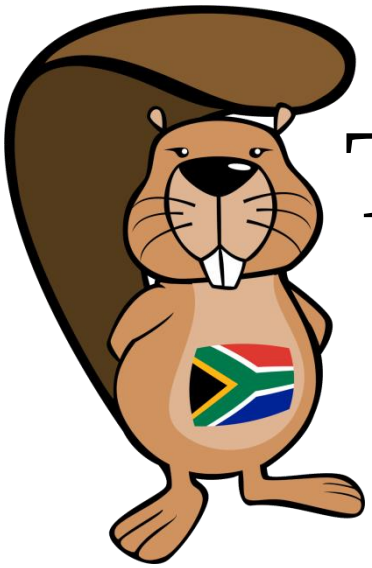
2022 Solutions

www.olympiad.org.za



Talent Search

Online

Introduction

The IITPSA Talent Search is available to all learners in South Africa from Grade 4 to Grade 12 and beyond. Students from other countries that do not have an organised Challenge identical to this in their own country are also welcome to join.

The Challenge aims to introduce learners and teachers to computational thinking and computer science. The Challenge aims at a large and broad audience with fun and engaging tasks. Talent Search allows a student to discover aptitude for computational thinking without requiring prior knowledge. It teaches participants that computational thinking is an integral part of solving problems in daily life.

The Talent Search philosophy: emphasize participation and celebrate achievement.

This Challenge was initiated in 2004 in Lithuania as the Bebras Challenge. “Bebras” is the Lithuanian word for “Beaver,” the hardworking star of many Bebras tasks. In the past years, over 2 million students from over 50 different countries have participated.

2011 was the first year in which the Bebras-type tasks were introduced to the larger community in South Africa. We were very excited about the positive feedback we have received from both learners and teachers.

The following pages show the tasks used in the Official Talent Search. The title of each page shows the name of the task, in which age group it was used and what the difficulty was (easy [A], medium [B] or difficult [C]).

For each question we not only show the answer, we also explain at least one way to reach the answer. On top of that there is a section that explains how the task is related to Computational Thinking.

We have also mapped the tasks to the Computational Thinking Concepts that feature in the Progression Pathways Assessment Framework created by Mark Dorling. This year we have also added a Computer Science domain to each task and some key word tags to help when searching for suitable tasks during the academic year. This is outlined below and is thanks to the work of Valentina Dagienė, Sue Sentance and Gabrielė Stupurienė:

We know many teachers have been waiting for this book. We hope the information in this book enhanced the usefulness of the challenge in your classroom!

The 2022 Talent Search was conducted in five age groups:

Group name	Recommended grade
Elementary	4 and 5
Junior	6 and 7
Intermediate	8 and 9
Senior	10 and 11
Elite	12 and above

Thank you

The challenge wouldn’t be the same without the help of these people:

Eljakim Schrijvers
Chris Roffey
Daphne Blokhuis
Dave Oostendorp
Arne Heijenga
Andrea Schrijvers
Kyra Willekes
Marissa Engels
and everybody that participated in the Bebras Workshop in 2021. Thank you!

Participating countries

Each problem in this booklet has a flag indicating the country of origin. However, many people were involved in the further editing, translating and providing additional material.

The US team is indebted to the generosity of spirit and community of computer scientists around the world!

Try the 2022 Talent Search Online

2022 Talent Search - Online Contests per division

Elementary
Grades 4 and 5

URL: http://challenge.beaver.org.za/index.php?action=anon_join&grp_id=329

Junior
Grades 6 and 7

URL: http://challenge.beaver.org.za/index.php?action=anon_join&grp_id=326

Intermediate
Grades 8 and 9

URL: http://challenge.beaver.org.za/index.php?action=anon_join&grp_id=327

Senior
Grades 10 and 11

URL: http://challenge.beaver.org.za/index.php?action=anon_join&grp_id=328

Elite
Grades 12 and above

URL: http://challenge.beaver.org.za/index.php?action=anon_join&grp_id=330

Overview of questions per division

Elementary
Grades 4 and 5

	Elementary
A1	Soccer Jerseys
A2	Moving the Ball
A3	Bridge Construction
A4	Go to the Market
B1	Stamping
B2	Preferences
B3	Farmer Beaver
B4	Turtle Path
C1	Necklaces
C2	Do they meet?
C3	Dentist
C4	Orange juice

