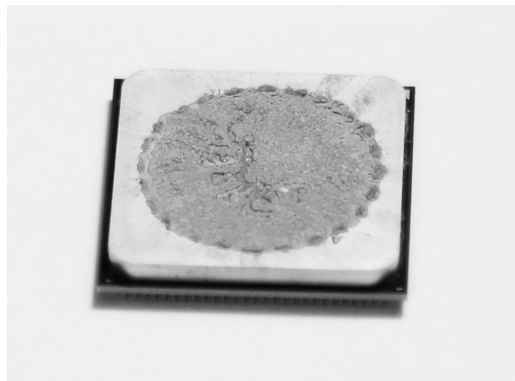
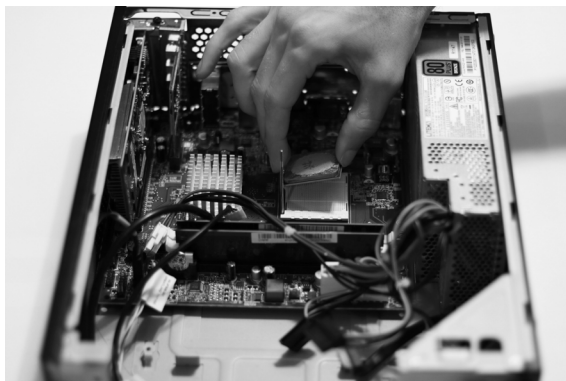


There's all sorts of things in there like circuit boards, fans, lights, and motors, but those aren't the smart parts of a computer. In fact, they're only there to support the computer's processor - a little rectangular chip under a big fan that cools it down.

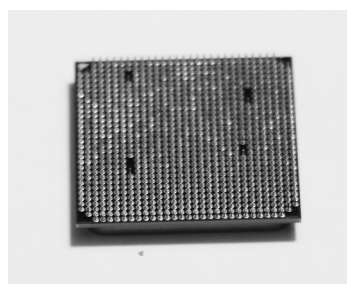


The computer processor (or central processing unit - CPU) is where all the “smarts” in a computer happen. It’s the part that runs programs and does math and logic. The processor does a lot of hard work when the computer is running and it generates a lot of heat in the process. The fan on top of it is there to cool it down so that it doesn’t overheat.

Here’s what this computer’s processor looks like when it’s taken out:



The goopy stuff on top of the processor is thermal paste. It makes it easier for heat to transfer out of the chip. If we clean off the thermal paste, it looks like this on top and bottom:



On the bottom of the processor, you can see over a thousand little pins sticking out. The pins connect the inside of the processor to things on the outside. Some pins are inputs - they send information into the processor or simply provide it power. Other pins are outputs - the processor uses the outputs to send information to the rest of the computer. For example, the computer’s keyboard would provide *input* to a processor while a screen would show information coming from the *output* of a processor.

What’s inside a computer processor?

Switches. Lots and lots of switches. *Billions* of switches. The switches are so small that you can’t even see them with your eye. In fact, they’re so small that you couldn’t even see them through a microscope because visible light itself is too big. These days, the switches in a computer

processor are about a thousand times smaller than the thickness of a human hair.

The following video zooms into a computer processor all the way until you can see the individual switches (called “transistors”) inside and the tiny copper wires connecting them together. As it zooms further and further in, you’ll notice the images change from color to black and white. That’s when the creators of the video had to switch from taking pictures with light to taking pictures with electrons, because they’re so much smaller. You can see the video here:

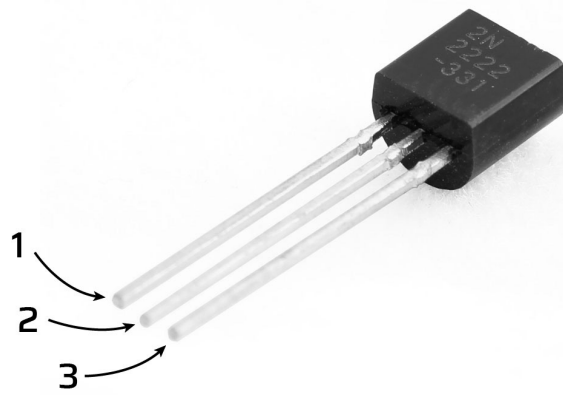
<https://youtu.be/Knd-U-avGOc>

How do switches do *anything* smart?

When you hear “switch” you probably think about the switch on your wall that turns the light on and off, and it seems impossible to think that switches like that could do anything other than turn stuff on and off.



And you’d be right about your light switch. It can’t do anything smart. In order for switches to be able to do smart things, **the key is that they must be able to be flipped by the same type of energy as they control.** That’s because switches need to be able to flip other switches to do smart stuff. A light switch can’t flip other switches because it takes *mechanical* energy to flip it, but it controls *electrical* energy. You can’t connect the output of one light switch to the input of another.



On the other hand, switches in a computer (transistors) are flipped by *electrical* energy and also control *electrical* energy. The image above is of a relatively large, individual transistor that's about the size of the nail on your pinky finger. The way it works is this: pin 1 controls the flow of electricity from pin 2 to pin 3. If electricity is being pushed into pin 1, electricity can flow from pin 2 to pin 3. Otherwise, it can't.

Similarly, in Turing Tumble, the switches (that is, the blue and purple parts we call "bits" and "gear bits") are flipped by *mechanical* energy, and they also control *mechanical* energy: They're flipped by a ball rolling over them, and they control whether a ball rolls off their left or right side.

Since the switches in computer processors and in Turing Tumble are flipped by the same type of energy as they control, it's possible for one switch to cause another switch to flip. As you work through the puzzles in Turing Tumble, you'll discover how this one, simple property makes it possible to build machines of limitless capability!

Computer Logic Lesson #2: Ramps

In these first challenges, the only part used is the “ramp”. It looks like this:



The purpose of the ramp is to make balls always go in a certain direction. If you put it on the board pointing to the left, the balls will go left. If you put it on the board pointing to the right, the balls will go to the right.

What do ramps look like in a regular, electronic computer?

In an electrical computer, wires are used for the same purpose as ramps. You use wires to direct electricity where you want it to go. Sort of like a pipe for electrons. Sometimes wires direct electricity to a switch or another electrical component, or sometimes they just lead to more wires that branch off.

There are even tiny wires in microchips like this one:



Photo of a Motorola 68040 processor taken by Konstantin Lanzet. Shared under CC BY-SA 3.0 license at https://en.wikipedia.org/wiki/Motorola_68040.

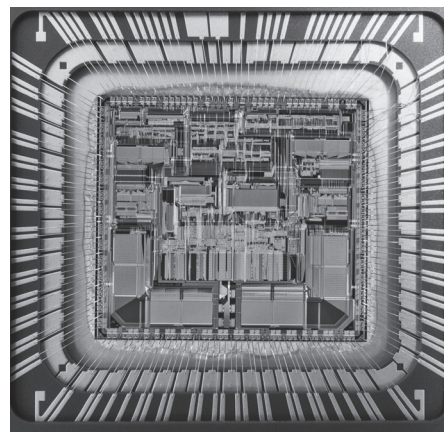
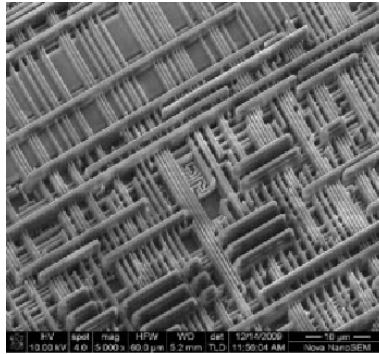


Photo of a decapped Motorola 68040 processor taken by Gregg M. Erickson. Shared under CC BY 3.0 license at https://en.wikipedia.org/wiki/Motorola_68040.

The picture on the left shows a microchip like you might see inside your computer. Microchips are just tiny electrical circuits, covered in plastic or ceramic to keep them safe. On the right is the same microchip, but without

the protective coating.

If you zoom in far enough, you can see the tiny copper wires connecting different parts of the electrical circuit.



This scanning electron microscope image of a decapped chip was used with permission from DELTA Microelectronics (<https://asic.madebydelta.com>)

In Turing Tumble, ramps are like wires and the balls are like electricity. When you place ramps on the board, you are making the paths the balls can travel, just like how wires make the paths through which electricity can travel.

Challenge #1: Gravity

(page 15 in puzzle book)



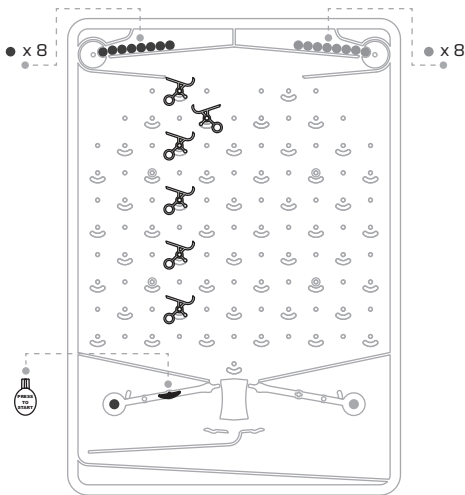
Challenge 1: Gravity

Objective: Make all of the blue balls (and only the blue balls) reach the end.

Required output:



Starting setup



Available parts

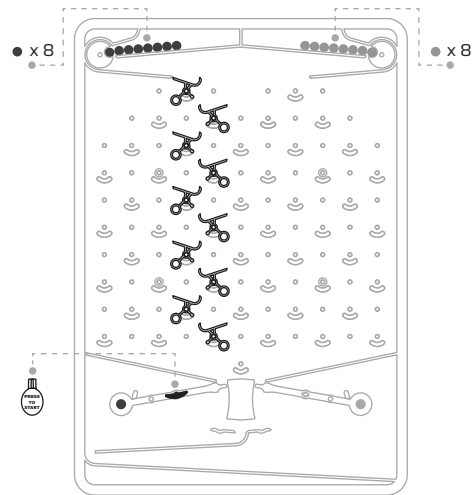


x 4

Challenge 1 Solution

Explanation: The four ramps complete the path from the top of the board to the bottom of the board.

Remember that it's against the rules to allow balls to fall freely for any distance! When a ball rolls off a part, it must immediately land on the next part.



What players learn about computer logic:

- The only part used in this challenge is the ramp. Ramps are just like wires in an electrical circuit and the balls are like electricity. When you place ramps on the board, you are setting up paths that balls can travel through, just like how wires set up paths that electricity can follow.

What players learn about the game:

- Familiarity with the layout of the puzzles: required output, starting setup and available parts.
- The levers on the bottom are connected to the ball release on top.
- Practice putting the ramps on the board. They will discover that the ramps are **reversible** and can go on the board in either direction, depending on where you want to route the balls.

Possible hang-ups:

- Your machine cannot let balls drop freely for any distance. Balls might be bouncing unpredictably because there aren't parts leading all the way down or the ramps are facing the wrong direction.
- Starting the machine: Once the machine is in motion, you cannot touch it or otherwise interfere with its operation. Press the start button down once to start the machine. Once a ball reaches the bottom, it will push a lever to trigger the next ball.
- Putting the ramps on the board: the ramps are reversible! Note which direction they face before putting them on the board.

Challenge #2: Re-entry

(page 16 in puzzle book)



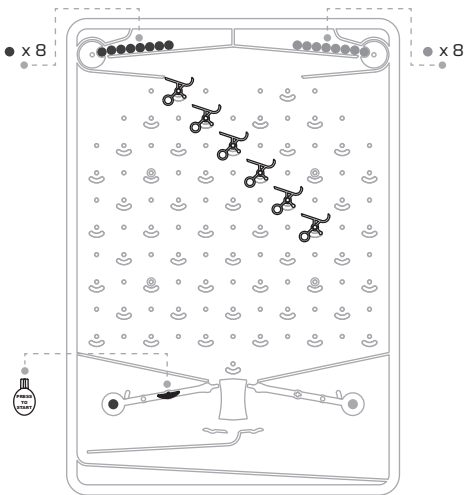
Challenge 2: Re-entry

Objective: Make all of the blue balls (and only the blue balls) reach the end.

Required output:



Starting setup



Available parts

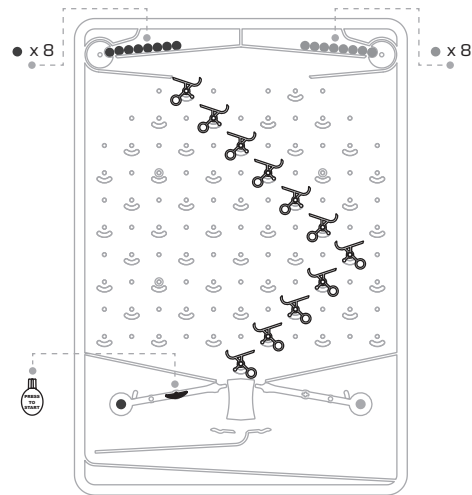


x 5

Challenge 2 Solution

Explanation: The starting setup leads the balls over onto the right side. That's a problem! If the balls hit the right lever, they'd release red balls, but you're only supposed to let the blue balls get to the bottom.

To solve this puzzle, you have to use the 5 ramps to lead balls back over to the left lever.



What players learn about computer logic:

- The only part used in this challenge is the ramp. Ramps are just like wires in an electrical circuit and the balls are like electricity. When you place ramps on the board, you are setting up paths that balls can travel through, just like how wires set up paths that electricity can follow.

What players learn about the game:

- Familiarity with the layout of the puzzles: required output, starting setup and available parts.
- The levers on the bottom are connected to the ball release on top.
- Practice putting the ramps on the board. They will discover that the ramps are **reversible** and can go on the board in either direction, depending on where you want to route the balls.

Possible hang-ups:

- Players might initially place ramps so that the ball triggers the right lever. This will be a good time to be sure they've looked at the back of the board to see how the lever on the bottom is connected to the ball release on top. Encourage them to place the ramps so that the balls are guided back over to the blue side.

Challenge #3: Ignition

(page 17 in puzzle book)



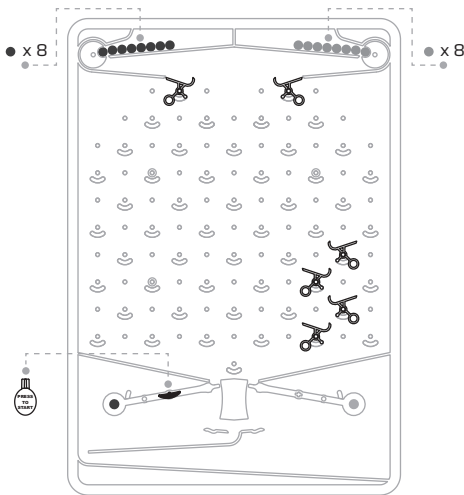
Challenge 3: Ignition

Objective: Release one blue ball and then all of the red balls.

Required output:



Starting setup



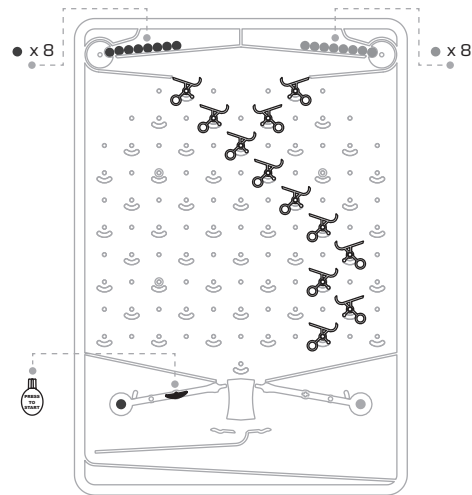
Available parts



x 6

Challenge 3 Solution

Explanation: Like the last puzzle, the ramps bring the paths of the red and blue balls into one path that leads to the right lever. After the first blue ball, all the rest are red.



What players learn about computer logic:

- The only part used in this challenge is the ramp. Ramps are just like wires in an electrical circuit and the balls are like electricity. When you place ramps on the board, you are setting up paths that balls can travel through, just like how wires set up paths that electricity can follow.

What players learn about the game:

- You don't need to press the right lever to get red balls to the bottom. A blue ball can trigger the ball release on the red side, allowing a red ball to fall.

Possible hang-ups:

- It can be tricky for players to figure out how to release one blue ball and all the rest red. Remind them that the blue will release when they press the start button, and they can determine what color ball comes next based on how they direct the balls with the ramps.

Challenge #4: Fusion

(page 18 in puzzle book)



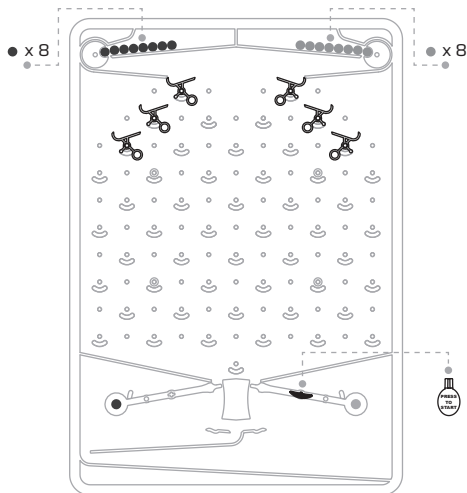
Challenge 4: Fusion

Objective: Release one red ball and then all of the blue balls.

