STEM Challenge

Construct a model that can go underwater

Suggested LEGO Spike build: Underwater Quest

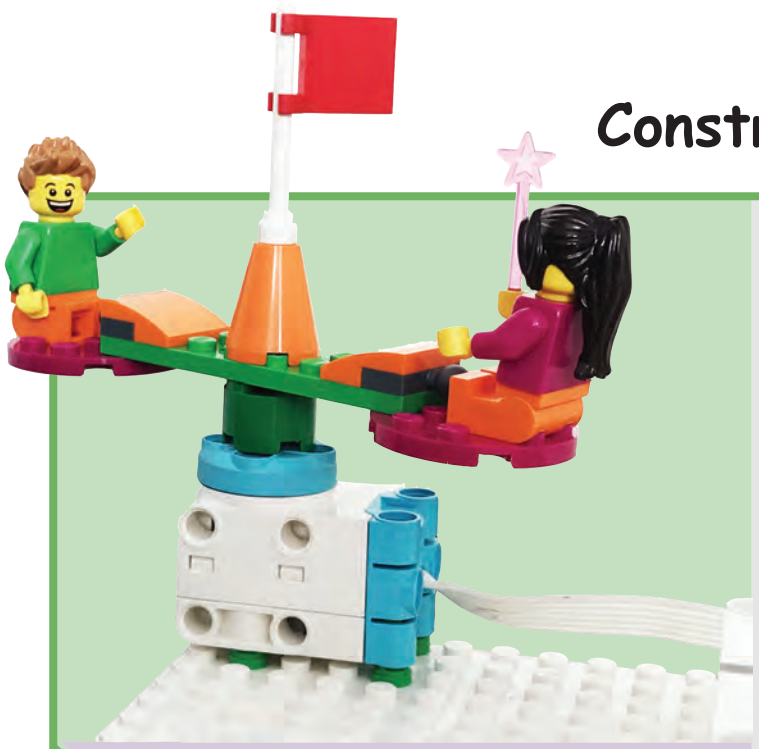
Remember to use the Design & Make Process!

Angles	How many degrees in a full rotation of the arm? What angle types and sizes can be seen in the model?
Measures	What is the area of the base of the model? How long can you make the arm that supports the submarine?
Problem Posing	Why does it need to go underwater? How would I make it suitable for two people? Could this be useful in my locality? Can you think of any other problems?
Engineering	How could I enclose the submarine? How do I keep it water tight?
Computational Thinking	What smaller tasks do we need to complete to build an underwater LEGO model? What are some common features that all underwater vehicles have? Can we identify these patterns and apply them to our LEGO model?

Thanks to Ashling C & Alan C, Askea BNS, Carlow for contributing this STEM Challenge idea

STEM Challenge

Construct an accessible carousel



Suggested LEGO Spike build: Classic Carousel

Remember to use the Design & Make Process!

Lines & Angles

- How many degrees will you turn in a quarter (1/4), half (1/2) and full rotation?

Measures

- What is the area of the base of the model?
- In a class of 30 students, how many turns can each person get in an hour?

Problem Posing

- Can you adapt your carousel so that anyone with a mobility aid could use it safely?
- Can you think of any other modifications that could be made? Why are they important to make?

Engineering

- Could you add a safety system using a sensor that would enable it to stop automatically when a student is close by?
- Could you add music that will play when the carousel is in motion?

Computational Thinking

- What lessons can you take away from this problem and apply to other problems?
- Does the result (of your LEGO build and/or code) match what you expected?

Thanks to Martin McE, Siobhán K, Colette C & Kate C from Scoil Mhuire gan Smál GNS, Carlow for contributing this STEM Challenge idea

STEM Challenge

Construct an accessible swing

Suggested LEGO Spike build: Underwater Quest

Remember to use the Design & Make Process!

Lines & Angles

- Draw a diagram of your modified swing. Make sure to measure it and add labels!

Visual Arts & SPHE

- Design a poster to advertise your inclusive swing.
- Devise a system so that everyone gets an equal amount of time on the swing

Problem Posing

- How could you modify the swing to make it inclusive for students with mobility aids?
- Could I add a second swing? Why/why not?
- Could this be useful in my locality?

Engineering

- How could you automatically lower the swing and return it to its original height?
- Will we need to modify the speed? Why?