Junior
Grades 6 and 7

	Junior
A1	Moving the Ball
A2	Go to the Market
A3	Bridge Construction
A4	Dancing Doll
A5	Farmer Beaver
B1	Coin Bag
B2	Turtle Path
B3	Necklaces
B4	Strawberry Thief
B5	Do they meet?
C1	Forest Observation
C2	Dentist
C3	Orange juice
C4	Pyramid of Coins
C5	Hercules and Hydra

Intermediate
Grades 8 and 9

	Intermediate
A1	Moving the Ball
A2	Dancing Doll
A3	Farmer Beaver
A4	Strawberry Thief
A5	Coin Bag
B1	Colourful Tube
B2	Turtle Path
B3	Do they meet?
B4	Forest Observation
B5	Hercules and Hydra
C1	A Desk Trouble
C2	Cuckoo Birds
C3	Hashing
C4	Meeting race
C5	Error Detection

Senior
Grades 10 and 11

	Senior
A1	Colourful Tubes
A2	Strawberry Thief
A3	Forest Observation
A4	Hercules and Hydra
A5	Truchet Tiles
B1	A Desk Trouble
B2	Jumping Jack
B3	Pyramid of Coins
B4	Fruit Stack
B5	Spider Quilts
C1	Hashing
C2	Cuckoo Birds
C3	Meeting race
C4	Playing with hats
C5	Error Detection

Elite
Grades 12 and above

	Elite
A1	Colourful Tubes
A2	Truchet Tiles
A3	A Desk Trouble
A4	Pyramid of Coins
A5	Jumping Jack
B1	Meeting race
B2	Spider Quilts
B3	Fruit Stack
B4	Hashing
B5	Playing with hats
C1	Unification
C2	Beaver Sort
C3	Team Building
C4	Error Detection
C5	Counting by Nodding

Contents

Dancing Doll

A dress-maker had to make dresses for four dolls.
Each dress must be made from four different materials.

Question:
Which doll is wearing a dress that is NOT made of four different materials?

Click on the correct answer.

Explanation

The correct option is:

The dress-maker has made dresses for the other dolls using all four materials, while with the doll above the rules are not met.

Background information

Data structures mean storing and manipulating data in the memory of a computer, in a way that the data can be used for constructing rules. This data can take the form of images, numbers or text.

In our example, the four materials that are required to be used are stored in the memory and each image is validated against the rules. The patterns of the images can be interpreted as colours as well. The patterns can be seen physically or calculated mathematically.

Forest Observation

Forest rangers need to observe the types of animals that wander onto paths in a forest.

They watch the paths from tall observation towers.

A ranger can only see the paths that connect directly to their tower.

Question

Click on the correct and least number of towers that need a forest ranger to observe all the paths.

Click on the observation towers that at least need to have a forest ranger to observe all the paths. Click on ‘Save” when you are done.

Explanation

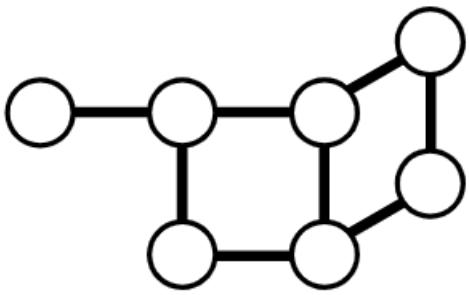
The correct answer is 3.

The three occupied observation towers and the respective overseen paths are shown in the image.

There are 8 paths. If only two observation towers were occupied, each one would have to observe at least 4 paths. This is not possible, because no observation tower is next to 4 paths.

Background information

In Informatics many things can be represented with graphs. Graphs consist of nodes (= circles) and edges (= lines), which connect the nodes. For our example a graph looks like this:



You can ask: “Which nodes (= observation towers) do you have to choose at least such that every edge (= forest path) is next to a chosen node (= observation tower)?” This question is also known as minimal vertex cover. It can for example be applied when putting up street lights, which should illuminate all streets. Another example is cameras, which should cover all hallways.