Required output:



Starting setup

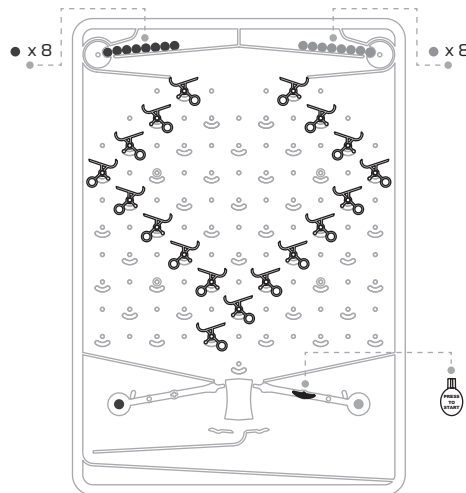


Available parts



Challenge 4 Solution

Explanation: The paths need to come together, but they start far apart! Use the ramps to bring the paths together and lead all of the balls to the left side.



What players learn about computer logic:

- The only part used in this challenge is the ramp. Ramps are just like wires in an electrical circuit and the balls are like electricity. When you place ramps on the board, you are setting up paths that balls can travel through, just like how wires set up paths that electricity can follow.

What players learn about the game:

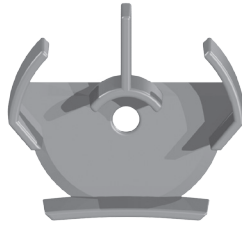
- The start button can be placed on either lever. In this challenge, it need to go on the right side so that a red ball is released first.
- As noted in the talk bubble, you can sometimes solve challenges with less parts than are listed in the “Available parts” section. You can challenge students who are finding the introductory puzzles simple to create the most elegant and simple solution.

Possible hang-ups:

- It can be tricky for players to figure out how to release one red ball and all the rest blue. Remind them that the red will release when they press the start button, and they can determine what color ball comes next based on how they direct the balls with the ramps.
- There are many ways a player can lay out the ramps to solve this challenge. The example on the top is just one solution.

Computer Logic Lesson #3: Crossovers

In puzzle 5, the “crossover” part is introduced. It looks like this:

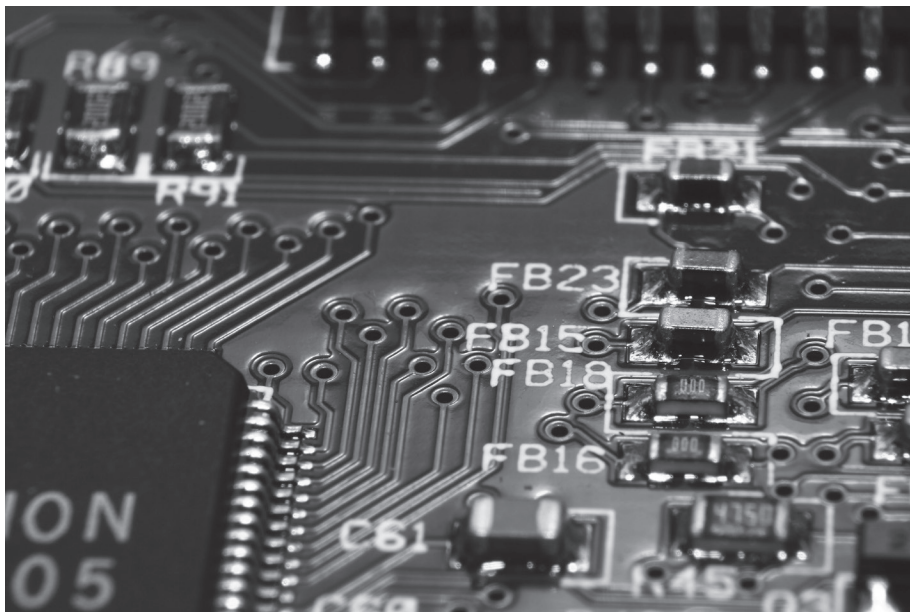


The purpose of the crossover is to let balls paths cross over each other. A ball coming in the left side exits on the right. A ball coming in the right side exits on the left.

What do crossovers look like in a regular, electronic computer?

The crossover acts like two wires crossing over each other, but *not* touching each other. Electricity can move along each of the wires, but the crossing paths don't interfere with each other. It would be impossible to create complicated circuits without wires that cross.

Circuit boards are used in electronic computers to keep all the wire connections sturdy, safe, and fixed in place. A circuit board is made of a hard, flat material. On the surface is a thin layer of copper, etched into a carefully designed pattern of wires that connect the electronic components on the board. Below is an example of a circuit board. You can see the copper wires in between all the electronic parts:



If the wires on a circuit board are all on a flat surface, how do they cross over each other without touching? Most circuit boards actually have multiple layers of copper wires sandwiched between insulators. Little holes, called “vias”, connect wires in the various layers of copper. In the picture above, you can see lots of little vias connecting the various layers of the circuit board, allowing the copper wires to cross over and under each other without touching.

In computer chips, there are also multiple layers of tiny wires that cross over and under each other, sort of like how overpasses allow cars to drive over other roads.

exit out the left.

- Solidifies their understanding of the levers at the bottom and ball releases at the top.

Possible hang-ups:

- In this puzzle, the start button goes back on the left lever.
- When putting the crossover pieces on the board, the smile on the crossover fits a bit snugly into the open smile on the board. It needs to be pushed all the way onto the board.
- Since this is a mechanical computer, it makes a difference if the parts are placed correctly. Players will start to notice if they didn't slide the parts all the way onto the board because the balls will be a little less predictable.

Challenge #6: Total Internal Reflection

(page 22 in puzzle book)



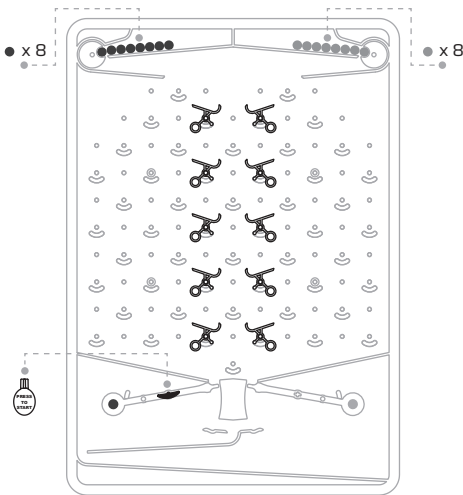
Challenge 6: Total Internal Reflection

Objective: Make the pattern blue, red, blue, red, blue, red...

Required output:



Starting setup



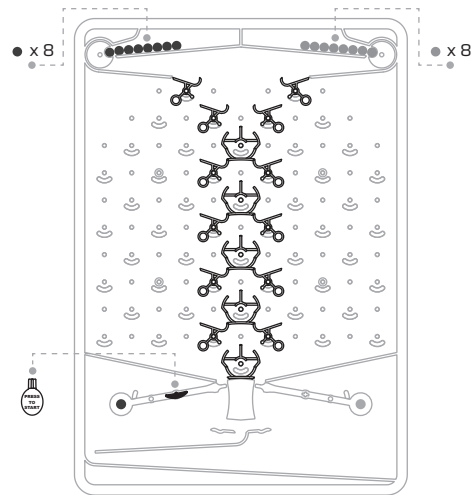
Available parts



Challenge 6 Solution

Explanation: The paths for the red and blue balls cross over each other five times. Crossovers must be placed at each point where they cross.

In the last two puzzles, the paths crossed each other one and three times. One, three, and five are all odd numbers. What would happen if the paths crossed over each other an even number of times?



What players learn about computer logic:

- The crossover is a mechanical version of two wires crossing over each other (but not touching). In electrical circuits, wires need to connect parts all over the circuit board to each other. Wires weave around, over, and under each other to route electricity where it needs to go.
- Circuit boards are usually made of several layers of wires. Wires can cross each other if they are on separate layers. Even on a one layer circuit board, wires can cross using “jumper wires”. A jumper wire is a wire that is soldered on top of a circuit board to hop over other wires on the circuit board.

What players learn about the game:

- This challenge requires the same output as challenge five, but it is a more simple and elegant solution.
- Practice putting the crossover pieces on the board.

- Solidifies their understanding of the levers at the bottom and ball releases at the top.

Possible hang-ups:

- When putting the crossover pieces on the board, the smile on the crossover fits a bit snugly into the open smile on the board. It needs to be pushed all the way onto the board.
- Since this is a mechanical computer, it makes a difference if the parts are placed correctly. Players will start to notice if they didn't slide the parts all the way onto the board because the balls will be a little less predictable.

Challenge #7: Path of Least Resistance

(page 23 in puzzle book)



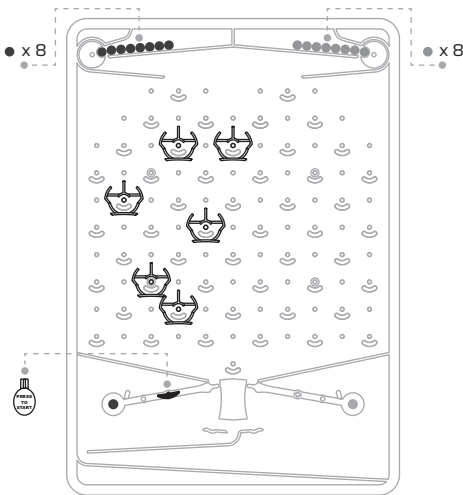
Challenge 7: Path of Least Resistance

Objective: Create a path for the blue balls to reach the output with only 6 ramps.

Required output:



Starting setup

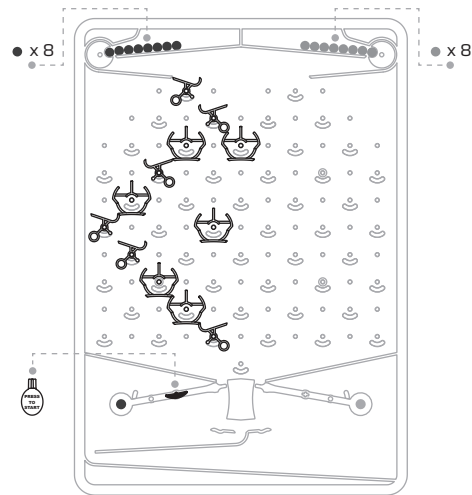


Available parts



Challenge 7 Solution

Explanation: This one is a little trickier than the last challenge. At the top, you must make a choice on whether to go to the left or to the right. You must go to the left this time.



What players learn about computer logic:

- The crossover is a mechanical version of two wires crossing over each other (but not touching). In electrical circuits, wires need to connect parts all over the circuit board to each other. Wires weave around, over, and under each other to route electricity where it needs to go.
- Circuit boards are usually made of several layers of wires. Wires can cross each other if they are on separate layers. Even on a one layer circuit board, wires can cross using “jumper wires”. A jumper wire is a wire that is soldered on top of a circuit board to hop over other wires on the circuit board.

What players learn about the game:

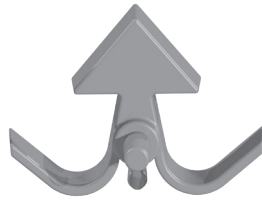
- Players don't have to make the balls go through every part on the board. Once in a while, the challenges have parts in the starting setup that do not need to be used.

Possible hang-ups:

- This challenge has parts on the starting setup that are not meant to be used. Encourage the players to think of the route that takes advantage of as many of the crossovers on the board as possible.

Computer Logic Lesson #4: Bits

In puzzle 8, the “bit” part is introduced. It looks like this:



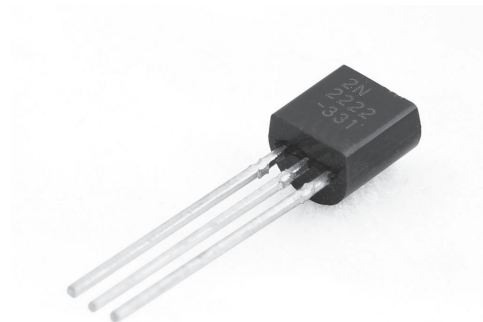
These bits are mechanical versions of the electronic switches inside computer chips. Electronic switches allow you to choose which way electricity will go based on how you set them. These mechanical switches allow you to choose which way the ball will roll off based on which direction they are pointed.

You often get to choose which direction the bits point when you start your machine: Should it point left or right? If you point the bit to the left, the next ball will fall to the right. If you point the bit to the right, the next ball will fall to the left.

You'll notice that bits in Turing Tumble are a little more tricky than electronic bits because a ball passing through them changes the direction of the bit for the next ball that falls.

What do bits look like in a regular, electronic computer?

The most basic type of electronic switch in a computer is called a “transistor”. Transistors are usually extremely tiny, but the one in the picture below is encased inside a relatively big, plastic package to make it easy to handle (it's still only about the size of a pinky fingernail):



See how there are three wires coming out? One of the wires is a control

wire. By changing the force of electricity applied to that wire (i.e., the “voltage”), it controls how much electricity can flow in through the second wire and out through the third wire.

That’s cool, but how can a switch like that store information? It turns out that if you take 4 of those transistors and connect them together in a certain way, you can create a little circuit that *remembers* if it’s been switched on or off, even when you stop pushing electricity into the control wire. That circuit is called a “flip-flop”. It’s one of the most important building blocks of a computer. Billions and billions of flip-flops are used to store information in computer memory.

What do bits look like in a programming language?

When bits are used to store information, they are like variables in a programming language. Of course, one bit doesn’t store much information, just a 1 or 0. But when you combine multiple bits into one variable, they can represent numbers, letters, or...anything else.

Bits also function as the most fundamental programming command of all: the ‘if’ statement. You could think of a single bit like this:

```
If (bit.direction = right) Then
    ball.send_left()
Else
    ball.send_right()
```


back to their original position.

Possible hang-ups:

- The symbol of the bit in the “Starting setup” shows it pointing up, but with two curved arrows over it pointing to the left and right. That indicates that you have to *choose* the starting direction the bit points.
- Encourage players to run their finger down the parts and watch how the counterweight of the ramps work in contrast to the bit turning and staying pointed in the new direction.

Challenge #9: Dimers

(page 27 in puzzle book)



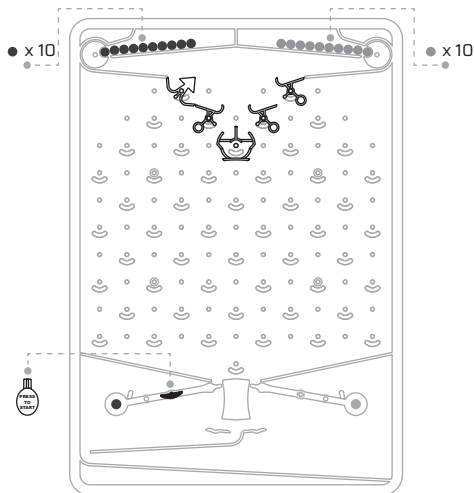
Challenge 9: Dimers

Objective: Make the pattern blue, blue, red, blue, blue, red...

Required output:



Starting setup



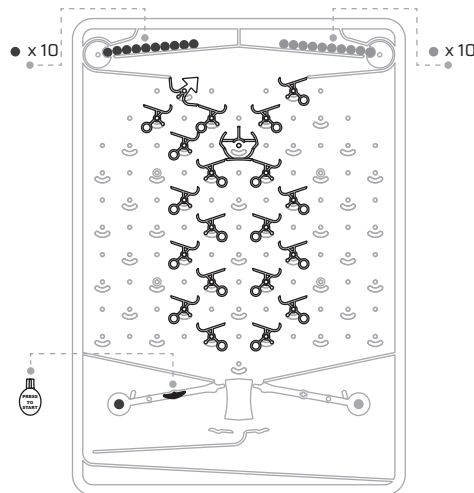
Available parts



Challenge 9 Solution

Explanation: Just like the last puzzle, the bit makes every **other** blue ball release a red ball, but every red ball releases a blue ball.

Do you see how the bit splits the path of the blue balls to go in two directions? This is very important in the coming puzzles!



What players learn about computer logic:

- The bits represent the switches inside computer chips. Switches allow you to choose where the electricity will go based on how you set them. In the game, you choose what direction the bits point: left or right. If you point the bit to the left, the ball will fall to the right. If you point the bit to the right, the ball will fall to the left. The bits allow you to route the ball one of two directions at a single point. The bits in the game are more tricky than computer chip switches because a ball passing through changes the direction of the bit for the next ball that falls.

What players learn about the game:

- The starting position of the bit is very important. In this challenge, the starting position of the bit is chosen for you: to the right. Since the bit is pointed right, the ball will fall to the left.
- When a ball goes through the bit, the bit stays pointed in the new direction.

This is in contrast to the ramps which have a counterweight that turns them back to their original position.

- The starting setup also indicates how many balls should start on top. This is the first challenge that uses ten of each instead of eight.
- The bit and crossover, working together, allow for a more complicated pattern because you are able to route the balls three different directions as they fall from the top.

Possible hang-ups:

- The symbol of the bit in the “Starting setup” shows it pointing to the right. Players do NOT get to choose the starting position of the bit in this challenge. It must be pointed right to start.
- This is the first time players are using the bit and crossover together. It might take a moment to think about the path the ball will go when the bit is pointed right (starting position) versus when it is pointing left. Encourage players to use their fingers to trace the path the balls will take.