Computational Thinking

- What are the different parts we need to construct a swing?
- Are there patterns or designs in real swings that we can identify and apply to our LEGO swing?
- Can we create an ordered set of actions that we can follow to ensure that our swing works correctly?

Thanks to Lorraine M, Mary K & Marian M, Askea GNS, Carlow for contributing this STEM Challenge idea

STEM Challenge

Construct a space car

Suggested LEGO Spike build: Cave Car

Remember to use the Design & Make Process!

Shapes & Space

- How many different 2D shapes you can you find on your car?
- How far can your car travel?

Measures

- How long is your car?
- How wide is your car?

Problem Posing

Imagine your space car needs to navigate through an alien landscape with unpredictable surfaces and gravity changes.

- How might you design it to handle these challenges?
- What unique obstacles it come across (steep slopes/low-gravity)?
- How could your space car adapt to these conditions and ensure successful exploration?

Engineering

- Where could you park your space car?
- Could you build something to keep your space car safe?

Computational Thinking

- What smaller tasks do we need to complete to build an underwater LEGO model?
- What are some common features that all underwater vehicles have? Can we identify these patterns and apply them to our LEGO model?

Thanks to Ashling C & Alan C, Askea BNS, Carlow for contributing this STEM Challenge idea

STEM Challenge

Invent a hovercraft

Suggested LEGO Spike build: Boat Trip

Remember to use the Design & Make Process!

Lines & Angles

- Draw a map with the route the hovercraft will take.
- Can you modify this to turn it into an obstacle course?

Measures

- Measure the distance travelled by the hovercraft.
- Measure the longest and the shortest route taken by the hovercraft to reach its destination.

Problem Posing

- How can you move between water/land without losing control?
- What happens if your hovercraft encounters an obstacle?
- What challenges did you face while building your hovercraft?
- How did you overcome these challenges?

Engineering

- How could I enclose the submarine?
- How do I keep it water tight?

Computational Thinking

- If your hovercraft encountered an unexpected obstacle, how could you program it to navigate around the obstacle?
- How could you modify it to carry a camera or sensor ?
- How could you modify your hovercraft's program to make it more efficient or faster at completing the obstacle course?
- What changes would you need to make to the code, and how would these changes affect the performance of the hovercraft?

Thanks to Sean K from Bishop Foley BNS, Carlow for contributing this STEM Challenge idea

STEM Challenge

Construct a submarine clock that stops every half hour

Suggested LEGO Spike build: Underwater Quest

Remember to use the Design & Make Process!

Angles

- How many degrees in a full rotation of the arm?
- What angle types and sizes can be seen in the model?

Measures

- Explore circle rotations, e.g. 180 degree turn of the spinning submarine.
- Design a clock big enough to fit in the background.

Problem Posing

- **Every half hour, passengers swap.**
- How can you get the submarine to stop every half
- How will you notify the passengers that it has stopped?

Engineering

- How could I enclose the submarine?
- How do I keep it water tight?

Computational Thinking

- Could you design a waiting platform for those waiting to get on?
- What code do you need to add sound to signal the half an hour is up?

Thanks to Ailbhe O'S, Megan C, Maria B & Claire S, St. Catherine's Senior GNS, Dublin for contributing this STEM Challenge idea

STEM Challenge

Construct a robot that can help with simple school tasks

Suggested LEGO Spike build: Little Big Helper

Remember to use the Design & Make Process!

Weight

- What is the maximum weight the robot can carry?
- What are some common objects that are heavy?
- What are some that are light?
- How does weight affect the movement of an object?
- Can heavy objects move as quickly as light object?

English & Geography

- Write a procedural text to tell your robot to do something.
- Draw a map for the Big Little Helper to get around the school.

Problem Posing

- How can we use our knowledge of weight to design and build a LEGO robot that can support a certain amount of weight?
- How can we design the robot to safely navigate the school's hallways and classrooms and avoid obstacles?

Engineering

- How can we create a robot is suitable to work both indoors and outdoors?

Computational Thinking

- What functions of the robot can you identify, for example; turning, stopping, moving forward, reversing etc.
- How will the robot identify items?
- If your robot is organising classroom materials, how can you decompose this task into actions like identifying items, picking them up, and delivering them to the correct location?