Football Jerseys

Áine is packing her bag for a football match. She needs to pack a shirt which has coloured sleeves, and a black collar. It must not have stripes.

Question:
Which shirt should Áine pack? Click on the correct answer.

Explanation

The correct answer is jersey B.

Jersey A is not correct because it has black sleeves.

Jersey B is correct because it has coloured sleeves, and a black collar, and does not have stripes.

Jersey C is not correct because it has stripes.

Jersey D is not correct because it has black sleeves and a white-collar.

Background information

In this task, two conditions have to be true (sleeve colour and collar colour) and one condition has to be false (stripes). Understanding conditions is very important in computer programming. All computer programming languages have conditions. Conditions can be used to tell which parts of a computer program should be run next (an "if" statement) and some conditions can be used to tell which objects should be included in, or left out of, lists of objects (a list comprehension).

This task can be used to introduce the logical Boolean operators AND and NOT. Because of the link between logic and set theory, this task can also be used to introduce membership of a set and intersections of sets.

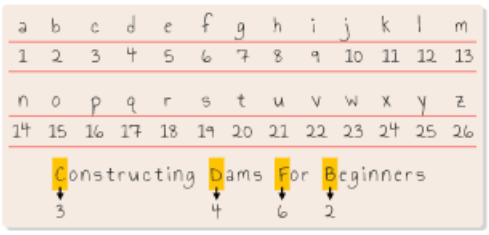
In the field of machine learning, classification is the concept of a computer program learning to group objects together based on their features. For example, the machine

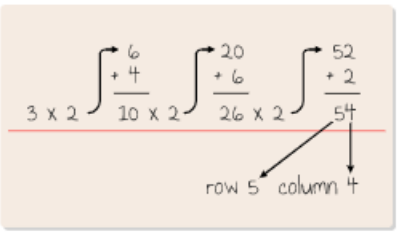
learning computer program might be given lots of example jerseys and it would figure out what combination of conditions (collar, sleeves, stripes, stars, colour, length, crest) are best to separate the two jersey types.

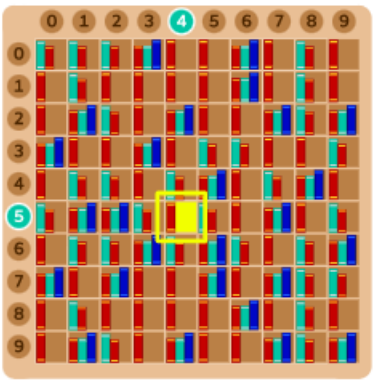
Hashing

Tim accompanied his older sister Sue to the Bebras Public Library. The library only has one huge bookcase. They wanted to borrow the book “Constructing Dams For Beginners”.

When they arrived, Sue went straight to the bookcase and pulled out the correct book. *“How did you know where the book was?”* Tim asked surprised. Sue smiled and showed him two pieces of paper:





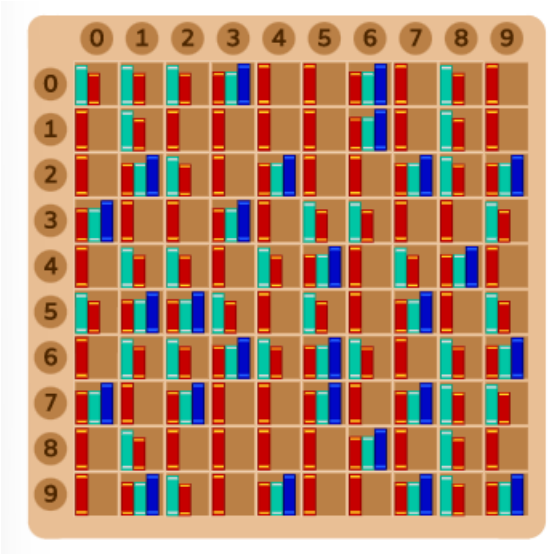


“I took the first letter of each word in the title and converted them into a number using the table. Then I multiplied the number of the first letter by 2 and added the number of the second letter. I then multiplied the result by 2 and added the number of the third letter. Finally, I multiplied that result by 2 once more and added the number of the last letter. I looked in the row of the second-to-last digit and the column of the last digit I obtained for the book. It is very easy to find that way, even if there were three books there,” explained Sue.