Challenge #10: Double Bond

(page 28 in puzzle book)



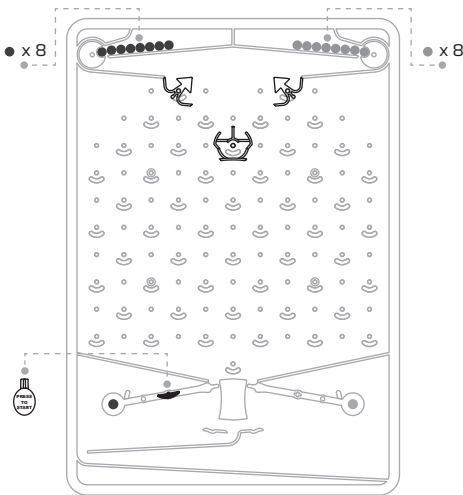
Challenge 10: Double Bond

Objective: Make the pattern blue, blue, red, red, blue, blue, red, red...

Required output:



Starting setup

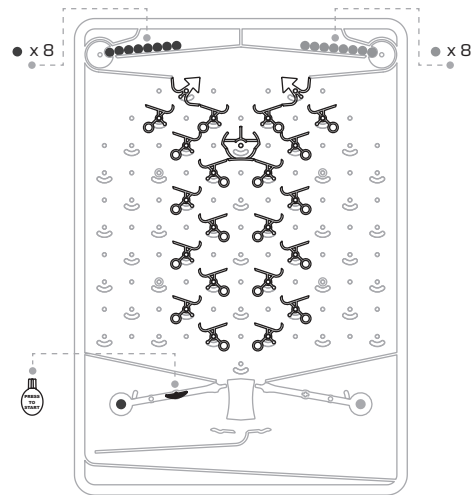


Available parts



Challenge 10 Solution

Explanation: This time, there are bits in the path of both the red and blue balls. The bits cause only every other ball to cross to the other side of the board.



What players learn about computer logic:

- The bits represent the switches inside computer chips. Switches allow you to choose where the electricity will go based on how you set them. In the game, you choose what direction the bits point: left or right. If you point the bit to the left, the ball will fall to the right. If you point the bit to the right, the ball will fall to the left. The bits allow you to route the ball one of two directions at a single point. The bits in the game are more tricky than computer chip switches because a ball passing through changes the direction of the bit for the next ball that falls.

What players learn about the game:

- The starting position of the bit is very important. In this challenge, the starting positions of the bits are chosen for you.
- When a ball goes through the bit, the bit stays pointed in the new direction. This is in contrast to the ramps which have a counterweight that turns them

back to their original position.

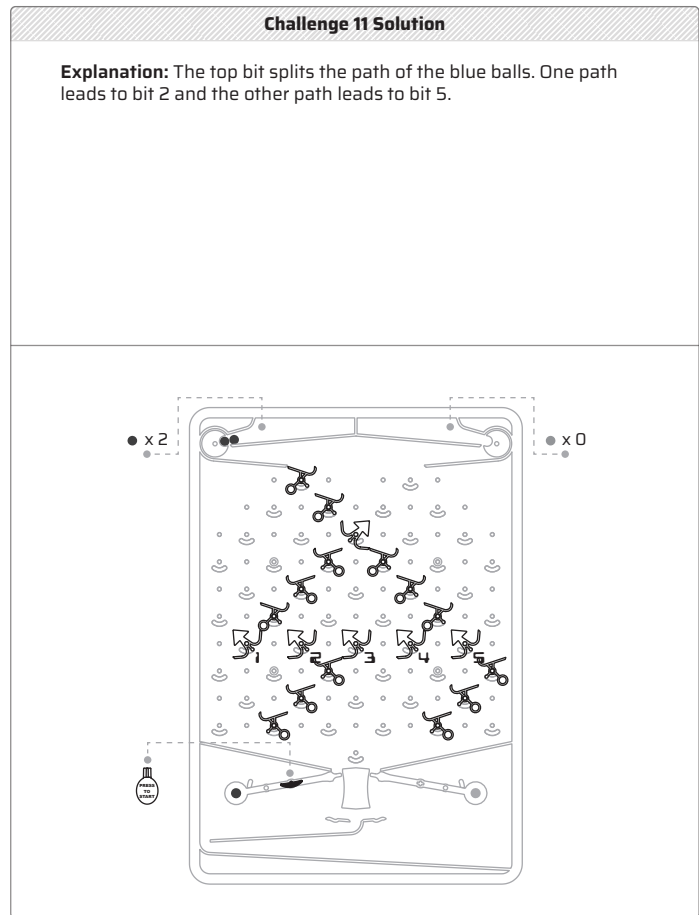
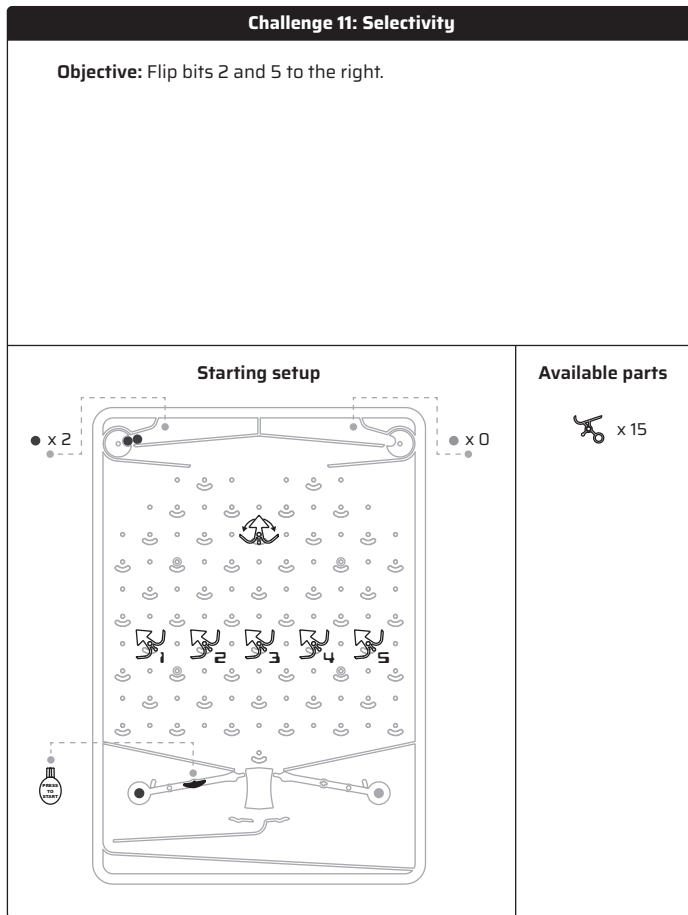
- The starting setup also indicates how many balls should start on top. This challenge has you go back to using eight of each.
- Two bits allow for a more complicated pattern because you are able to route the balls four different directions as they fall from the top.

Possible hang-ups:

- The symbols of the bit in the “Starting setup” shows one pointing right and one pointing left. Players do NOT get to choose the starting position of the bit in this challenge.
- While you have four paths for balls at the top, you only need two paths once you get to the bottom: one path to trigger the left (blue) lever and one path to trigger the right (red) lever.

Challenge #11: Selectivity

(page 29 in puzzle book)



What players learn about computer logic:

- The bits represent the switches inside computer chips. Switches allow you to choose where the electricity will go based on how you set them. In the game, you choose what direction the bits point: left or right. If you point the bit to the left, the ball will fall to the right. If you point the bit to the right, the ball will fall to the left. The bits allow you to route the ball one of two directions at a single point. The bits in the game are more tricky than computer chip switches because a ball passing through changes the direction of the bit for the next ball that falls.

What players learn about the game:

- The starting setup also indicates how many balls should start on top. This challenge has players only using two blue balls.
- The starting position of the bit is very important. If a bit is pointed left, the ball will fall right. If the bit is pointed right, the ball will fall to the left. In this

challenge, players choose the starting position of the top bit. The starting position of bits one through five is left.

- Players don't have to make the balls go through every part on the board.

Possible hang-ups:

- The starting setup also indicates how many balls should start on top. This challenge has players only using two blue balls.
- This challenge has parts in the starting setup that balls will never go through.

Computer Logic Lesson #5: Interceptors

In puzzle 12, the “interceptor” part is introduced. It looks like this:



The purpose of the interceptor is to stop the computer from running. When a ball falls into it, the computer stops because the ball is stuck. It can't reach the bottom to push down a lever and keep it running.

What do interceptors look like in a regular, electronic computer?

In modern computers, the closest analogy to an interceptor would be an automatic shutoff switch in a computer. When you tell your computer to shut down, it completes running its tasks and turns itself off by an electronic switch that cuts the power to the computer.

In Turing Tumble, you'll find that you often use the interceptor to stop the computer from running every time it completes a program. Bigger, electronic computers also stop running a program when it is complete, but the computer continues to run afterward. Instead of shutting down the whole computer when the program completes, the computer simply moves on to run other programs (and the operating system) instead.

What do interceptors look like in a programming language?

You could think of the interceptor, most directly, as a command that automatically shuts down the computer. But you rarely do that sort of thing when you're writing code. Instead, you could think of the interceptor as a command that ends the program:

End Program



Challenge #12: Duality - Part 1

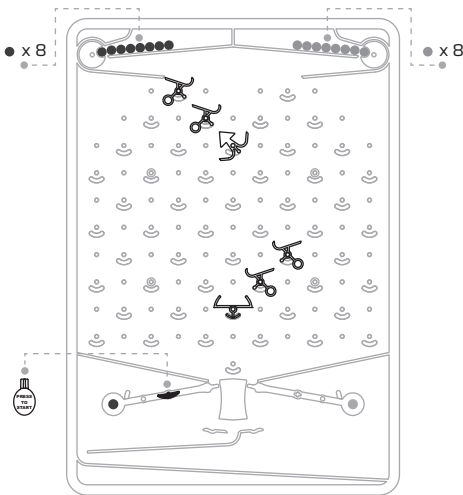
(page 32 in puzzle book)



Challenge 12: Duality - Part 1

Objective: Intercept a blue ball.

Starting setup

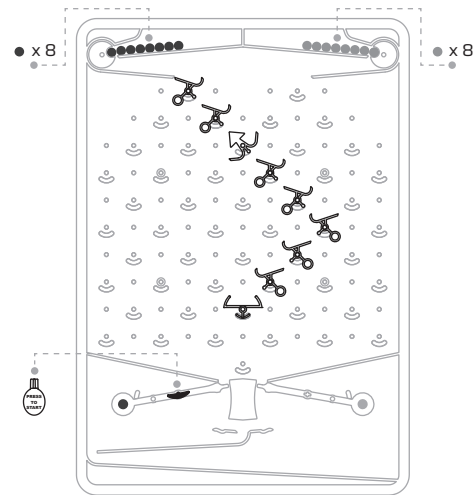


Available parts



Challenge 12 Solution

Explanation: The ramps must be used to complete the path of a single ball to the interceptor.



What players learn about computer logic:

- The interceptor represents an “End Program” command. While some circuit boards might actually be able turn themselves off from code in a program (usually battery powered devices do), usually programs end themselves and the processor just switches over to running other code.

What players learn about the game:

- Players learn how to use an interceptor to stop the machine. When an interceptor is added to the board at the right spot, a ball is caught, making it so that the bottom levers are not triggered again and no more balls are released.

Possible hang-ups:

- Players might overthink this one. It is a really simple solution that prepares them for more complicated challenges to come.

Challenge #13: Duality - Part 2

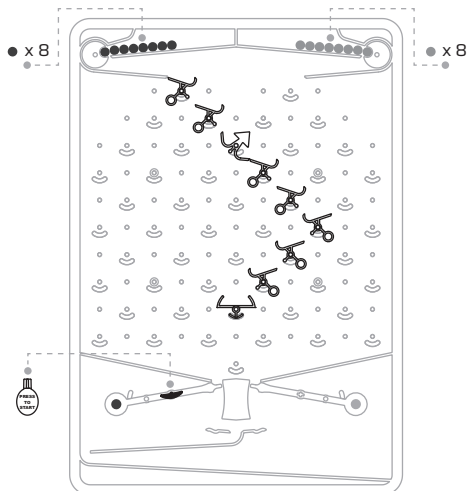
(page 33 in puzzle book)



Challenge 13: Duality - Part 2

Objective: Intercept a red ball.

Starting setup



Available parts

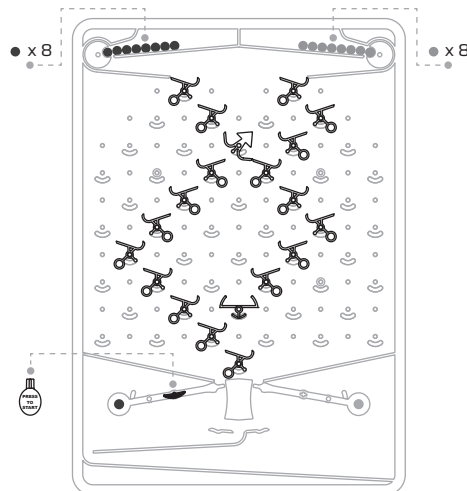


x 12

Challenge 13 Solution

Explanation: Even though you start by releasing a blue ball, you've got to get a red ball in the interceptor. How do you do that?

Here's how: Lead the first blue ball to the right lever, releasing a red ball. Then lead the red ball to the interceptor with ramps.



What players learn about computer logic:

- The interceptor represents an “End Program” command. While some circuit boards might actually be able turn themselves off from code in a program (usually battery powered devices do), usually programs end themselves and the processor just switches over to running other code.

What players learn about the game:

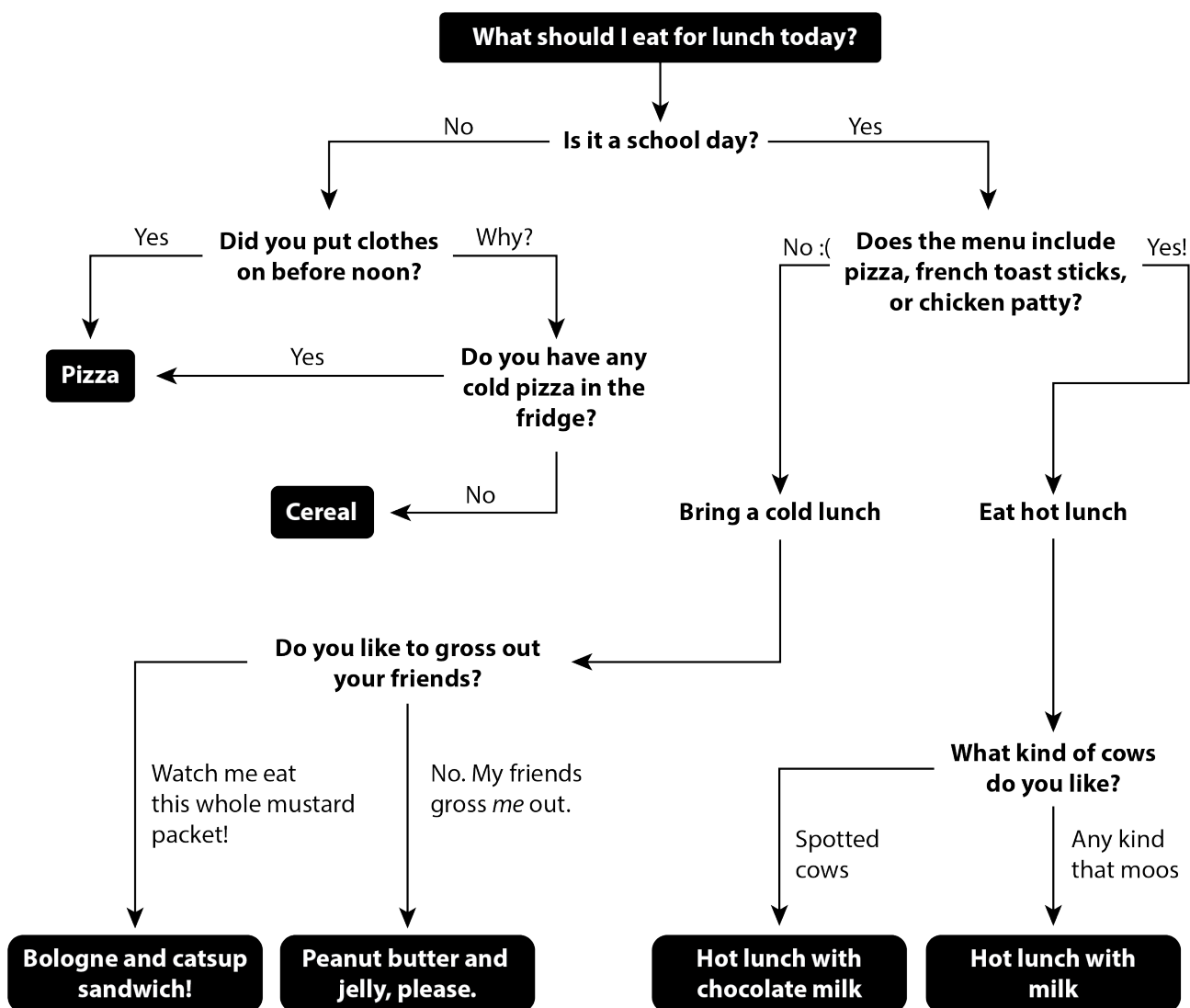
- The starting setup tells you to start the machine by pressing down the left lever (releasing a blue ball), but then a red ball must be caught in the interceptor. This is the first puzzle where players make a direct path from the blue side all the way over to the right flipper.