Thanks to Emma N & Fiona F, St. Finbarr's BNS, Dublin for contributing this STEM Challenge idea

STEM Challenge

Design and build a pirate ship

Suggested LEGO Spike build: Swamp Boat

Remember to use the Design & Make Process!

Coordinates

- Imagine your boat is a pirate ship and you are searching for buried treasure..
- Draw a map of the area that the boat will sail on.
- Include co-ordinates of the important locations on the map.
- Don't forget to include the location of the buried treasure!

PE: obstacle course & orienteering

- Divide students into "ship crews" and set up an obstacle course that mimics the challenges a pirate ship might face at sea.
- Activities could include: crawling under "rigging" (ropes), balancing on planks, and jumping over "waves" (hurdles).

Problem Posing

- Can you design a pulley system to load treasure onto the boat?
- Can you motorise the boat?
- Design a boat to travel on all terrains - land and water.

Engineering

- Add features to your boat to turn it into a pirate ship e.g. plank, mast, flag etc.

Computational Thinking

- Design a program that enables your robot to recognise and "hoist" different pirate flags.
- Design a program that instructs the robot to navigate to the correct spot on the map, pick up the treasure, and return to their ship.

Thanks to Sarah J, Sarah S, Nora A & Germaine F from St. Mary'Help of Christian's, Dublin for contributing this STEM Challenge idea

STEM Challenge

Design a modern day vehicle to send back to WWI

Suggested LEGO Spike build: Arctic Ride

Remember to use the Design & Make Process!

Geography	▸ Mapping - Define countries of Europe at the time ▸ Focus on the terrain; how it changed at different stages of the war etc.
History	▸ Research the vehicles present at the time. ▸ Identify functions of vehicles that might have been used.
English	▸ Novel Study: War Horse
Problem Posing	▸ Design a vehicle for a variety of purpose, for example: rescue, transport, fighting, medical aid
Engineering	▸ How could you create a vehicle suitable for the conditions presented? ▸ What does the vehicle need to provide safety for all its passengers? ▸ How is the vehicle going to be fuelled? ▸ What elements of the robot are going to provide back up if there was to be damage during war?
Computational Thinking	▸ Create a program that uses sensors to detect and relay information about "enemy troops" (colored objects) or "hazardous zones" (marked areas).

Thanks to Michelle B, Eoin F, Gary O'M & Aoife M from Our Lady of Victories BNS, Dublin for contributing this STEM Challenge idea

STEM Challenge

Construct an eco-friendly cabin

Suggested LEGO Spike build: Treehouse Camp

Remember to use the Design & Make Process!

Shape & Space	▸ What 2D or 3D shapes can you recognise in your cabin? ▸ Can you identify the number of vertices on your cabin?
Measures	▸ Using base plates, calculate the area and perimeter of the land your cabin is located on.
Problem Posing	▸ You'd like to invite large group of friends to see your new cabin. Will they fit? What will you show them? ▸ Could this be useful in my locality? ▸ Can you think of any other problems?
Engineering	▸ How could you create a ladder for the cabin that is safe and easy to climb? ▸ Could you create a pulley system for lifting and transporting supplies and materials up to the cabin?
Computational Thinking	▸ Program your robot to simulate a rainwater collection system. Your robot must open and close "valves" based on weather conditions and the cabin's water needs. ▸ Create a fan to keep your friends cool - can you adjust the speed and direction of the fan?

STEM Challenge

Invent a time-travelling machine

Suggested LEGO Spike build: Ferris Wheel

Remember to use the Design & Make Process!

Line & Angles	Draw a map to chart the journey of your time-travelling machine (must include right angles/perpendicular lines etc)
Measures	Measure the distance travelled by your machine on its adventures.
Problem Posing	What problems might you encounter on your adventures? How would you solve them? What might you discover if you were to travel back in time in your local town or village?
Engineering	How could you include sensors to collect specimens or to allow it to detect obstacles and avoid collisions? Could you add features to your time travelling machine such as headlights, tail-lights and indicators?
Computational Thinking	Can you write a program to control the indicators and signal left and right? Program your robot to interact with historical figures or events. Your program must allow your robot to engage in conversations or tasks relevant to the specific era it visits.

STEM Challenge

Create/Invent a _____

Suggested LEGO Spike build: _____

Remember to use the Design & Make Process!