“But what about numbers greater than 99?” asked Tim.

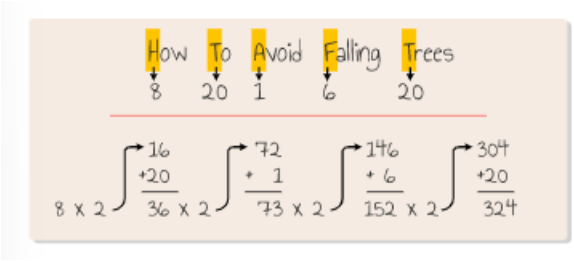
Sue replied: *“I just ignore all digits except for the last two.”*

Question:
In which bay can you find the book “How To Avoid Falling Trees”? Click on the correct numbers on the sides of the bookcase to select the corresponding bay. Click on ‘Save” when you are done.



Explanation

The correct answer is that the book is on the shelf in row 2 and column 4. The calculation is:



Background information

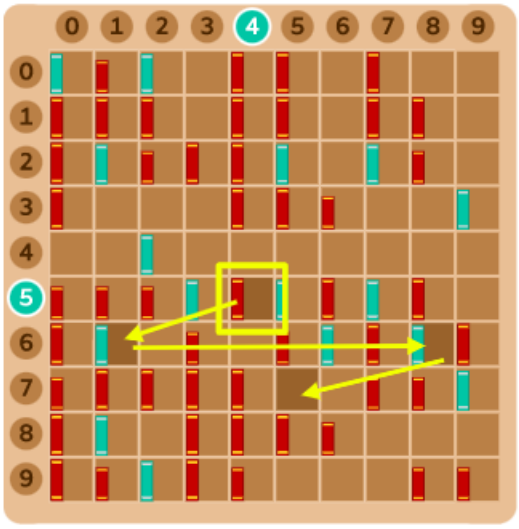
Behind the algorithm of the Bebras Public Library is a concept called “hashing”.

If no system is behind the library a (linear) search of every book would be necessary to find the book. On average you would have to check 50% of all books to find the desired book. Imagine searching the Library of Alexandria (~100’000 books), the Library of Congress (~38’000’000 books) or even just your local or your school’s library.

The problem not only exists for libraries. Big pharmacies also need to have a system to store and retrieve their medicine. In the last few years, more and more pharmacies are adopting automated storage systems. For them, a systematic order for instance by type of medicine doesn’t matter. Instead, they look for an even distribution of the shelf.

Here the concept of “hashing” comes into play. A hash value is a value calculated via a hash function from the properties of an item. In the case of this task, the title of a book is transformed into two digits that make up the row and the column of a place on the shelf. Of course, different books can end up with the same hash value like the books “Tree Bark Gourmet Guide” and Tasty Trees to Gnaw On”. There are different ways of dealing with such a conflict. One way is to simply put several items into the same place like in a place on the shelf. If that’s not possible the next empty place is chosen or an empty place *n* places away to have a more even distribution. When searching for said item simply the same number of places is checked also until an empty place is found. To

eventually fill up all places *n* is chosen to coprime to the number of places (like the number 7 in the example below).



Cuckoo Birds

Crested cuckoo birds don't build nests. Instead, they move into empty nests built by other birds.