Possible hang-ups:

- The starting setup tells you to start the machine by pressing down the left lever (releasing a blue ball), but then a red ball must be caught in the interceptor. Encourage players to make a path for the blue ball to cross over and drop down on the right side, releasing a red ball. Then the red ball must be directed to the interceptor.
- Players do not need to use all 12 available ramp parts to solve this puzzle.

Computer Logic Lesson #6: Conditional Statements

In puzzle 14, “conditional statements” are introduced (often called “if-then statements”). Conditional statements are fundamental to computer programming. They’re fundamental in life as well! For example, you might say to a student: IF you turn in all your homework, THEN you will get at least a B in this class. Computer programs are full of these conditional statements. In fact, programmers often write their code as a flow chart before putting it into computer language. Take, for instance, the following flow chart that you can use to choose what to eat for lunch:

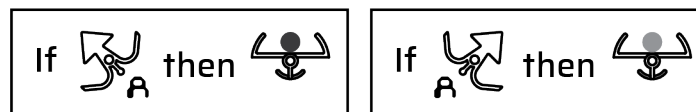


When a computer program runs, it zips through the conditions quickly, doing different things depending on whether the conditions are true or

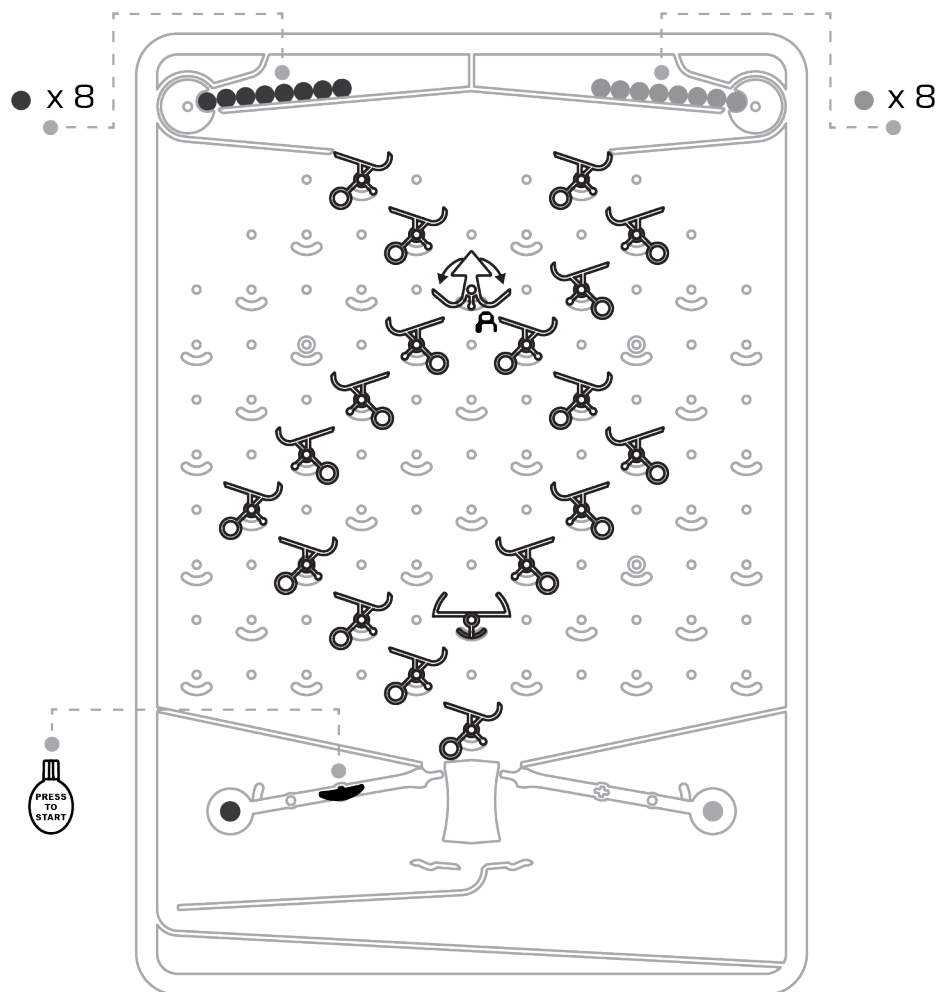
false.

In the following challenge, the conditional statement is posed: “If the machine starts with bit A pointing to the left, then intercept a blue ball. If the machine starts with bit A pointing to the right, then intercept a red ball.” Players must consider the two starting conditions when they are building the machine.

Puzzles like these can be a little confusing at first. Students sometimes have trouble understanding that the goal is to build *one* machine that works for *all* starting conditions. For example, the goal of the next puzzle is to build a machine that works for the following two conditions:

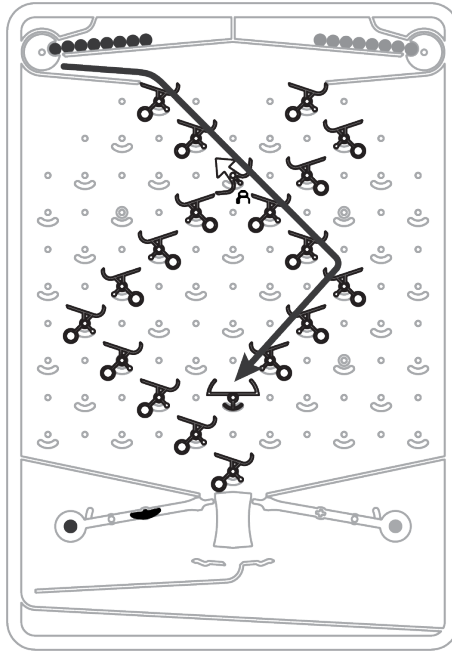


And here's a solution:

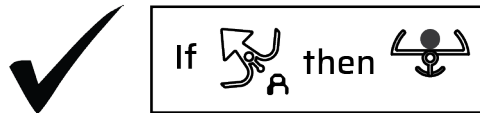


How do you check to see if this solution is correct? You have to run the machine under each of the two starting conditions. You start it once with

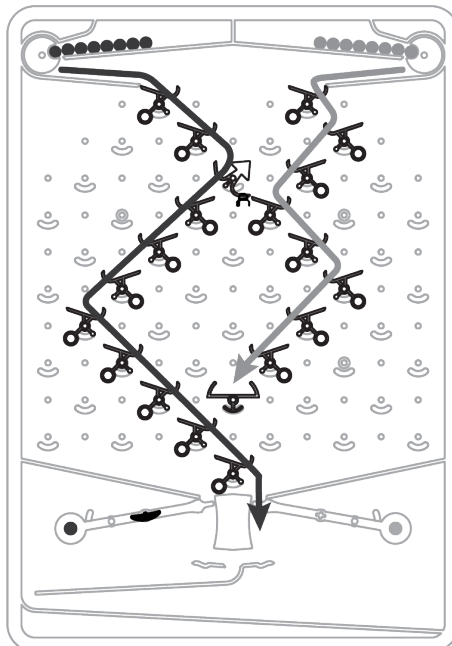
bit A pointed to the left, like this:



When you press down the left lever, a blue ball lands in the interceptor, meeting the requirement of the first condition above.



Now for the second condition. To test whether it works for the second condition, we start the machine with bit A pointing to the *right*. Like this:

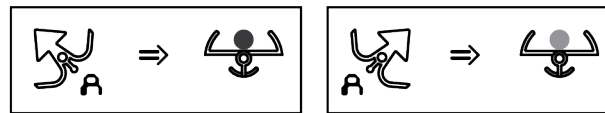


When you start the machine by pressing down the left lever, a red ball lands in the interceptor, meeting the requirement of the second condition above.

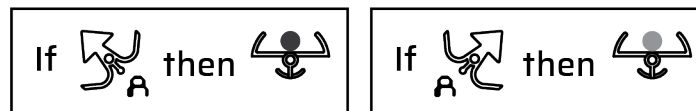


That's it! Since it did what it was supposed to do for each starting condition, you know you have a correct solution. Puzzle complete!

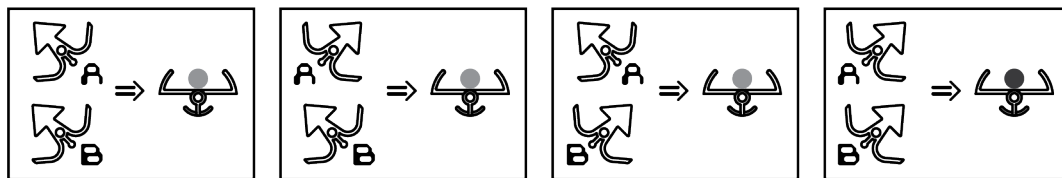
Starting in puzzle 15, we use a symbol in place of the “if” and “then” words. The symbol looks like this: “ \Rightarrow ” It means “if” whatever’s on the left, “then” whatever’s on the right. For example, this:



means the same thing as this:



As the puzzles progress, sometimes there are more than two conditions to check. For example, in puzzle 19, there are actually four conditions you have to test to make sure the machine works correctly:



If you'd like an additional, fun resource to explain how conditionals work in coding, the following video is a great resource:

<https://www.flocabulary.com/unit/coding-conditionals/>

Challenge #14: Duality - Part 3

(page 34 in puzzle book)



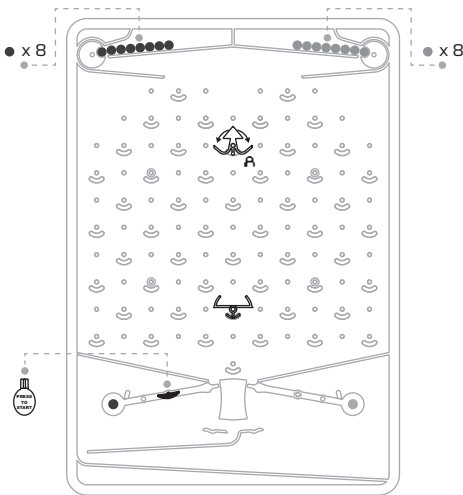
Challenge 14: Duality - Part 3

Objective: If the machine starts with bit A pointing to the left, intercept a blue ball. Otherwise, intercept a red ball.

Examples:



Starting setup



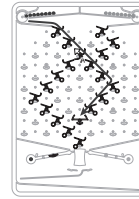
Available parts



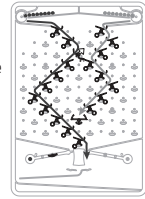
Challenge 14 Solution

Explanation: Let's think about each of the two starting conditions separately.

A starts left:
Like puzzle 12, the first blue ball simply falls into the interceptor.



A starts right:
Like puzzle 13, the first blue ball falls onto the lever on the right side of the board, releasing a red ball that lands in the interceptor.



What players learn about computer logic:

- This challenge introduces conditional statements (often called IF-THEN statements) which are fundamental in computer programming. Conditional statements are fundamental in life as well. You might say to a student: If you turn in all your homework, then you will get at least a B in this class. Computer programs are full of these conditional statements, in fact programmers often write their code as a flow chart before putting it into computer language. When a program runs, it zips through the conditions quickly, executing different options depending on whether the condition is true or false.
- In this challenge, the conditional statement is posed: "If the machine starts with bit A pointing to the left, then intercept a blue ball. If the machine starts with bit A pointing to the right, then intercept a red ball." Players must consider the two starting conditions when they are building the machine.

What players learn about the game:

- The starting position of the bit is very important. If a bit is pointed left, the ball will fall right. If the bit is pointed right, the ball will fall to the left. In this challenge, players choose the starting position of the bit.
- This is the first time players must build one machine that works for more than one starting condition. This will require some thoughtful planning and testing, and players will have to test it with both starting conditions to be sure it works.
- Reading the talk bubbles can help get you on the right path to solving the challenge.

Possible hang-ups:

- The red ball does not have to pass through the bit. In order for a red ball to fall, a blue ball would have already tested its direction.
- It only matters what direction the bit points to begin! The first ball going through the bit tests the starting direction and that is the only time the direction matters.

Challenge #15: Inversion

(page 35 in puzzle book)



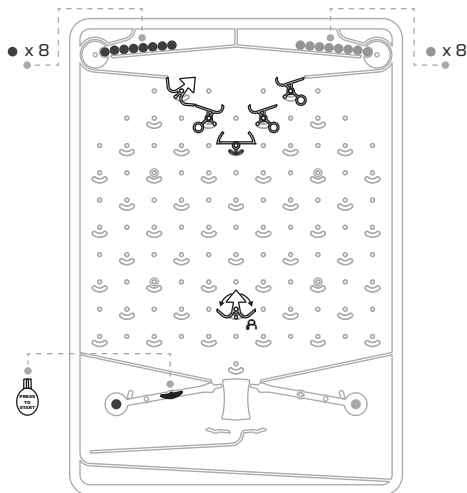
Challenge 15: Inversion

Objective: If bit A starts to the left, intercept a blue ball. If bit A starts to the right, intercept a red ball.

Examples:



Starting setup

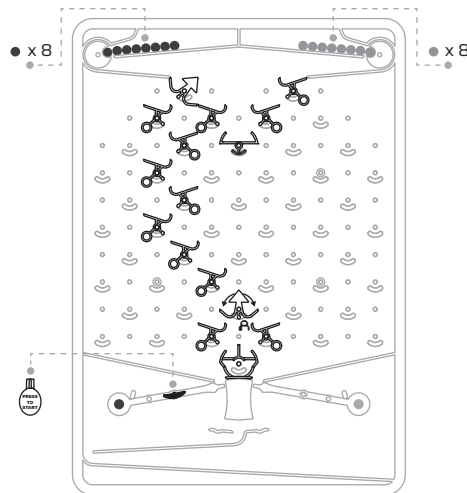


Available parts



Challenge 15 Solution

Explanation: Like the previous puzzle, the bit at the bottom of the board decides the color of the ball that must go in the interceptor. But for this puzzle, the colors are reversed! Here we do that with a carefully placed crossover.



What players learn about computer logic:

- This challenge introduces conditional statements (often called IF-THEN statements) which are fundamental in computer programming.
- ‘=>’ symbol: “if” whatever’s on the left, “then” whatever’s on the right, as noted in the thought bubble.
- Conditional statements are fundamental in life as well. You might say to a student: If you turn in all your homework, then you will get at least a B in this class. Computer programs are full of these conditional statements, in fact programmers often write their code as a flow chart before putting it into computer language. When a program runs, it zips through the conditions quickly, executing different options depending on whether the condition is true or false.
- In this challenge, the conditional statement is posed using the ‘=>’ symbol. “If bit A starts to the left, then intercept a blue ball. If bit A starts to the right, then intercept a red ball.” Players must consider the two starting conditions

when they are building the machine.

What players learn about the game:

- The objective on this puzzle is the same as the last one, as noted in the thought bubble.
- The starting position of the bit is very important. If a bit is pointed left, the ball will fall right. If the bit is pointed right, the ball will fall to the left. In this challenge, the top bit begins pointing to the right, but players must choose the starting position of the bottom bit “A”.
- This machine must work for more than one starting condition. This will require some thoughtful planning and testing, and players will have to test it with both starting conditions to be sure it works.
- Reading the talk bubbles can help get you on the right path to solving the challenge.

Possible hang-ups:

- Note that there is a practice guide with additional puzzles to help players practice the skills learned so far. You can download it for free at edu.turingtumble.com. The practice challenges will prepare them for challenge 16 and beyond.
- The red ball does not have to pass through the bit. In order for a red ball to fall, a blue ball would have already tested its direction.
- It only matters what direction the bit points to begin! The first ball going through the bit tests the starting direction and that is the only time the direction matters.

Challenge #16: Termination


(page 36 in puzzle book)



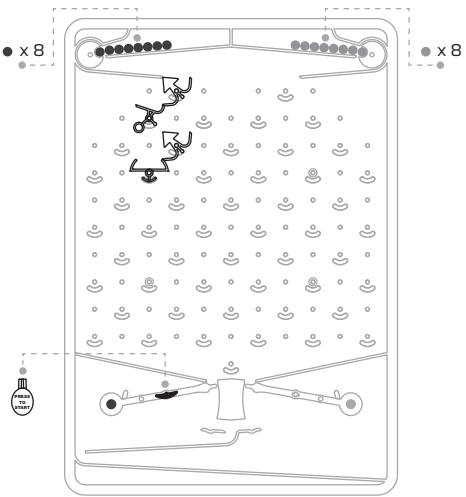
Challenge 16: Termination

Objective: Let only 3 blue balls reach the bottom and catch the 4th ball in the interceptor.


Required output:



Starting setup



Available parts

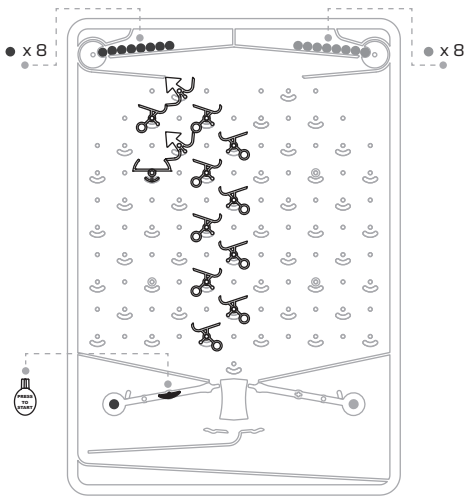


Challenge 16 Solution

Explanation: Like the previous puzzle, think about the direction the bits point when each ball is released:

| 1st ball | 2nd ball | 3rd ball | 4th ball |
|----------|----------|----------|----------|
| | | | |
| | | | |

The way these bits count out balls is important! You'll use it many times in the coming puzzles.