_____

_____

_____

_____

Problem Posing

_____

_____

_____

_____

Engineering

_____

_____

_____

_____

Computational Thinking

_____

_____

_____

_____

Question Prompts

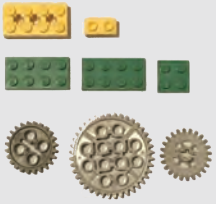
Making and understanding computational objects

Decomposition

1. What details can be noticed in this problem or object?
2. What parts are familiar/unfamiliar?
3. Can we break down the parts further into smaller parts?
4. How can we use the details to identify parts of this problem or object?
5. What are the different ways we could break down this problem, code or object?
6. How might breaking down this problem be helpful for solving or understanding it?

Pattern Recognition

1. What similarities or patterns do we notice between the problems or objects?
   For example, how many objects/colours are there?
2. How can we use the details to identify parts of this problem, robot, or object?
3. How can we describe the patterns?
4. How could we use the pattern to make predictions or draw conclusions?

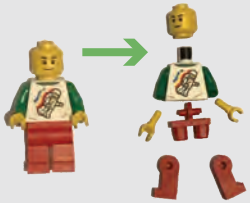


Abstraction

1. How can we simplify this problem/task?
2. What information is most important for solving this?
3. What information can we ignore in solving this?
4. How can we represent the important information?
5. What lessons can we take away from this problem and apply to other problems?

Debugging

1. Does the result match what we expected?
2. How can we tell whether or not our plan, model, or solution worked?
3. How can we modify our approach to address the problem?
4. How do we know that we have fixed the error?



Algorithmic Thinking

1. How can we create a clear, step-by-step set of instructions for our robot to follow?
2. Can we make our robot do the same task in fewer steps?
3. What if we want our robot to do different things based on certain conditions.
   For example, if it bumps into something, what happens then?
4. Can we make our robot repeat certain actions a specific number of times?
   For example, spinning in a circle or moving forward and backward?

Question Prompts

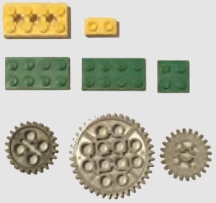
Making and understanding computational objects

Decomposition

1. What details can be noticed in this problem or object?
2. What parts are familiar/unfamiliar?
3. Can we break down the parts further into smaller parts?
4. How can we use the details to identify parts of this problem or object?
5. What are the different ways we could break down this problem, code or object?
6. How might breaking down this problem be helpful for solving or understanding it?

Pattern Recognition

1. What similarities or patterns do we notice between the problems or objects?
   For example, how many objects/colours are there?
2. How can we use the details to identify parts of this problem, robot, or object?
3. How can we describe the patterns?
4. How could we use the pattern to make predictions or draw conclusions?

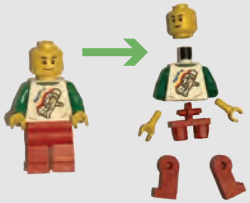


Abstraction

1. How can we simplify this problem/task?
2. What information is most important for solving this?
3. What information can we ignore in solving this?
4. How can we represent the important information?
5. What lessons can we take away from this problem and apply to other problems?

Debugging

1. Does the result match what we expected?
2. How can we tell whether or not our plan, model, or solution worked?
3. How can we modify our approach to address the problem?
4. How do we know that we have fixed the error?



Algorithmic Thinking

1. How can we create a clear, step-by-step set of instructions for our robot to follow?
2. Can we make our robot do the same task in fewer steps?
3. What if we want our robot to do different things based on certain conditions.
   For example, if it bumps into something, what happens then?
4. Can we make our robot repeat certain actions a specific number of times?
   For example, spinning in a circle or moving forward and backward?

Question Prompts

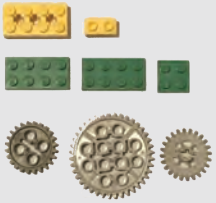
Making and understanding computational objects

Decomposition

1. What details can be noticed in this problem or object?
2. What parts are familiar/unfamiliar?
3. Can we break down the parts further into smaller parts?
4. How can we use the details to identify parts of this problem or object?
5. What are the different ways we could break down this problem, code or object?
6. How might breaking down this problem be helpful for solving or understanding it?