When a crested cuckoo bird finds a tree with empty nests, it moves into a nest as follows:

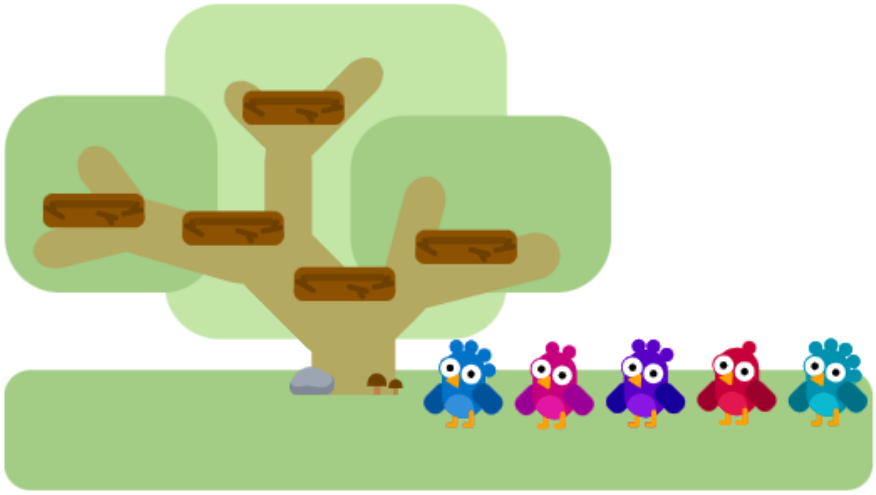
It starts at the bottom of the tree. It repeats the following steps until it finds an empty nest:

1. The bird goes up until it finds a nest.
2. If the nest is empty, it moves into the nest. Done.
3. If the nest is occupied, the bird looks at the other cuckoo bird already in the nest: if the other bird has more combs on its head, the bird continues searching to the left. If the other bird has the same number or fewer combs, the bird continues searching to the right.

There is a tree with five empty nests, and there are five cuckoo birds.
The birds move into the empty nests in order from left to right; the bird with four combs is the first.

Question

Drag each bird into the right nest following the method above. Click on ‘Save” when you are done.



Explanation

This way you get it right:

The first bird, with four combs, moves into the lowest nest.	
The second bird has two combs. The lowest nest is occupied by the first bird, with four combs. Since four is more than two, the second bird continues to the left and moves into the next empty nest.	
The third bird has three combs. Since four is more than three, at the lowest nest the bird continues to the left. The next nest is occupied by the second bird, with two combs. Since two is less than three, the bird continues to the right and moves into the next empty nest – which is the highest nest.	
The fourth bird has one comb. Like the birds before, at the lowest nest, the bird continues to the left. At the next nest, it must continue to the left, and move into the next empty nest, at the very left.	
The last bird has five combs. It has to go right at the lowest nest and find the next nest (at the very right) empty, where it moves in.	

Background information

Assigning birds to nests in this way has an interesting advantage. It makes finding particular birds quite efficient. If the bird you are looking for has fewer combs than the current bird being observed, look in the left portion of the tree. Otherwise, check the portion of the tree to the right. By repeatedly splitting the tree in half, the bird you are looking for can be quickly found.

This structured way of organizing data is called a binary search tree. It is often used in computer applications when it is necessary to be able to quickly retrieve data.

Truchet Tiles

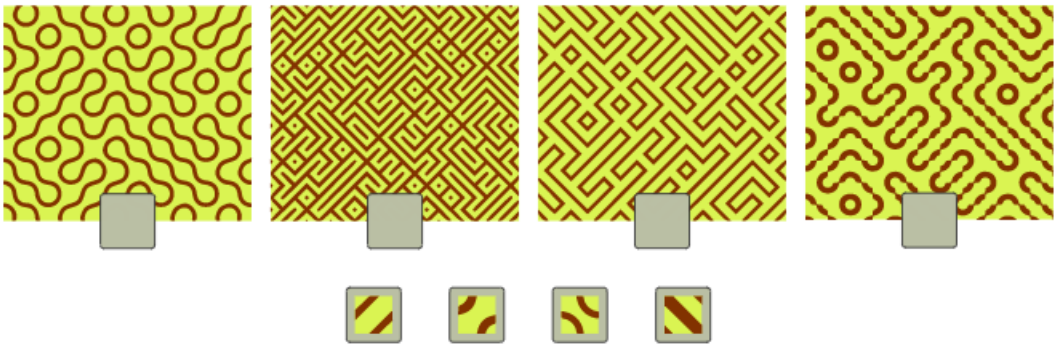
The following patterns were each created from a single tile.