What players learn about computer logic:

- This challenge shows how bits can be used for more than running conditional tests. The direction the bits point actually set the “state” of the board. Every time a ball passes through a bit, it changes the direction the bit points and the “state” of the board for the next ball. Just as you would think multiple steps ahead when coding, you must think through how the bits change each time and how that affects the outcome.
- In this puzzle, students are exposed, just a little, to how Turing Tumble can count, but they don't learn about binary until challenge 21.

What players learn about the game:

- The starting position of the bit is very important. In this challenge, the top and bottom bits begin pointed to the left.
- Every time the ball goes down, it goes through the top bit, flipping it to the left. It only goes through the bottom bit every other time.

- You must think ahead multiple steps in order to solve this challenge.

Possible hang-ups:

- To catch the fourth ball, it takes two balls to flip the bottom bit to the right, and then two more balls to flip it so that the fourth ball is caught in the interceptor. You can think of it this way:
 - Ball 1: Flips top bit to the right
 - Ball 2: Flips top bit to the left and bottom bit right
 - Ball 3: Flips top bit to the right
 - Ball 4: Flips top bit to the left and bottom bit left
- It might be helpful to jot down some notes about where each ball travels as it goes through bits and down the board.

Challenge #17: Fixed Ratio

(page 37 in puzzle book)



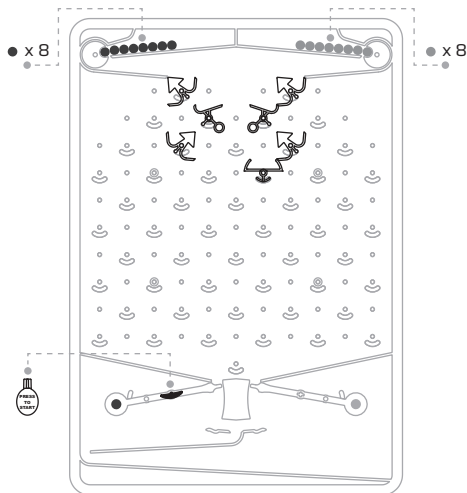
Challenge 17: Fixed Ratio

Objective: Make the pattern blue, blue, blue, red, red, red.

Required output:



Starting setup

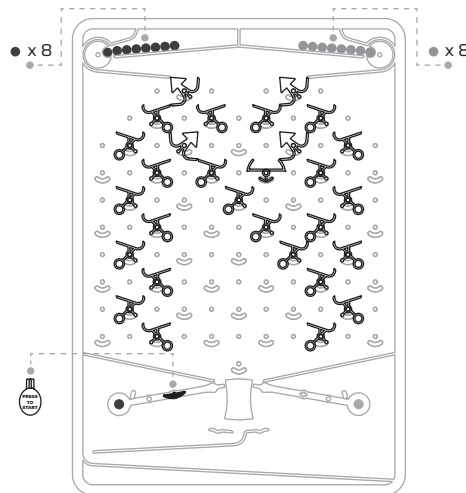


Available parts



Challenge 17 Solution

Explanation: Now we have two 2-bit counters - one on the left and one on the right. Notice the starting positions of the bits! To get three blues and three reds in the output, we have to count out 3 blue balls, and 4 red balls (the 4th ball goes into the interceptor!).



What players learn about computer logic:

- In this puzzle, students are exposed, just a little, to how Turing Tumble can count, but they don't learn about binary until challenge 21.
- In this challenge, you will set up the board so that it does something for a fixed number of times, and then it moves onto something else in the program. Specifically it will run through balls on the left side of the board until three balls have passed through. Then it will start sending balls down the right side. That is similar to what a loop does in coding. It executes a block of code a number of times before moving on to the next line of code.

What players learn about the game:

- The starting position of the bit is very important. In this challenge, the top two bits point to the left. The bottom bits point to the right (left bit) and to the left (right bit).
- You can use as many ramps as you want to solve this challenge.

- Players will learn that you can count on the left side and then instigate something else to happen on the right side.

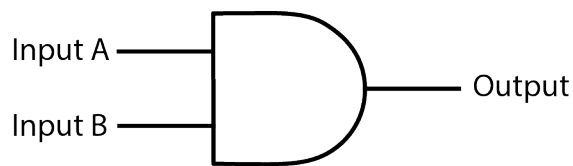
Possible hang-ups:

- It might be hard to know what to do when it finishes counting out three balls. Guide the players to lead the third ball over to the red side to trigger the lever on the right side.
- Players might not pay close attention to the starting setup as they test their solutions. If they think they have the solution or can't get it to work consistently, encourage them to check the starting setup each time, before pushing down the start button.
- It might be helpful to jot down some notes about where each ball travels as it goes through bits and down the board.

Computer Logic Lesson #7: Logic Gates

The fundamental unit of computers is the switch, but if you're designing something as complicated as a computer processor, you don't think much about the individual switches. Instead, you think about groupings of switches. Sort of like when you build a brick wall, you don't think about the individual rocks inside of bricks, you think about the bricks and how to fit them together to make a bigger structure.

Logic gates are little groupings of switches that behave in a certain way. For example, one type of logic gate is the AND gate. The symbol for an AND gate looks like a 'D' with two lines coming in from the left and one line leaving the right.

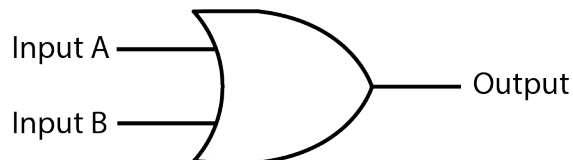


AND gate

The two lines coming in on the left are the inputs. Each input can either be 'true' or 'false' (on or off, 1 or 0). The output comes out the right side. In an AND gate, the output is 'true' only if input A is 'true' *and* input B is 'true'. Otherwise, the output is 'false'.

For a real world example, imagine this household rule: "If you completed your after school chores *AND* you ate your vegetables at dinner, then you can eat dessert." If one or the other responsibility didn't happen, you don't get dessert. In this situation, a parent takes the two inputs, compares them, and then outputs the result: either dessert, or no dessert.

Another type of logic gate is the OR gate. When you draw the symbol for an OR gate, it looks like this:



OR gate

An OR gate also has two inputs that can be 'true' or 'false' and one output. In an OR gate, the output is 'true' if input A is 'true' *or* input B is 'true'.

Otherwise, the output is 'false'.

We'll see more logic gates in the next puzzles and lessons. For now, let's take a look at the next challenge (challenge 18). In this challenge, players learn to build an AND gate. There are two input bits on the board (bit A and bit B). The output is given by which of the two interceptors a ball lands in; a ball could land in interceptor T (true) or interceptor F (false). Your goal is to build a machine that drops a ball in interceptor T only if bit A *and* bit B are pointed to the right. Otherwise, your machine must drop a ball in interceptor F.

Logic "operations" are also frequently coded into conditional statements in programs. For example, you might code challenge 18 like this:

```
If (bit_a.direction = right AND bit_b.direction = right) Then
    interceptor_t.add_ball()
Else
    interceptor_f.add_ball()
```

Here, "AND" is the logic "operation". It checks the condition to its left and the condition to its right. If they're both true, then whole conditional statement is true and it runs the line of code right underneath the "If" statement. But if either one is false, then the whole conditional statement is false and it runs the line of code under "Else".

Challenge #18: Entanglement

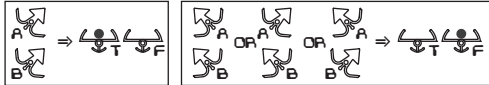
(page 38 in puzzle book)



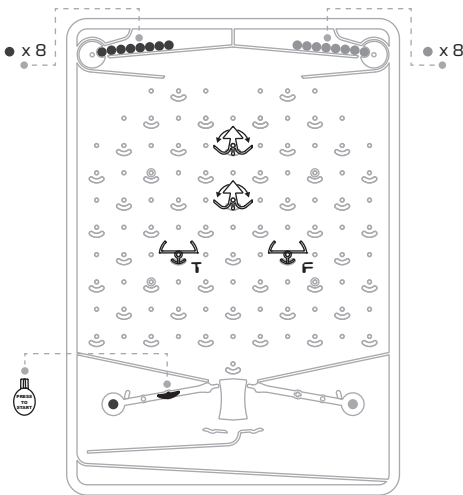
Challenge 18: Entanglement

Objective: If the top bit AND the bottom bit start pointed to the right, put a ball in interceptor T. Otherwise put a ball in interceptor F.

Examples:



Starting setup



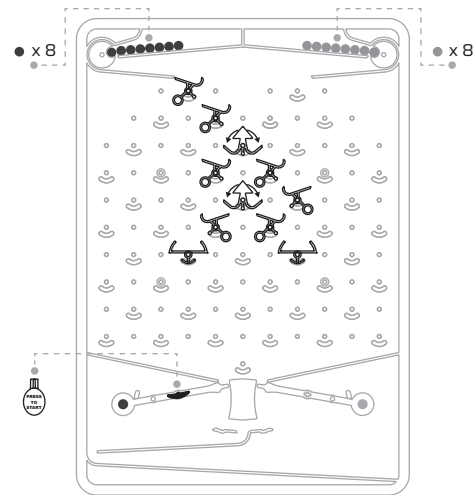
Available parts



x 7

Challenge 18 Solution

Explanation: This puzzle is the same as the last puzzle, except that the path the ball must take in each case is a little different.



What players learn about computer logic:

- In this challenge, there are two starting positions and an output. The two starting positions are the directions players point the bits. The output is which interceptor the ball falls into. The ball could fall into interceptor T (True) or interceptor F (False). When solving the challenge, you must work under the condition that if both bits are pointed to the right to start, the ball must go into interceptor T. If either bit is pointed to the left, the ball goes into interceptor F. This introduces the concept of logic gates, specifically an AND gate.
- In a processor chip, there are many, many logic gates. A logic gate is created by tiny transistors connected together in certain ways so that the chip can receive electrical signals, compare them, and then send out a new electrical signal depending on what it finds (and what type of logic gate it is).
- Logic gates can also be coded into a computer program. The computer

takes two inputs, compares them, and then outputs the result. In coding you would write “if _____ AND _____ , then_____ .”

- For a real world example, imagine this household rule: “If you completed your after school chores AND you ate your vegetables at dinner, then you can eat dessert.” If one or the other responsibility didn’t happen, you don’t get dessert. In this situation, a parent takes the two inputs, compares them, and then outputs the result: either dessert, or no dessert.
- This challenge also introduces truth tables which will be explained in Challenge 20 and the next Computer Logic Lesson.

What players learn about the game:

- Players learn that bits and interceptors are labeled, and that those labels are important for understanding the directions.
- The conditional symbol ‘=>’ is used to show examples of the written directions. ‘=>’ means ‘if’ whatever’s on the left, ‘then’ whatever’s on the right.
- NO balls should reach the bottom. The starting position of the bits can be tested by just releasing one blue ball at a time.
- NO red balls should be released.

Possible hang-ups:

- If the top bit is pointed to the left, it does not need to go through the second bit. Players can build a path directly to interceptor F (False).
- If the top bit is pointed to the right, it must pass through the second bit so that it can test its direction.
- Once the puzzle is solved, players should test all the possible starting positions listed under “examples” so that they know for sure they have solved it.

Computer Logic Lesson #8: Truth Tables

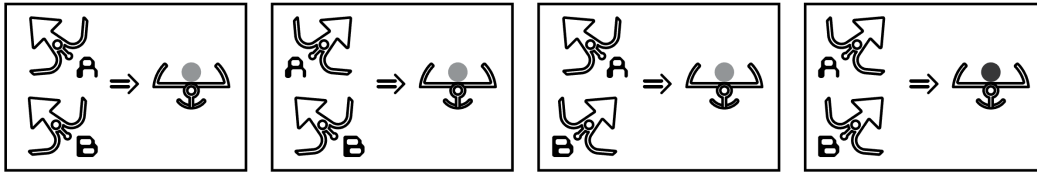
In the challenges coming up, the challenges often have examples underneath the objective. In each box, they show various starting setups and the outcomes they must generate. The starting positions of the bits are the input (on the left side of the '=>' symbol), and the color of the ball that falls into the interceptor is the output (on the right side of the '=>' symbol).

You'll notice the game uses some conventions throughout the puzzles. For example, a bit pointed right is always represents true, whereas a bit pointed left always represents false. Also, in the output, a blue ball always represents true and a red ball always represents false.

Example from the puzzle book:

Objective: If the top bit AND the bottom bit start pointed to the right, intercept a blue ball. Otherwise, intercept a red ball.

Examples:



In math and computer science, the organizational tool that is used to communicate this type of information is a “truth table”. Truth tables are a convenient and clear way to show the way a logic circuit behaves. The input combinations go on the left and the output goes on the right. The inputs and outputs can be written as ‘True’ or ‘False’ or as ‘1’s and ‘0’s.

Truth Table for an AND gate as True and False

| A | B | A AND B |
|-------|-------|---------|
| False | False | False |
| True | False | False |
| False | True | False |
| True | True | True |

Truth Table for an AND gate as 1 and 0

| A | B | A AND B |
|---|---|---------|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

If you were to write the objective for challenge 19 as a Truth Table, this is what you would get:

Truth Table for Challenge 19

| A | B | A AND B |
|-----------|-----------|------------------|
| False (👉) | False (👉) | False (red ball) |
| True (👈) | False (👉) | False (red ball) |
| False (👉) | True (👈) | False (red ball) |
| True (👈) | True (👈) | True (blue ball) |

Truth tables are an especially good way to represent the behavior of all of the logic gates, including AND gates and OR gates (which were discussed in the previous computer logic lesson), as well as NAND gates, NOT gates, and XOR gates, to name a few.

In Challenge 20, students will be building an OR gate. If the top bit (A) OR bottom bit (B) are pointed to the right to start, a blue ball must fall into the interceptor. Otherwise a red ball must fall into the interceptor. When put into a truth table, this OR gate looks like this:

Truth Table for Challenge 20

| A | B | A OR B |
|-----------|-----------|------------------|
| False (👉) | False (👉) | False (red ball) |
| True (👈) | False (👉) | True (blue ball) |
| False (👉) | True (👈) | True (blue ball) |
| True (👈) | True (👈) | True (blue ball) |

Players will encounter other challenges involving logic gates further along in the puzzle book:

- Challenge 34 is an XOR gate
- Challenge 36 is another AND gate
- Challenge 43 is another XOR gate

Stretching Activity:

Once students have worked through challenges 19 and 20, they can be challenged to build some new gates on their own, based on what they have learned. You can vary the difficulty of this activity by giving them the objective, asking them to write their own objective, giving them the truth table, or having them make their own.

Stretching Challenge #1: Make a NAND gate

A NAND gate is the opposite of an AND gate. You would say the output is false if both inputs are true, otherwise the output is true. “Not A and B” - so everything is true except when A and B are true.

Real World Example: Imagine you have a display of fine jewelry at a museum, secured in a glass case with an alarm system that sounds if it is jostled or tampered with. Once the alarm is triggered, two museum security guards must work together to silence it after they have assessed the situation. There is a keypad that requires a code and six feet away there is a switch that must be turned with a special key. The alarm will keep ringing until the code and switch are turned at the same time. These two “true” actions result in the alarm turning off or “false”. If nothing is done, the alarm will keep sounding. If only the key is turned, the alarm will keep sounding. If only the code is entered, the alarm will keep sounding.

Objective: If top bit (A) and bottom bit (B) are pointed right to start, then a red ball falls into the interceptor. Otherwise a blue ball falls into the interceptor.

Truth Table for a NAND Gate

| A | B | A OR B |
|-----------|-----------|------------------|
| False (👉) | False (👉) | True (blue ball) |
| True (👈) | False (👉) | True (blue ball) |
| False (👉) | True (👈) | True (blue ball) |
| True (👈) | True (👈) | False (red ball) |

Stretching Challenge #2: Make a NOR gate

A NOR gate is the opposite of an OR gate. It is only true if the inputs are “not A or B”.

Real World Example: Imagine you went into the doctor on Friday to find out that you are sick with strep throat. You don't want to miss school on Monday because your class is going on a field trip. If you have neither a fever nor a sore throat by Sunday at 9am, you can go on your trip. If you have a fever and a sore throat, you can't go. If you have a fever but not a sore throat, you can't go. If you don't have a fever but you still have sore throat, you can't go.

Objective: If neither the top bit (A) or the bottom bit (B) are pointed to the right to start, a blue ball falls into the interceptor. Otherwise a red ball falls into the interceptor.