Pattern Recognition

1. What similarities or patterns do we notice between the problems or objects?
   For example, how many objects/colours are there?
2. How can we use the details to identify parts of this problem, robot, or object?
3. How can we describe the patterns?
4. How could we use the pattern to make predictions or draw conclusions?

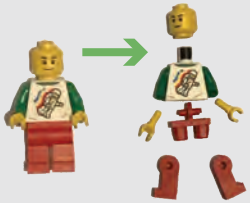


Abstraction

1. How can we simplify this problem/task?
2. What information is most important for solving this?
3. What information can we ignore in solving this?
4. How can we represent the important information?
5. What lessons can we take away from this problem and apply to other problems?

Debugging

1. Does the result match what we expected?
2. How can we tell whether or not our plan, model, or solution worked?
3. How can we modify our approach to address the problem?
4. How do we know that we have fixed the error?



Algorithmic Thinking

1. How can we create a clear, step-by-step set of instructions for our robot to follow?
2. Can we make our robot do the same task in fewer steps?
3. What if we want our robot to do different things based on certain conditions.
   For example, if it bumps into something, what happens then?
4. Can we make our robot repeat certain actions a specific number of times?
   For example, spinning in a circle or moving forward and backward?

Question Prompts

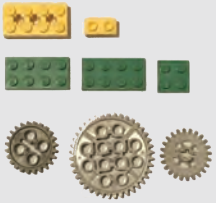
Making and understanding computational objects

Decomposition

1. What details can be noticed in this problem or object?
2. What parts are familiar/unfamiliar?
3. Can we break down the parts further into smaller parts?
4. How can we use the details to identify parts of this problem or object?
5. What are the different ways we could break down this problem, code or object?
6. How might breaking down this problem be helpful for solving or understanding it?

Pattern Recognition

1. What similarities or patterns do we notice between the problems or objects?
   For example, how many objects/colours are there?
2. How can we use the details to identify parts of this problem, robot, or object?
3. How can we describe the patterns?
4. How could we use the pattern to make predictions or draw conclusions?

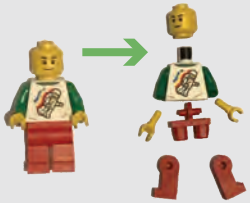


Abstraction

1. How can we simplify this problem/task?
2. What information is most important for solving this?
3. What information can we ignore in solving this?
4. How can we represent the important information?
5. What lessons can we take away from this problem and apply to other problems?

Debugging

1. Does the result match what we expected?
2. How can we tell whether or not our plan, model, or solution worked?
3. How can we modify our approach to address the problem?
4. How do we know that we have fixed the error?



Algorithmic Thinking

1. How can we create a clear, step-by-step set of instructions for our robot to follow?
2. Can we make our robot do the same task in fewer steps?
3. What if we want our robot to do different things based on certain conditions.
   For example, if it bumps into something, what happens then?
4. Can we make our robot repeat certain actions a specific number of times?
   For example, spinning in a circle or moving forward and backward?

Question Prompts

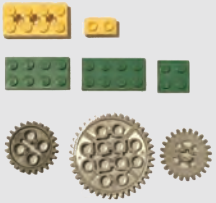
Making and understanding computational objects

Decomposition

1. What details can be noticed in this problem or object?
2. What parts are familiar/unfamiliar?
3. Can we break down the parts further into smaller parts?
4. How can we use the details to identify parts of this problem or object?
5. What are the different ways we could break down this problem, code or object?
6. How might breaking down this problem be helpful for solving or understanding it?

Pattern Recognition

1. What similarities or patterns do we notice between the problems or objects?
   For example, how many objects/colours are there?
2. How can we use the details to identify parts of this problem, robot, or object?
3. How can we describe the patterns?
4. How could we use the pattern to make predictions or draw conclusions?

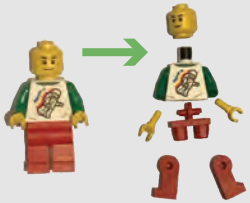


Abstraction

1. How can we simplify this problem/task?
2. What information is most important for solving this?
3. What information can we ignore in solving this?
4. How can we represent the important information?
5. What lessons can we take away from this problem and apply to other problems?