Question

Match the tiles to their possible patterns.

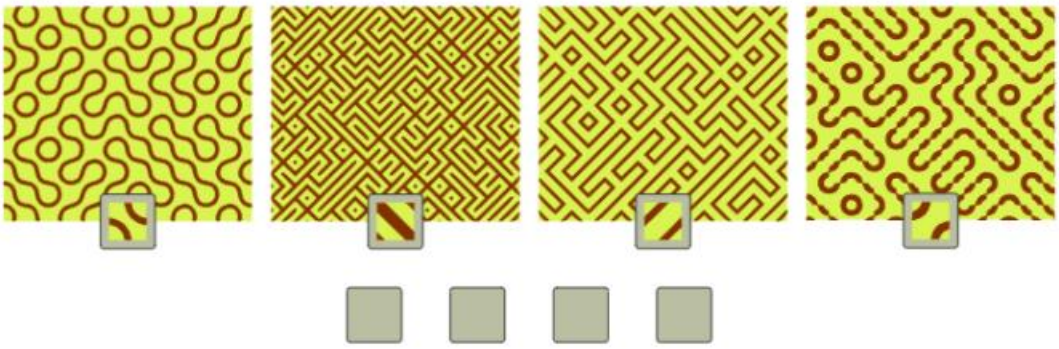
To do this, drag the single tiles into the grey fields under the patterns.

Click on ‘Save” when you are done.



Explanation

This is the correct answer.



The best way to find the correct answer is to look at where the lines on the tiles would meet.

In the pattern on the left, we see one continuous wavy line. So the line in the tiles that make this pattern would need to be curved and connect in the middle of the sides of the tile.

There is only one other pattern with curved lines so the other tile with a curve has to be used to create that one.

One of the remaining tiles has a dark colour right in the corner. If you look at one of the corners in the remaining two patterns you will notice only one of them has the same dark colour there.

Now only one tile and one pattern remain that have to go together.

Background information

These tiles are named after Sébastien Truchet who worked on variants of these tiles. Tiles with 4 same sides form a subset of Truchet tiles (Truchet tiles don't necessarily have to have 4 same sides like in the three patterns above).

The fact that complex patterns can be created with very simple building blocks has fascinated people for a long time.

Truchet tiles are studied in mathematics and Computer Science and they are used in computer games to generate mazes or decorations.

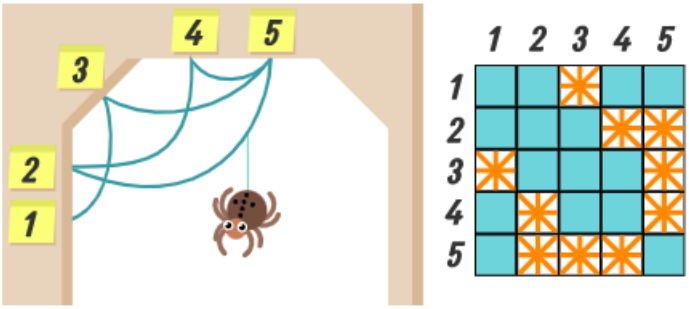
Spider Quilts

When Wanda sees an interesting web she uses it to design a new quilt.

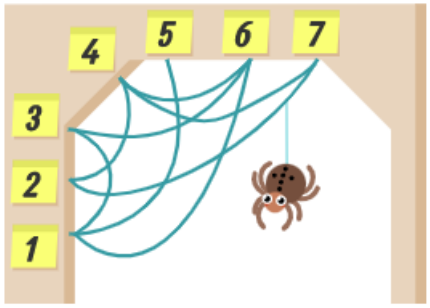
She numbers the web's anchor points from 1 to N and then arranges squares of fabric into an N-by-N grid as follows:

- For every piece of silk, if its anchors are numbered X and Y, she places two crossed fabric squares in her grid:
  - One crossed fabric square is placed where row X and column Y meet.
  - Another crossed fabric square is placed where row Y and column X meet.
- The rest of the grid is filled using solid fabric squares.