Truth Table for a NOR Gate

| A | B | A OR B |
|-----------|-----------|------------------|
| False (👉) | False (👉) | True (blue ball) |
| True (👈) | False (👉) | False (red ball) |
| False (👉) | True (👈) | False (red ball) |
| True (👈) | True (👈) | False (red ball) |

Challenge #19: Entanglement

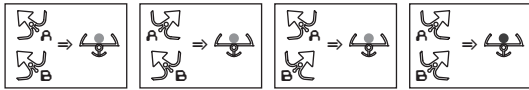
(page 39 in puzzle book)



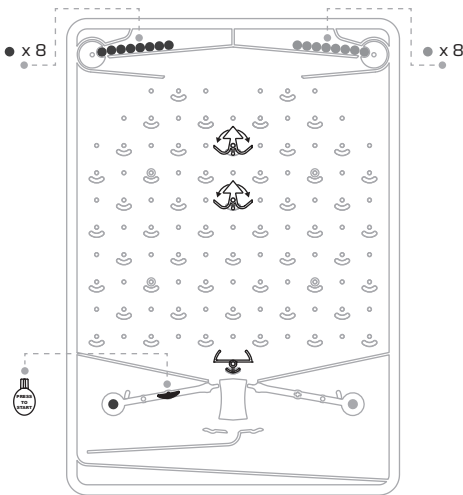
Challenge 19: Entanglement

Objective: If the top bit AND the bottom bit start pointed to the right, intercept a blue ball. Otherwise, intercept a red ball.

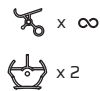
Examples:



Starting setup

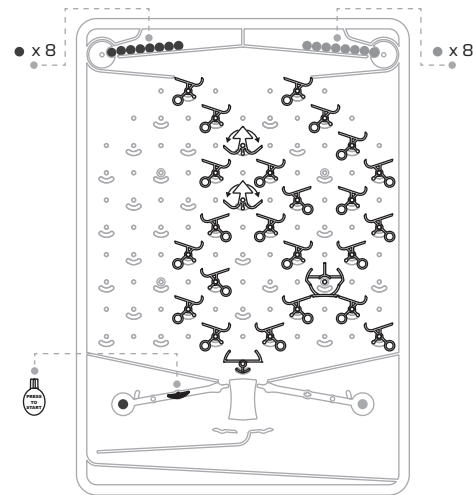


Available parts



Challenge 19 Solution

Explanation: This time there is only one interceptor. Instead of sending the ball to interceptor F if the bits aren't both pointed right, you must direct the ball to the right lever. Then direct the red ball that's released to the interceptor.



What players learn about computer logic:

- In this challenge, there are two inputs and an output. The two inputs are the starting directions players point the two bits. The output is the color ball that falls into the interceptor. If the top bit (A) AND bottom bit (B) are pointed to the right to start, a blue ball falls into the interceptor. Otherwise a red ball falls into the interceptor. This is another example of a logic gate, specifically an AND gate.
- In a processor chip, there are many, many logic gates. A logic gate is created by tiny transistors connected together in a certain way so that the chip can receive two electrical signals, compare them, and then send out a new electrical signal depending on what it finds.
- Logic gates can also be coded into a computer program. The computer takes two inputs, compares them, and then outputs the result. In coding you would write "if _____ AND _____, then _____."
- For a real world example, imagine this household rule: "If you completed

your after school chores AND you ate your vegetables at dinner, then you can eat dessert.” If one or the other responsibility didn’t happen, you don’t get dessert. In this situation, a parent takes the two inputs, compares them, and then outputs the result: either dessert, or no dessert.

- If a bit is pointing right, it is true. If it is pointing left, it is false.

Truth Table for Challenge 19

| A | B | A AND B |
|-----------|-----------|------------------|
| False (↩) | False (↩) | False (red ball) |
| True (↪) | False (↩) | False (red ball) |
| False (↩) | True (↪) | False (red ball) |
| True (↪) | True (↪) | True (blue ball) |

What players learn about the game:

- Players learn that bits are labeled, and that those labels are important for understanding the directions.
- The conditional symbol “=>” is used to show examples of the written directions. “=>” means “if _____, then_____.”

Possible hang-ups:

- If the top bit is pointed to the left, it does not need to go through the second bit. Players can build a path directly to the bottom to trigger a red ball.
- If the top bit is pointed to the right, it must pass through the second bit so that it can test its direction.
- Once it triggers a red ball, that ball can go directly into the interceptor without passing through the bits.
- Players don’t have to use both crossovers to solve the challenge.

Challenge #20: Symbiosis

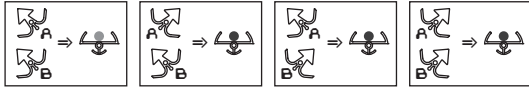
(page 40 in puzzle book)



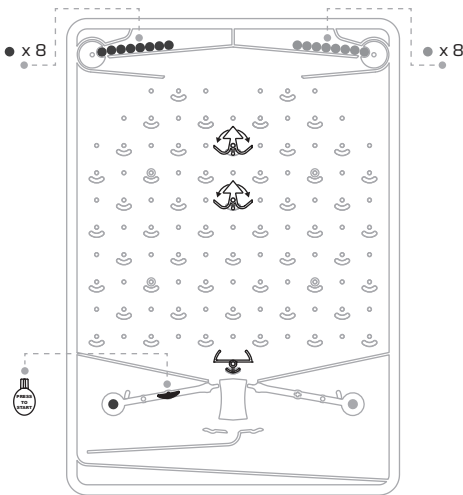
Challenge 20: Symbiosis

Objective: If the top bit OR the bottom bit start pointed to the right, intercept a blue ball. Otherwise, intercept a red ball.

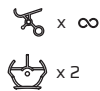
Examples:



Starting setup

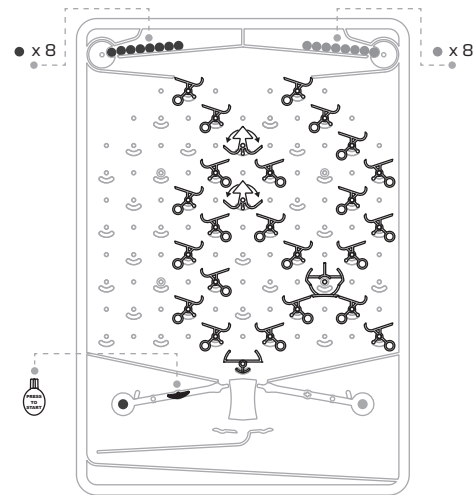


Available parts



Challenge 20 Solution

Explanation: It's easiest to consider the first condition - when both bits are pointed left. Make that path go directly to the red lever and make a path for the red ball to go directly to the interceptor. Then make every other possible path for that first blue ball lead to the interceptor.



What players learn about computer logic:

- Truth Tables are a convenient way of representing the behavior of logic gates (see the previous computer logic lesson).
- In this challenge, there are two inputs and an output. The two inputs are the starting directions players point the two bits. The output is the color of the ball that falls into the interceptor. If the top bit (A) OR bottom bit (B) are pointed to the right to start, a blue ball must fall into the interceptor. Otherwise a red ball must fall into the interceptor. This is an example of a logic gate called an 'OR' gate.
- When written as a Truth Table, the input and output are usually written as 'true' or 'false'. In this game, a bit pointed right is always represents true, a bit pointed left is always represents false. In the output, the blue ball always represents true, and the red ball always represents false:

Truth Table for Challenge 20

| A | B | A OR B |
|-----------|-----------|------------------|
| False (👉) | False (👉) | False (red ball) |
| True (👈) | False (👉) | True (blue ball) |
| False (👉) | True (👈) | True (blue ball) |
| True (👈) | True (👈) | True (blue ball) |

What players learn about the game:

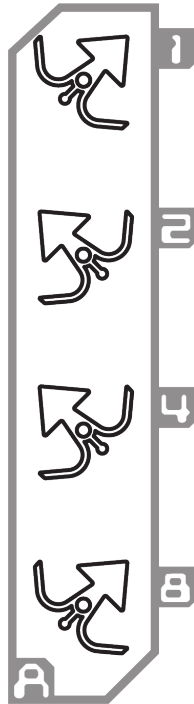
- Players learn that bits are labeled, and that those labels are important for understanding the directions.
- The conditional symbol “=>” is used to show examples of the challenge objective. “=>” means “IF whatever’s on the left, THEN whatever’s on the right.”

Possible hang-ups:

- If the top bit is pointed to the left, it must pass through the second bit so that it can test its direction.
- If the top bit is pointed to the right, it does not need to go through the second bit. Players can build a path directly to the interceptor.
- Once it triggers a red ball, that ball can go directly into the interceptor without passing through the bits.
- Players don’t have to use both crossovers to solve the challenge.

Computer Logic Lesson #9: Registers

A register is a group of bits that, together, make a number. You can read the number by adding together the numbers next to each right-pointing bit (more on that below).



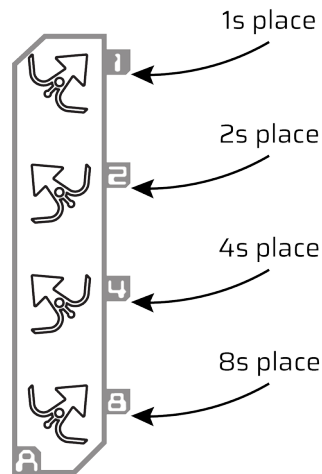
Computers use registers to store limited bits of information that they are working on at a time. They're sort of like the processor's workspace. You can think about it this way: You might have a shed in the backyard with lots of building supplies, but you only work with a few materials at a time in your workspace. A simple computer processor might have eight registers. A more powerful computer processor that runs a laptop might have 32, 64, or even more registers.

Registers can hold an instruction or some data. They can be used separately or together. For example, to add two numbers, a processor would first grab two numbers out of memory and load them each into registers. Then it would add the two registers together and store the sum in a third register.

In Turing Tumble, players can make registers and then use those registers to do any operation: add, subtract, multiply, divide, compare two numbers, and more. But to start out, players use the registers to count. For those who are new to coding and computer logic, using registers feels really abstract. But the more practice players get with the registers, it quickly

becomes second nature to read them.

How to read registers



Each bit in the register is a different place value.

- The top bit is the ones place. If it points to the right, you add 1. If it points left, you don't add it.
- The second bit is the twos place. If it points right, you add 2 to the total, otherwise you don't add it.
- The third bit is the fours place. If it points right, you add 4 to the total, otherwise you don't add it.
- The fourth bit is the eights place. If it points right, you add 8 to the total, otherwise you don't add it.

With that in mind, can you figure out the value shown in the picture above?

That's right, the ones place and the eights place are the only bits pointed right, so we add 1 and 8 together to get 9. The value of the register above is 9!

Way to go, not only do you understand how to read registers in Turing Tumble, you also just learned how **binary** works!

What is binary and why do computers use it?

Binary (also called "base 2") only uses two digits to represent numbers: 0 and 1. In fact, this is precisely why it is used in computers: A switch only has two positions: on or off, 0 or 1! All it takes is a series of switches to

represent a binary number.

When you see binary numbers written the conventional way, they aren't written top to bottom like the bits on a Turing Tumble board. They are written left to right, just like base 10 numbers, with the smallest place value on the right and the largest on the left. This table shows a comparison of how numbers look in decimal, binary, and on a Turing Tumble board:

Numbers 0 Through 15 Shown in Decimal, Binary, and in Turing Tumble

| Decimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---------------------------------|------|------|------|------|------|------|------|------|
| Binary | 0 | 1 | 10 | 11 | 100 | 101 | 110 | 111 |
| Binary (leading zeros shown) | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
| Turing Tumble | | | | | | | | |

- Continued -

| Decimal | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---------------------------------|------|------|------|------|------|------|------|------|
| Binary | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| Binary (leading zeros shown) | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| Turing Tumble | | | | | | | | |

Challenge 1:

Challenge 21 will have players watching the register count. Encourage your students to stop the computer at any time by holding up the left black lever so that no new balls are released. Have them compare the reading on the bits with the number of balls on the bottom.

Challenge 2:

The practice guide has some fun reproducible practice challenges and activities pages for reinforcing these concepts. Pages 77-82 in the practice guide can be found at www.turingtumble.com/edu.

Challenge #21: Quantum Number

(page 43 in puzzle book)



Challenge 21: Quantum Number

Objective: Use register A to count the number of blue balls. (Use 15 or fewer balls.)

Examples:

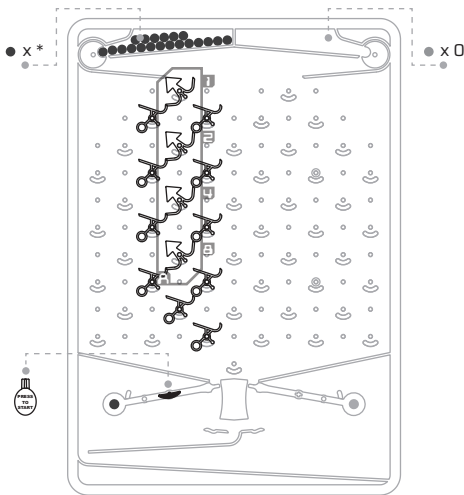
● x 5 ⇒ A = 5

● x 14 ⇒ A = 14

● x 7 ⇒ A = 7

● x 11 ⇒ A = 11

Starting setup



Available parts



x 5

Challenge 21 Solution

Explanation: This seems a lot more complicated than it really is!

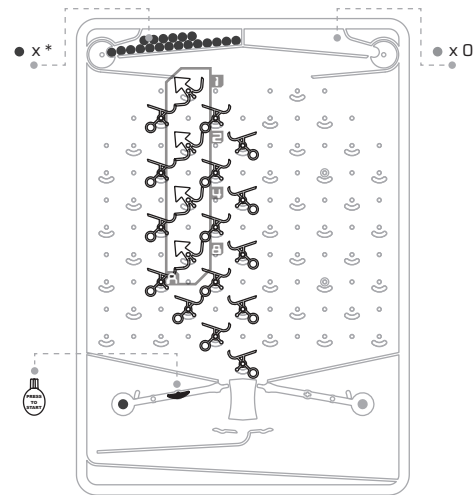
The top bit flips with every ball.

The 2nd bit flips every 2nd ball (whenever the top bit flips to the left).

The 3rd bit flips every 4th ball (whenever the 2nd bit flips to the left).

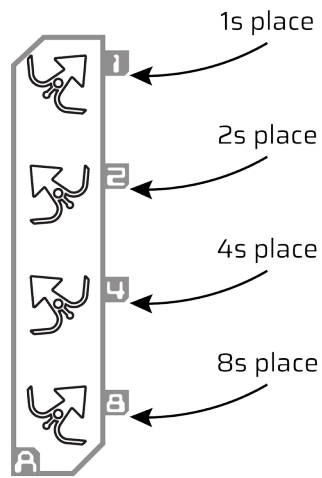
The 4th bit flips every 8th ball (whenever the 3rd bit flips to the left).

That's it! Remember how this works - it'll be in a lot of the puzzles to come.



What players learn about computer logic:

- In this challenge, players are introduced to registers.
- Registers are like a computer processor's workspace. The processor can load information into them from memory, perform operations on them (addition, subtraction, etc.), and then save the resulting information back into memory for later use.
- Numbers are stored on the registers by the direction each bit points. You can read the numeric value of a registers in the following way:



Each bit in the register is a different place value.

- The top bit is the ones place. If it points to the right, you add 1. If it points left, you don't add it.
- The second bit is the twos place. If it points right, you add 2 to the total, otherwise you don't add it.
- The third bit is the fours place. If it points right, you add 4 to the total, otherwise you don't add it.
- The fourth bit is the eights place. If it points right, you add 8 to the total, otherwise you don't add it.

Therefore, in the image above, the value shown is '9' ($1 + 8 = 9$).

What players learn about the game:

- Note the thought bubble, in the puzzle book, blue ball x * means the solution has to work for any number of balls.
- This challenge is not hard to build because it is focused on exploring and playing with the concept of registers. Players should take some extra time to experiment on their own with this challenge before moving onto the next one.

Possible hang-ups:

- When building the challenge, players will notice that the balls are fed through the blue bits on the left side, and are routed down to the left lever on the right side.
- Players might not understand that part of the challenge is watch the register count and learn how it works. Encourage them to stop the computer at any time by holding up the left black lever so that no new balls

are released. Have them compare the reading on the bits with the number of balls on the bottom.

- More practice may be needed before they are able to figure out challenge 22. There are more challenges and activity pages to reinforce the concepts in the practice guide found online at www.turingtumble.com/edu

Challenge #22: Depletion

(page 44 in puzzle book)



Challenge 22: Depletion

Objective: Register A starts at 15. Subtract the number of blue balls from the register. (Use 15 or fewer balls.)

Examples:

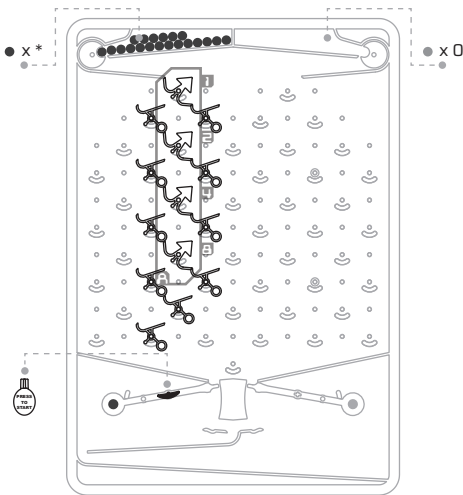
● x 1 ⇒ A = 14

● x 4 ⇒ A = 11

● x 10 ⇒ A = 5

● x 15 ⇒ A = 0

Starting setup

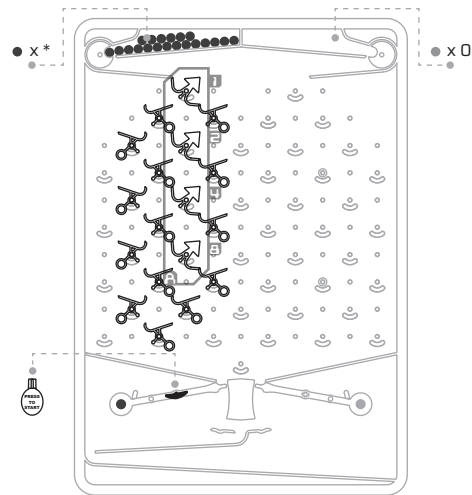


Available parts



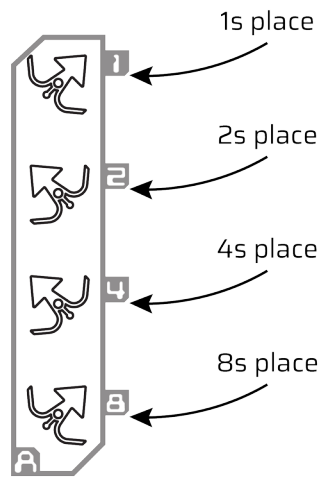
Challenge 22 Solution

Explanation: You can make a register count down instead of up by flipping the ramps around to the other side of the register!