Debugging

1. Does the result match what we expected?
2. How can we tell whether or not our plan, model, or solution worked?
3. How can we modify our approach to address the problem?
4. How do we know that we have fixed the error?



Algorithmic Thinking

1. How can we create a clear, step-by-step set of instructions for our robot to follow?
2. Can we make our robot do the same task in fewer steps?
3. What if we want our robot to do different things based on certain conditions.
   For example, if it bumps into something, what happens then?
4. Can we make our robot repeat certain actions a specific number of times?
   For example, spinning in a circle or moving forward and backward?

Question Prompts

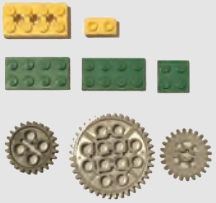
Making and understanding computational objects

Decomposition

1. What details can be noticed in this problem or object?
2. What parts are familiar/unfamiliar?
3. Can we break down the parts further into smaller parts?
4. How can we use the details to identify parts of this problem or object?
5. What are the different ways we could break down this problem, code or object?
6. How might breaking down this problem be helpful for solving or understanding it?

Pattern Recognition

1. What similarities or patterns do we notice between the problems or objects?
   For example, how many objects/colours are there?
2. How can we use the details to identify parts of this problem, robot, or object?
3. How can we describe the patterns?
4. How could we use the pattern to make predictions or draw conclusions?

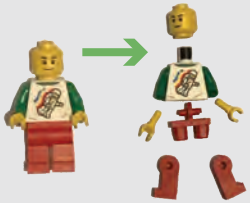


Abstraction

1. How can we simplify this problem/task?
2. What information is most important for solving this?
3. What information can we ignore in solving this?
4. How can we represent the important information?
5. What lessons can we take away from this problem and apply to other problems?

Debugging

1. Does the result match what we expected?
2. How can we tell whether or not our plan, model, or solution worked?
3. How can we modify our approach to address the problem?
4. How do we know that we have fixed the error?



Algorithmic Thinking

1. How can we create a clear, step-by-step set of instructions for our robot to follow?
2. Can we make our robot do the same task in fewer steps?
3. What if we want our robot to do different things based on certain conditions.
   For example, if it bumps into something, what happens then?
4. Can we make our robot repeat certain actions a specific number of times?
   For example, spinning in a circle or moving forward and backward?

Question Prompts

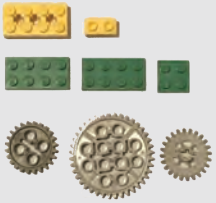
Making and understanding computational objects

Decomposition

1. What details can be noticed in this problem or object?
2. What parts are familiar/unfamiliar?
3. Can we break down the parts further into smaller parts?
4. How can we use the details to identify parts of this problem or object?
5. What are the different ways we could break down this problem, code or object?
6. How might breaking down this problem be helpful for solving or understanding it?

Pattern Recognition

1. What similarities or patterns do we notice between the problems or objects?
   For example, how many objects/colours are there?
2. How can we use the details to identify parts of this problem, robot, or object?
3. How can we describe the patterns?
4. How could we use the pattern to make predictions or draw conclusions?

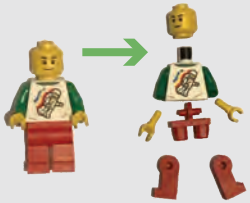


Abstraction

1. How can we simplify this problem/task?
2. What information is most important for solving this?
3. What information can we ignore in solving this?
4. How can we represent the important information?
5. What lessons can we take away from this problem and apply to other problems?

Debugging

1. Does the result match what we expected?
2. How can we tell whether or not our plan, model, or solution worked?
3. How can we modify our approach to address the problem?
4. How do we know that we have fixed the error?



Algorithmic Thinking

1. How can we create a clear, step-by-step set of instructions for our robot to follow?
2. Can we make our robot do the same task in fewer steps?
3. What if we want our robot to do different things based on certain conditions.
   For example, if it bumps into something, what happens then?
4. Can we make our robot repeat certain actions a specific number of times?
   For example, spinning in a circle or moving forward and backward?

Question Prompts

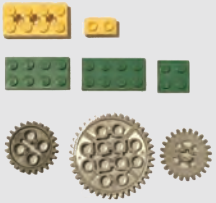
Making and understanding computational objects

Decomposition

1. What details can be noticed in this problem or object?
2. What parts are familiar/unfamiliar?
3. Can we break down the parts further into smaller parts?
4. How can we use the details to identify parts of this problem or object?
5. What are the different ways we could break down this problem, code or object?
6. How might breaking down this problem be helpful for solving or understanding it?