For example, the spider web on the left inspired the quilt on the right.

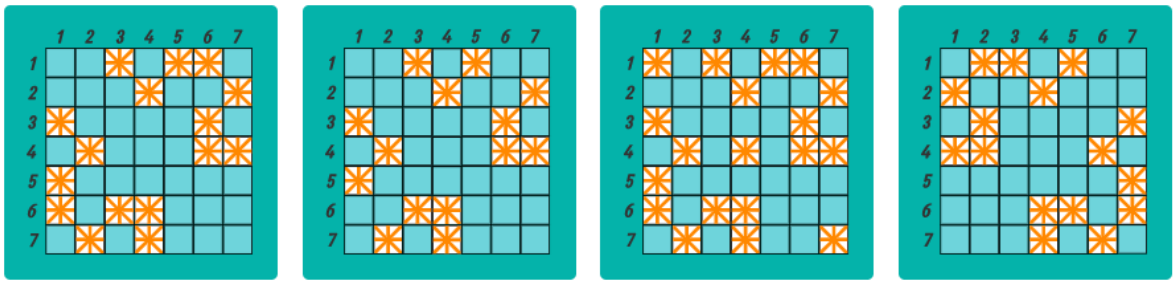


Wanda now sees the following web and wishes to design a new quilt:



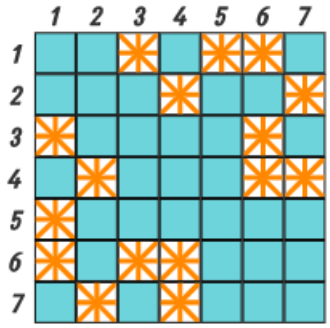
Question

What will her quilt look like? Click on the correct answer.



Explanation

The answer is:



The web has silk joining anchor 1 with anchors 3, 5, and 6. So the first row of the quilt will have crossed fabric in columns 3, 5, and 6.

The web has silk joining anchor 2 with anchors 4 and 7. So the second row of the quilt will have crossed fabric in columns 4 and 7.

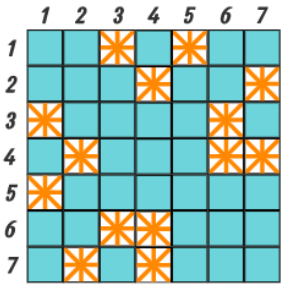
The web has silk joining anchor 3 with anchors 1 and 6. So the third row of the quilt will have crossed fabric in columns 1 and 6.

The web has silk joining anchor 4 with anchors 2, 6, and 7. So the fourth row of the quilt will have crossed fabric in columns 2, 6, and 7.

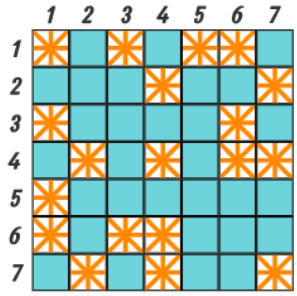
The web has silk joining anchor 5 with anchor 1. So the fifth row of the quilt will have crossed fabric in column 1.

The web has silk joining anchor 6 with anchors 1, 3, and 4. So the sixth row of the quilt will have crossed fabric in columns 1, 3, and 4.

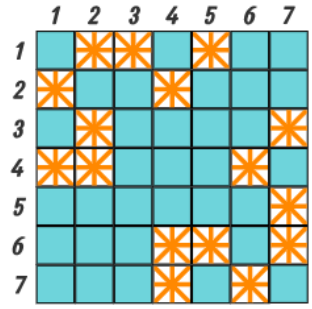
The web has silk joining anchor 7 with anchors 2 and 4. So the seventh row of the quilt will have crossed fabric in columns 2 and 4.



is incorrect because it is missing crossed fabric in row 1 column 6 and row 6 column 1.



is incorrect because it has crossed fabric incorrectly placed in row 1 column 1, row 4 column 4, and in row 7 column 7.



is incorrect because the entire quilt pattern is rotated 90 degrees.