What players learn about computer logic:

- In this challenge, players are introduced to **subtraction** using a register. In the previous challenge, players were using a register to count *up*, and now they are creating one that can count *down*.
- Registers are like a computer processor's workspace. The processor can load information into them from memory, perform operations on them (addition, subtraction, etc.), and then save the resulting information back into memory for later use.
- Numbers are stored on the registers by the direction each bit points. You can read the numeric value of a registers in the following way:



Each bit in the register is a different place value.

- The top bit is the ones place. If it points to the right, you add 1. If it points left, you don't add it.
- The second bit is the twos place. If it points right, you add 2 to the total, otherwise you don't add it.
- The third bit is the fours place. If it points right, you add 4 to the total, otherwise you don't add it.
- The fourth bit is the eights place. If it points right, you add 8 to the total, otherwise you don't add it.

Therefore, in the image above, the value shown is '9' ($1 + 8 = 9$).

- In this challenge, the number of balls on the bottom is the number subtracted and the remaining total is shown on the register above.

What players learn about the game:

- This challenge is not hard to build because it is focused on exploring and playing with the concept of registers. Players should take some extra time to experiment on their own with this challenge before moving onto the next one.

Possible hang-ups:

- When building the challenge, players will notice that the balls are fed through the blue bits on the right side, and are routed down to the left lever on the left side. This challenge is built in the opposite direction of Challenge 21 because players are subtracting from the register.
- Players might not understand that part of the challenge is watch the register count down and learn how it works. Encourage them to stop the

computer at any time by holding up the left black lever so that no new balls are released. Every time they stop the computer, they should read the bits to check the answer.

- More practice may be needed before they are able to figure out challenge 22. There are more challenges and activity pages to reinforce the concepts in the practice guide found online at www.turingtumble.com/edu.



Challenge #23: Tetrad

(page 45 in puzzle book)



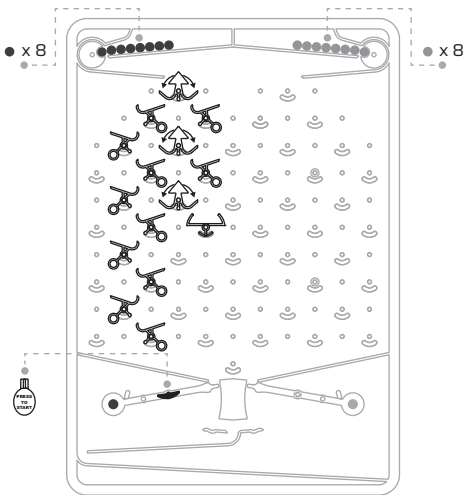
Challenge 23: Tetrad

Objective: Let exactly 4 blue balls reach the end. (Intercept the 5th.)

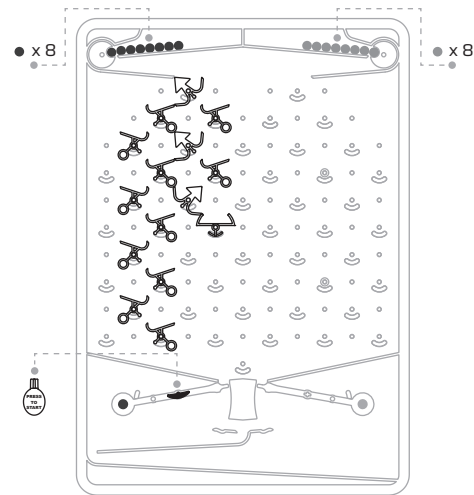
Required output:



Starting setup



Available parts



Challenge 23 Solution

Explanation: Again, think of the bits as a register. The register is set up to count down. Set the starting value of the register to 4. When you start the computer, it will allow 4 balls to pass through and capture the 5th ball in the interceptor.

What players learn about computer logic:

- In the previous puzzle, players learned how a register can be set up to count down, but why is that useful? This puzzle shows how it can be used for a simple purpose: to count out a certain number of balls and then stop itself. In a computer, counters are frequently used to loop through code a certain number of times before moving on to another part of the program.
- Registers are limited in how high (or low) they can count. The 3-bit register in this puzzle can only show numbers 0 through 7. So what happens when you count up past 7? Or, as this puzzle shows, what happens when you count down below 0? The answer is: it wraps around. If you count up one past 7, the register wraps around to 0. If you count down one below 0, it wraps around to 7.
- In computers, you must pay attention to the number of bits in your variables. Here's an example of a situation where it's especially important: A computer's timer. As soon as you turn a computer on, there is a variable

inside that starts at a value of '0' and increments once every thousandth of a second. This special variable is used to measure very accurate time intervals from within programs. The problem is, there are an awful lot of milliseconds in a day! You need a big variable to store that number. Unfortunately, even in modern computers, a 32-bit variable is often used to record it. The largest number that can be represented by a 32-bit variable is 4,294,967,296. If you do the math, you find that the variable wraps around to zero every 49 days! You can imagine the havoc it could cause in a program when it looks like time suddenly ran backwards 49 days. A keen programmer would take the size of the variable into account and create code to handle that rare occurrence.

- Players can see this happening when using a Turing Tumble register. In this puzzle, players are asked to let 4 blue balls reach the end and intercept the 5th. To do it, the register must be set to start at 4 (that is, only the bottom bit should point to the right). As the first 4 balls pass through, the register will count down to zero. The 5th ball then counts down one below zero, making it wrap around to '7', flipping all the bits to the right. When the last bit flips to the right, the ball drops into the interceptor.

What players learn about the game:

- This challenge integrates what players accomplished in challenge 16 (catching the 4th ball in the interceptor) with the knowledge gained from challenge 22 (counting registers down). Players learn how to make a machine count out a certain number of balls and stop itself.
- Players may also pick up on the fact that when a register counts down below 0, it wraps around to the highest number it can represent. In this case, with a 3-bit register, it wraps around to 7. The wraparound back to 7 is important because it is the only time when the bottom bit flips to the right, dropping a ball into the interceptor to stop the machine.

Possible hang-ups:

- It will be tricky to figure out what direction to set the bits. Players must set the register to the number 4 at the start. That way, 4 balls will reach the bottom and the 5th ball will fall into the interceptor.
- Players should use the counting down method they discovered in challenge 16 and 22.

Challenge #24: Ennead

(page 46 in puzzle book)



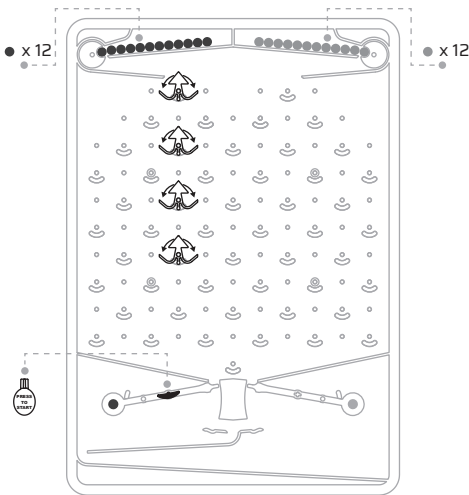
Challenge 24: Ennead

Objective: Let exactly 9 blue balls reach the end. (Intercept the 10th.)

Required output:



Starting setup



Available parts



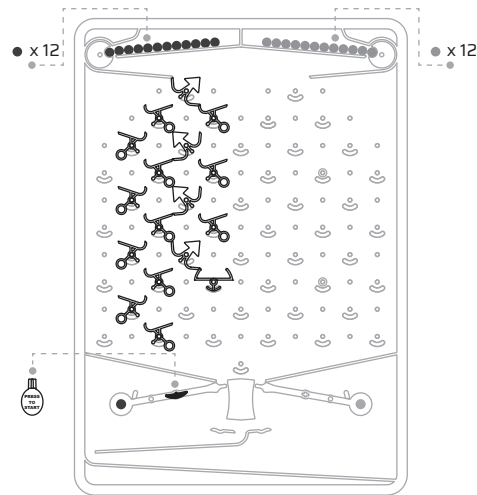
x 14



x 1

Challenge 24 Solution

Explanation: Think of the four bits as a register. The ramps are set up to make it count down. Set the starting value of the register to 9. When you start the computer, it will allow 9 balls to pass through and intercept the 10th.



What players learn about computer logic:

- See the previous challenge. The only difference here is that the countdown starts from 9 instead of 4.

What players learn about the game:

- This challenge builds on challenges 22 and 23. In challenge 22, players learned how to make a register count down. In challenge 23, players learned how to make the computer stop after the register counted down past zero. In this puzzle, their understanding of how to count down is reinforced as they create another countdown, but this time the countdown uses a bigger register (4 bits instead of 3 bits), and the register must be set to a value of '9' when players start the computer.

Possible hang-ups:

- The first step to solving this puzzle is to figure out where the ramps go. Players may not see the connection to the previous puzzle where they had to count out 4 balls, but the ramps need to be placed the same way as the previous solution
- Players might have figured out the direction to point the bits in the previous puzzle by trial and error. That is, they just pointed the bits in random directions until it eventually worked. In this puzzle, trial and error won't work as well. They'll need to understand that they are starting the register at a value of '9' so that it counts out 9 balls.

Challenge #25: Regular Expression

(page 47 in puzzle book)



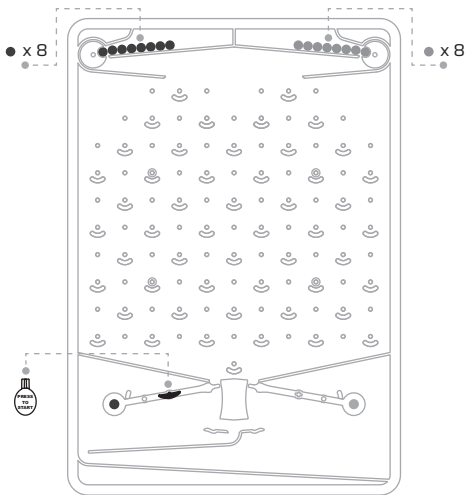
Challenge 25: Regular Expression

Objective: Generate the required pattern.

Required output:



Starting setup



Available parts

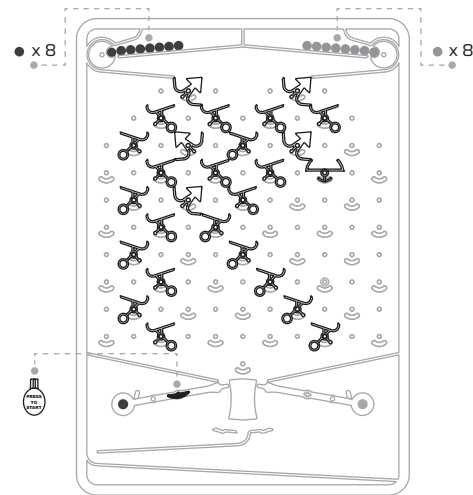


Challenge 25 Solution

Explanation: Make two countdown registers - one on the left side and one on the right.

To let out 6 blue balls, start the left register with a value of 5 (the 6th ball will be used to trigger the right lever).

To let out 3 red balls, start the right register with a value of 3. It will allow three red balls to reach the bottom and intercept the 4th ball.



What players learn about computer logic:

- Programs frequently use loops. Loops cause a section of code to run over and over again. To keep them from running forever, the program checks each time through the loop whether it should keep looping or not. Once it exits the loop, it moves on to run other code. In this challenge, we have two loops. When the first loop exits after releasing 6 balls, it begins running a second loop until it releases 4 red balls.

What players learn about the game:

- In challenges 23 and 24, players counted out a certain number of balls and used a final ball to stop the machine. In this puzzle, players learn that they can use that final ball to trigger other things. In this case, they will use the ball to start another countdown with red balls.

Possible hang-ups:

- Players should take this challenge in two parts: the first part is to create a countdown register that releases 6 blue balls. The second part is to create another countdown register that releases 4 red balls (the 4th red ball gets caught in an interceptor).
- This time, there are no parts shown in the starting setup. Players will have to use the knowledge gained from the previous challenges to build two registers that count down. Players probably won't have trouble building the register on the left since it's just like the previous two puzzles, but they might have trouble building the register on the right. Remember that it should be built the same way as the one on the left - not as its mirror image.

Challenge #26: Nucleus

(page 48 in puzzle book)



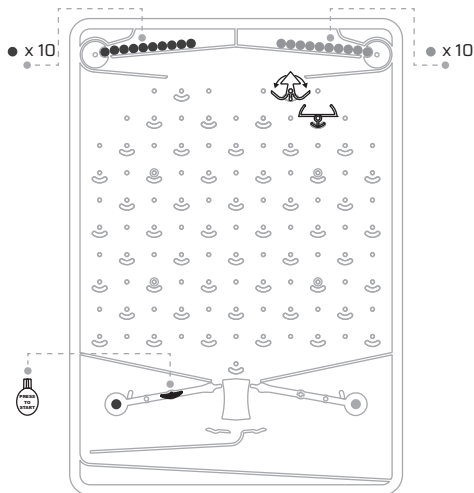
Challenge 26: Nucleus

Objective: Generate the required pattern.

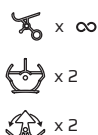
Required output:



Starting setup

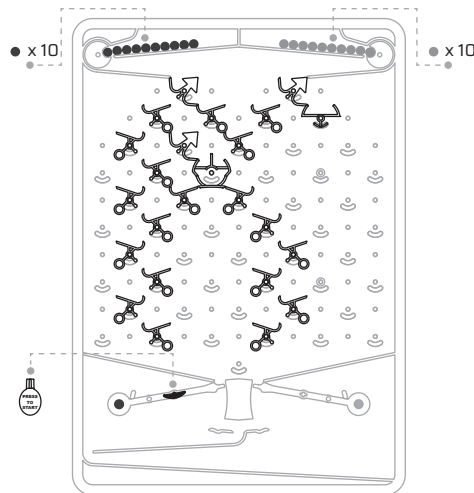


Available parts



Challenge 26 Solution

Explanation: The 2-bit countdown register on the left releases four blue balls, the last of which triggers the right lever, releasing a red ball. That red ball goes back over to the left side, triggering the release of four more blue balls, the last of which triggers the release of a second red ball. That second red ball is caught by the interceptor.



What players learn about computer logic:

- In computer programs, it's common to place loops inside of other loops. Loops placed inside of other loops are called “nested loops”. An inner loop could run through its code, say, 5 times. If the outer loop runs 3 times, the code in the inner loop will run 15 times.
- In this puzzle solution, you can think of the left countdown register as an inner loop. It releases a blue ball each time through. You can think of the right bit as the outer loop. The outer loop runs twice. Each time it runs, it runs through the inner loop and then it releases a red ball.

What players learn about the game:

- In this puzzle, players learn how to use a single register to count down more than once. Players build a countdown register on the left that counts down from 4, two times.
- Players get additional experience making a register count down below

zero, causing it to wrap around to its largest value. In this case, the register starts at a value of '3'. It counts down below zero and then wraps around to '3' again, making it ready to release another 4 blue balls.

Possible hang-ups:

- It might be tricky for players to understand what that bit on the right is for. It's there to direct the red balls. The first time a red ball goes over it, it directs the ball over to the blue side, causing the left register to begin counting down again. The second time a red ball goes over it, it directs the ball to the interceptor, stopping it.
- Students might not realize that the register they build on the left can be used to count down twice. If they build it correctly, it will reset itself to its starting value of '3' as soon as the 4th ball runs through it.

Challenge #27: Reflection

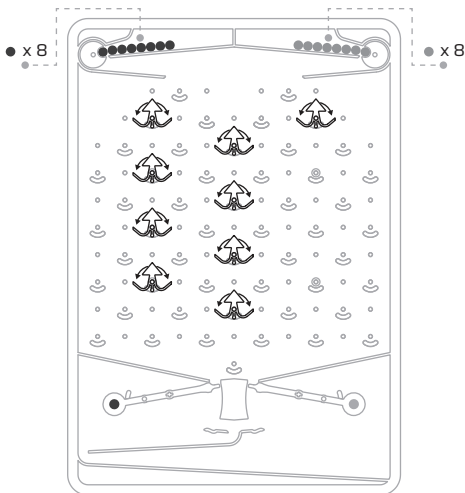
(page 49 in puzzle book)



Challenge 27: Reflection

Objective: Reverse the direction of each of the 9 starting bits, regardless of the direction they point to start.