Pattern Recognition

1. What similarities or patterns do we notice between the problems or objects?
   For example, how many objects/colours are there?
2. How can we use the details to identify parts of this problem, robot, or object?
3. How can we describe the patterns?
4. How could we use the pattern to make predictions or draw conclusions?

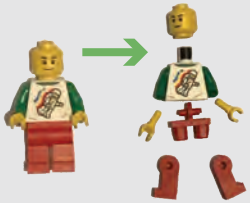


Abstraction

1. How can we simplify this problem/task?
2. What information is most important for solving this?
3. What information can we ignore in solving this?
4. How can we represent the important information?
5. What lessons can we take away from this problem and apply to other problems?

Debugging

1. Does the result match what we expected?
2. How can we tell whether or not our plan, model, or solution worked?
3. How can we modify our approach to address the problem?
4. How do we know that we have fixed the error?



Algorithmic Thinking

1. How can we create a clear, step-by-step set of instructions for our robot to follow?
2. Can we make our robot do the same task in fewer steps?
3. What if we want our robot to do different things based on certain conditions.
   For example, if it bumps into something, what happens then?
4. Can we make our robot repeat certain actions a specific number of times?
   For example, spinning in a circle or moving forward and backward?

Question Prompts

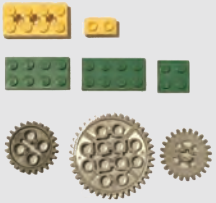
Making and understanding computational objects

Decomposition

1. What details can be noticed in this problem or object?
2. What parts are familiar/unfamiliar?
3. Can we break down the parts further into smaller parts?
4. How can we use the details to identify parts of this problem or object?
5. What are the different ways we could break down this problem, code or object?
6. How might breaking down this problem be helpful for solving or understanding it?

Pattern Recognition

1. What similarities or patterns do we notice between the problems or objects?
   For example, how many objects/colours are there?
2. How can we use the details to identify parts of this problem, robot, or object?
3. How can we describe the patterns?
4. How could we use the pattern to make predictions or draw conclusions?

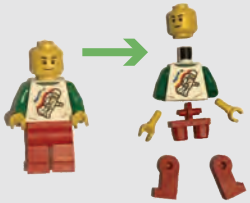


Abstraction

1. How can we simplify this problem/task?
2. What information is most important for solving this?
3. What information can we ignore in solving this?
4. How can we represent the important information?
5. What lessons can we take away from this problem and apply to other problems?

Debugging

1. Does the result match what we expected?
2. How can we tell whether or not our plan, model, or solution worked?
3. How can we modify our approach to address the problem?
4. How do we know that we have fixed the error?



Algorithmic Thinking

1. How can we create a clear, step-by-step set of instructions for our robot to follow?
2. Can we make our robot do the same task in fewer steps?
3. What if we want our robot to do different things based on certain conditions.
   For example, if it bumps into something, what happens then?
4. Can we make our robot repeat certain actions a specific number of times?
   For example, spinning in a circle or moving forward and backward?

Question Prompts

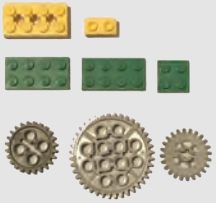
Making and understanding computational objects

Decomposition

1. What details can be noticed in this problem or object?
2. What parts are familiar/unfamiliar?
3. Can we break down the parts further into smaller parts?
4. How can we use the details to identify parts of this problem or object?
5. What are the different ways we could break down this problem, code or object?
6. How might breaking down this problem be helpful for solving or understanding it?

Pattern Recognition

1. What similarities or patterns do we notice between the problems or objects?
   For example, how many objects/colours are there?
2. How can we use the details to identify parts of this problem, robot, or object?
3. How can we describe the patterns?
4. How could we use the pattern to make predictions or draw conclusions?

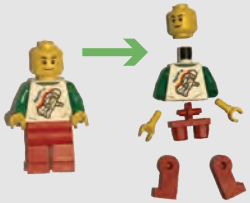


Abstraction

1. How can we simplify this problem/task?
2. What information is most important for solving this?
3. What information can we ignore in solving this?
4. How can we represent the important information?
5. What lessons can we take away from this problem and apply to other problems?