Starting setup

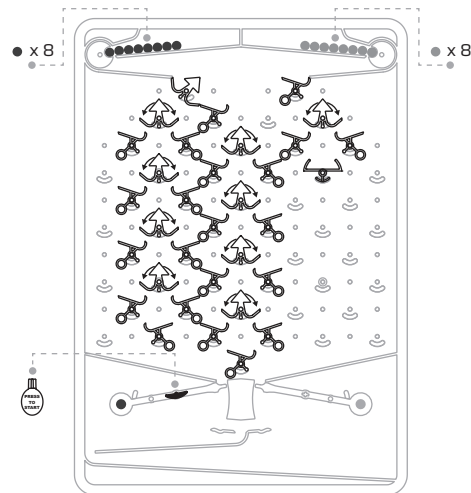


Available parts



Challenge 27 Solution

Explanation: Like the last puzzle, use one bit to split the path of the blue balls. That takes care of the first two columns of bits. The final bit on the right can then be flipped with a red ball, triggered by the second blue ball.



What players learn about the game:

- In this puzzle, there are three columns of bits that need to all be switched. The trick is to find a way to guide a ball through each column once.
- This puzzle helps players begin to think of bits as a way to distribute balls along different paths. In the solution, you can see that a bit is placed on top-left, under the blue ball releaser. That bit sends the first blue ball through the leftmost column of bits and it sends the second blue ball through the middle column of bits.

Possible hang-ups:

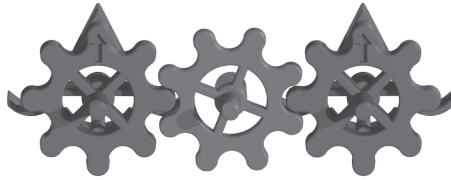
- To flip a bit in the opposite direction, a ball needs to go over it just once.
- Since bits can start pointed in either direction, the player must put ramps on either side of each bit to catch balls as they could drop off either way.
- The idea of using a bit to distribute balls along different paths might be a

tough one. It may be helpful to demonstrate how a bit can be used to split a single path into two paths.



Computer Logic Lesson #10: Gears and Gear Bits

In puzzle 28, the “gear” and the “gear bit” parts are introduced. They look like this:



Gears and gear bits are undoubtedly the coolest parts in Turing Tumble. They make it possible for one gear bit to turn other gear bits on the board. Chains of gears and gear bits can be built so that a gear bit can turn other gear bits that are below them, above them, or to their side.

They also make the game “Turing-complete”, which means that it has the ability to do anything a regular computer can do. We’ll get to that in a moment, but first, it’s important to understand how to use gears and gear bits. They can be a little confusing.

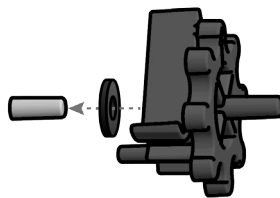
How to use gears and gear bits

Gears and gear bits are always used together. The purple gear bits fit on the regular pegs that have a ‘smile’ underneath them, while the red gears fit on the pegs in between that don’t have the smile.

Never use gear bits by themselves. They aren’t weighted correctly, so they bounce back the other direction after they’ve been flipped. If you want to use a gear bit by itself, use a regular bit instead.

High friction washers

Don’t forget about the high friction washers!



The high friction washers come in a little bag. It’s important to put them behind the gear bits (like in the picture below) *when only 2 gear bits are connected together*. Without them, the gear bits can bounce back in the

other direction after they've been flipped.

However, *when three or more gear bits are connected together, do not use high friction washers.* In that case, there is enough friction already.

Why do the gears and gear bits make it Turing-complete?

Without gears and gear bits, information can only be sent *down* the board, not up. In other words, a bit near the top of the board can affect whether bits below it get flipped, but a bit at the bottom of the board can't have any effect on bits above it.

Gear bits change that. They make it possible to send information both down *and* up the board. And that's important for basic computer functions like loops. For example, take a look at the following commands.

```
x = 0
While x < 5
  x = x + 1
Loop
End program
```

In this example, the variable *x* starts at 0. The 'while' statement checks the value of 'x' to see if it's less than 5. If it is, it continues to the next line, adding 1 to 'x' and then looping back to the 'while' statement again until *x* is equal to or greater than 5. Then it will skip the loop and the program will end.

Now imagine how you might implement this code on the Turing Tumble board. You'd have to have a register for the variable 'x'. And you'd probably have it set up so that every time a ball dropped, it incremented 'x' by 1. That's not too bad, you've already made things like that before.

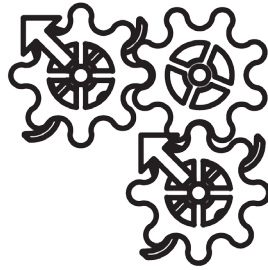
But here's where it gets tricky: Near the top of the board, you'd also need something to check the value of 'x' and make sure it's less than 5 before your ball goes on to increment it again. How can you check the value of bits that are lower down on the board? Or, put another way, how could the bits of register 'x' send information up the board to the part where it checks to see if 'x' is less than 5? You'd need a chain of gear bits.

Common combinations of gears and gear bits

In the coming puzzles, students will discover some useful combinations of gear bits that they'll continue to use in later puzzles. Below are three important ones:

Latch

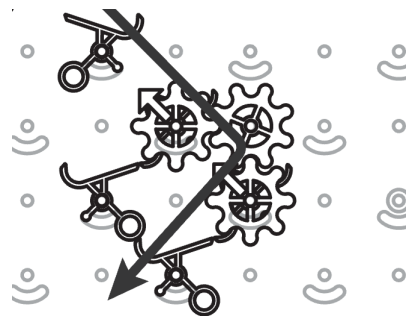
The latch is made up of two gear bits and a gear, arranged like this:



The latch is a permanent switch. In the pictures below, you can see how it works. The gear bits start pointing to the right. The first ball goes through and flips the gear bits to the left. From then on, they stay pointed to the left. The second ball, and every ball after that, flips them to the right and immediately back to the left again.



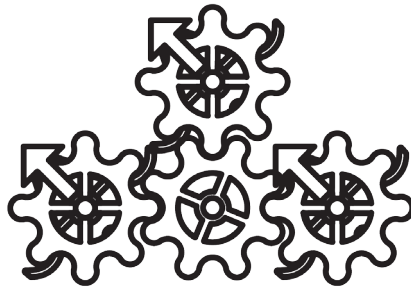
1st ball



2nd ball (and 3rd and 4th
and 5th...)

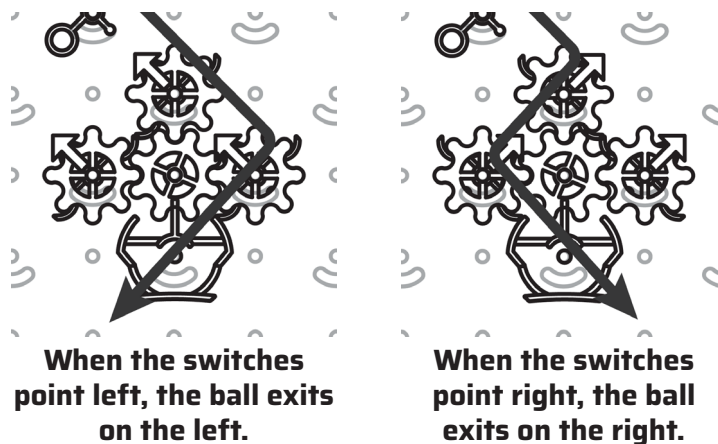
Fixed position switch

The fixed position switch is made up of three gear bits and a single gear, arranged like this:



Bits can be tricky to use in some situations because you can't test which way they point without changing the way they point. Any time you run a ball over a bit, it changes direction. However, this fixed position switch lets you test the direction the gear bits point without changing it.

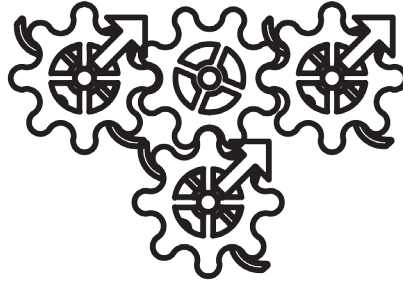
Here's how it works:



Any time a ball goes through the fixed position switch, it flips it twice, putting it back to its original position. By putting a crossover under the fixed position switch, you can detect which way the switch is pointing. If the ball exits from the left side of the crossover, then you know the switch is pointing left. If it exits from the right side of the crossover, then you know the switch is pointing right.

Set switch

The set switch is also made up of three gear bits and one gear:



The set switch gives you an easy way to make bits point a certain direction. If you drop a ball on the top-left gear bit, they will all point left, regardless of the direction they were pointing to start. If you drop a ball on the top-right gear bit, they will all point right, regardless of the direction they were pointing to start.

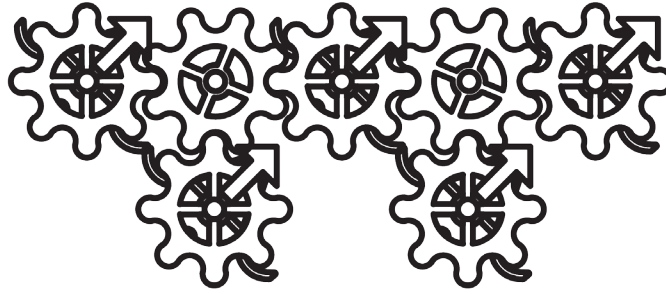
Flip-flop

Here's a challenge for your students. How can you make a special switch that combines the characteristics of the fixed position switch *and* the set switch? In other words, how can you build a switch that lets you test which way it points without changing the direction it points (like the fixed position switch), *and* you can also set it to point left or right regardless of the direction it points to start (like the set switch).

Here's a hint: It takes 5 gear bits and 2 gears.

(See the answer on the next page.)

Here's the solution:



This isn't actually used in any of the puzzles, but it would be a very useful switch if the board were larger. It works like this:

- If you drop a ball onto the top-middle gear, it behaves like the fixed position switch - it tests the direction it points without changing it.
- If you drop a ball onto the top-left gear, it makes the gear bits point left, regardless of the direction they pointed to start.
- If you drop a ball onto the top-right gear, it makes the gear bits point right, regardless of the direction they pointed to start.

This switch behaves just like a “flip-flop” in a computer. Flip-flops are tiny groups of switches that store a single bit of information. Computer memory is made of billions of flip-flops. The state of each flip-flop can be tested (without changing its state), or it can be intentionally set to a new state.

Challenge #28: Latch

(page 52 in puzzle book)



| Challenge 28: Latch | | Challenge 28 Solution |
|--|--|---|
| <p>Objective: Release only the blue balls.</p> <p>Required output:</p> | | <p>Explanation: Use a latch to send all balls to the ramps on the left side.</p> |
| <p>Starting setup</p> | <p>Available parts</p> <ul style="list-style-type: none"> x 1 x 1 x 1 | |

What players learn about the game:

- This is the first puzzle that uses gears and gear bits. Here, students learn a simple combination of gear bits: the latch. Once the latch flips to the left, it never flips back.

Possible hang-ups:

- If the gear bits don't work reliably, it's probably because there are no high friction washers behind them. Remember that whenever there are two gear bits connected together, the washers must be placed behind the gear bits to add a little extra friction. But whenever there are three or more gear bits connected together, you can leave them off.

Challenge #29: One-Shot Switch

(page 53 in puzzle book)



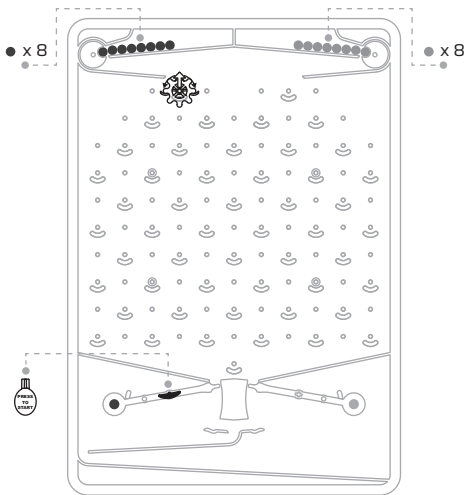
Challenge 29: One-Shot Switch

Objective: Release a blue ball, a red ball, and then the rest of the blue balls.

Required output:



Starting setup



Available parts



x ∞



x 1



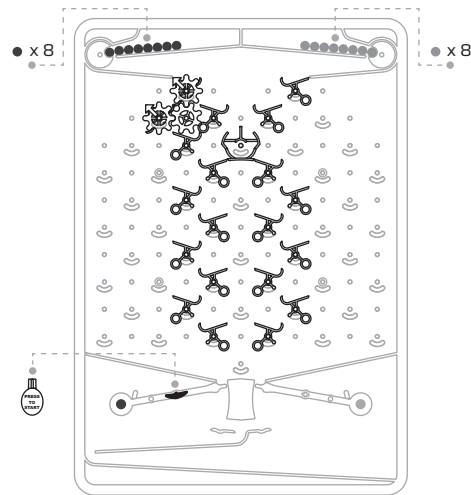
x 1



x 1

Challenge 29 Solution

Explanation: Use the first blue ball to flip the latch and trigger a red ball. The red ball triggers another blue ball that, from then on, is forced to trigger more blue balls by the (now flipped) latch.



What players learn about the game:

- Players discover a new way to use a latch - to cause the first ball through it to go one way and then all of the remaining balls to go a different way.

Possible hang-ups:

- Players might not realize that the latch makes the first ball go a different direction than all subsequent balls. Encourage them to trace their finger through the latch multiple times to see where the balls exit the latch.
- If the gear bits don't work reliably, it's probably because there are no high friction washers behind them. Remember that whenever there are two gear bits connected together, the washers must be placed behind the gear bits to add a little extra friction. But whenever there are three or more gear bits connected together, you can leave them off.

Challenge #30: Overflow

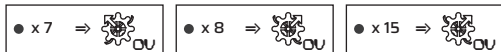
(page 54 in puzzle book)



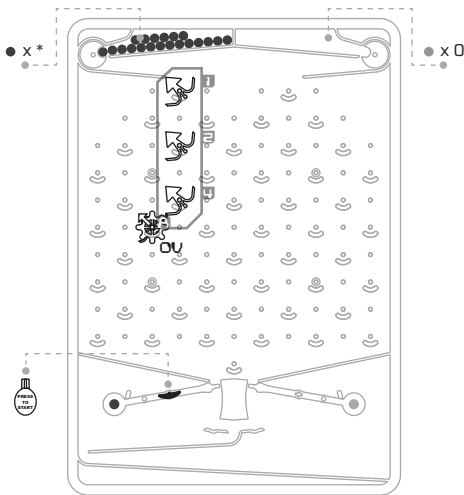
Challenge 30: Overflow

Objective: Count the blue balls in register A. If there are more than 7, gear bit OV must flip right (and stay right) to indicate the overflow.

Examples:



Starting setup



Available parts



x 14



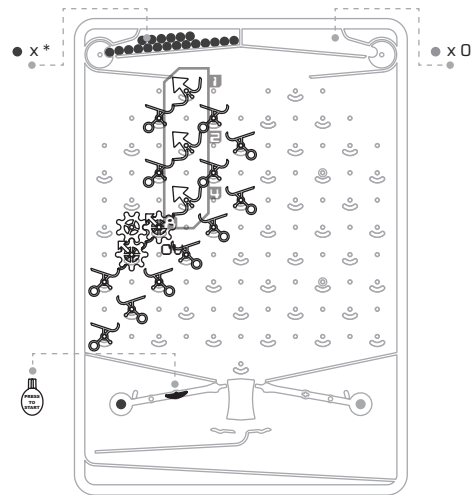
x 1



x 1

Challenge 30 Solution

Explanation: The latch flips when the count up register increments past 7. Once flipped, it remains flipped no matter how many balls run through the register.



What players learn about computer logic:

- By now we're pretty comfortable with registers. Registers contain bits that represent numbers. They hold numbers that are used by the processor for things like addition, subtraction, etc. But there is also something in a processor called a "status register". Each bit of the status register tells you something different. One of the bits in the status register is called the "overflow" bit. When a math operation results in a number that's too big for a register to hold, the overflow bit turns on. For example, if you tried to add $200 + 200$ and store the result in an 8-bit register, the overflow bit would turn on because an 8-bit register can only hold numbers up to 255. The result, 400, is too large.

What players learn about the game:

- In this puzzle, players learn a practical application for a latch: to indicate when a number has grown too big to fit in the register.

Possible hang-ups:

- Players may not understand the purpose of the gear bits in this puzzle. It may be helpful to explain what an “overflow” is. An overflow is when the number in a register gets too big to fit. In this challenge, we have a 3-bit register. It can only hold numbers between 0 and 7. But what if we try to count 9 balls with the register? On the 8th ball, the register would wrap around to ‘0’, and at the 9th ball, the register show a value of ‘1’. But that’s not correct! The overflow tells you, “Hey, don’t pay attention to the number in the register! It’s not correct. The real number it’s supposed to show is too big to fit in the register.”
- If the gear bits don’t work reliably, it’s probably because there are no high friction washers behind them. Remember that whenever there are two gear bits connected together, the washers must be placed behind the gear bits to add a little extra friction. But whenever there are three or more gear bits connected together, you can leave them off.