Debugging

1. Does the result match what we expected?
2. How can we tell whether or not our plan, model, or solution worked?
3. How can we modify our approach to address the problem?
4. How do we know that we have fixed the error?



Algorithmic Thinking

1. How can we create a clear, step-by-step set of instructions for our robot to follow?
2. Can we make our robot do the same task in fewer steps?
3. What if we want our robot to do different things based on certain conditions.
   For example, if it bumps into something, what happens then?
4. Can we make our robot repeat certain actions a specific number of times?
   For example, spinning in a circle or moving forward and backward?

Question Prompts

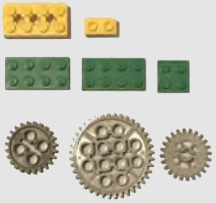
Making and understanding computational objects

Decomposition

1. What details can be noticed in this problem or object?
2. What parts are familiar/unfamiliar?
3. Can we break down the parts further into smaller parts?
4. How can we use the details to identify parts of this problem or object?
5. What are the different ways we could break down this problem, code or object?
6. How might breaking down this problem be helpful for solving or understanding it?

Pattern Recognition

1. What similarities or patterns do we notice between the problems or objects?
   For example, how many objects/colours are there?
2. How can we use the details to identify parts of this problem, robot, or object?
3. How can we describe the patterns?
4. How could we use the pattern to make predictions or draw conclusions?

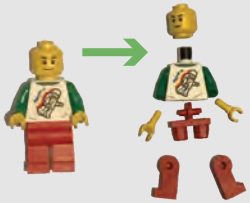


Abstraction

1. How can we simplify this problem/task?
2. What information is most important for solving this?
3. What information can we ignore in solving this?
4. How can we represent the important information?
5. What lessons can we take away from this problem and apply to other problems?

Debugging

1. Does the result match what we expected?
2. How can we tell whether or not our plan, model, or solution worked?
3. How can we modify our approach to address the problem?
4. How do we know that we have fixed the error?



Algorithmic Thinking

1. How can we create a clear, step-by-step set of instructions for our robot to follow?
2. Can we make our robot do the same task in fewer steps?
3. What if we want our robot to do different things based on certain conditions.
   For example, if it bumps into something, what happens then?
4. Can we make our robot repeat certain actions a specific number of times?
   For example, spinning in a circle or moving forward and backward?

Question Prompts

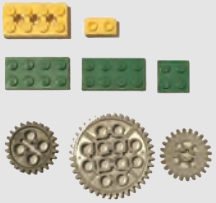
Making and understanding computational objects

Decomposition

1. What details can be noticed in this problem or object?
2. What parts are familiar/unfamiliar?
3. Can we break down the parts further into smaller parts?
4. How can we use the details to identify parts of this problem or object?
5. What are the different ways we could break down this problem, code or object?
6. How might breaking down this problem be helpful for solving or understanding it?

Pattern Recognition

1. What similarities or patterns do we notice between the problems or objects?
   For example, how many objects/colours are there?
2. How can we use the details to identify parts of this problem, robot, or object?
3. How can we describe the patterns?
4. How could we use the pattern to make predictions or draw conclusions?

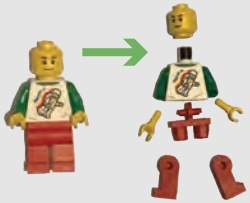


Abstraction

1. How can we simplify this problem/task?
2. What information is most important for solving this?
3. What information can we ignore in solving this?
4. How can we represent the important information?
5. What lessons can we take away from this problem and apply to other problems?

Debugging

1. Does the result match what we expected?
2. How can we tell whether or not our plan, model, or solution worked?
3. How can we modify our approach to address the problem?
4. How do we know that we have fixed the error?



Algorithmic Thinking

1. How can we create a clear, step-by-step set of instructions for our robot to follow?
2. Can we make our robot do the same task in fewer steps?
3. What if we want our robot to do different things based on certain conditions.
   For example, if it bumps into something, what happens then?
4. Can we make our robot repeat certain actions a specific number of times?
   For example, spinning in a circle or moving forward and backward?