Background information

The spider web can be considered a graph, a concept that is often used in Computer Science.
A graph is composed of vertices (the anchor points of the web) and edges (the pieces of silk between two anchor points). Graphs are used to represent objects and the relationships between objects. For example, a graph could show people who are friends on social media, or flights between countries.

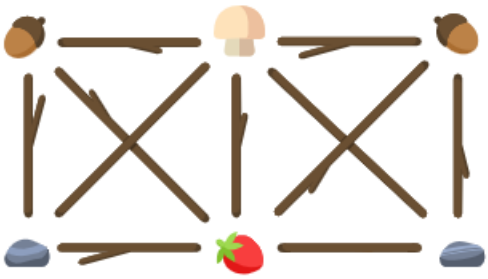
In this task, Wanda's quilt demonstrates an alternative way to represent a graph, known as an adjacency matrix. Adjacency matrices are useful representations since they provide an efficient way to answer questions about the structure of a graph. For example, does a particular edge exist? And how many edges connect to a given vertex?

Strawberry Thief

Anja is playing outdoors and makes a design on the ground using four types of objects: acorns, mushrooms, pebbles, and strawberries. She then adds sticks to her design according to her Very Important Rule:

A stick can go between two objects only if the two objects are different types.

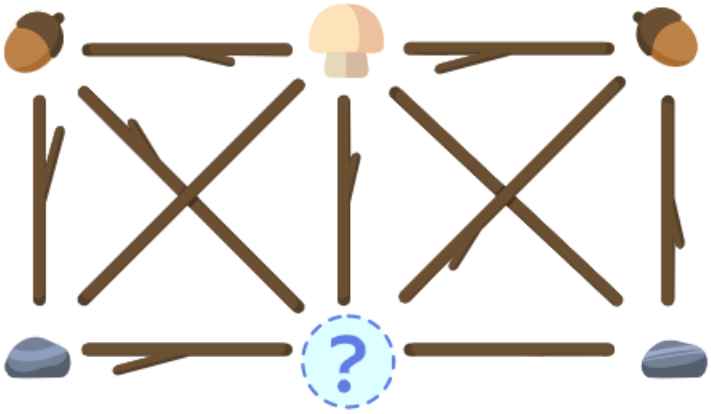
Here is Anja's completed design:



Anja's sister Zoë sees the design and eats the strawberry! To hide what she has done she replaces the strawberry with a different type of object.
She also removes exactly one stick so that the Very Important Rule will not be broken.

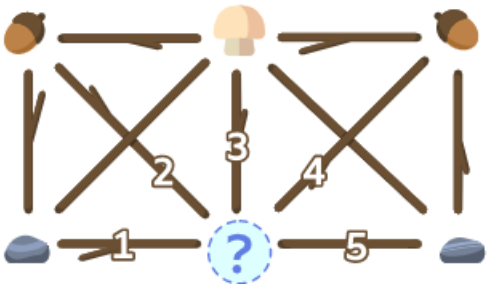
Question

Click on the stick Zoë removes and click on the question mark until you see the object with which she replaces the strawberry. Click on 'Save" when you are done.



Explanation

Zoë replaced the strawberry with a mushroom and removed the stick labelled 3 since it violates the Very Important Rule by connecting two objects of the same type.



All other strawberry replacements would require Zoë to remove more than one stick.
If Zoë replaced the strawberry with an acorn she would have had to remove sticks 2 and 4.
If Zoë replaced the strawberry with a stone she would have had to remove sticks 1 and 5.

Background information

Anja's design can be called a graph. The objects can be called nodes and the sticks can be called edges. In a graph, edges connect nodes. Two nodes that share an edge are called neighbours.

A subset of nodes where each node is a neighbour of every other node in the subset is called a clique. Anja's design contains two cliques of size four: the left half and the right half of the design.

Now suppose you wanted to assign the nodes of a graph a colour so that no edges connect two nodes of the same colour. It turns out that the number of colours needed to do this is at least the size of the largest clique.

Removing the strawberry is like trying to colour Anja's graph using at most three colours. This is not possible with cliques of size four, which is one reason why Zoë needed to remove a stick as well.

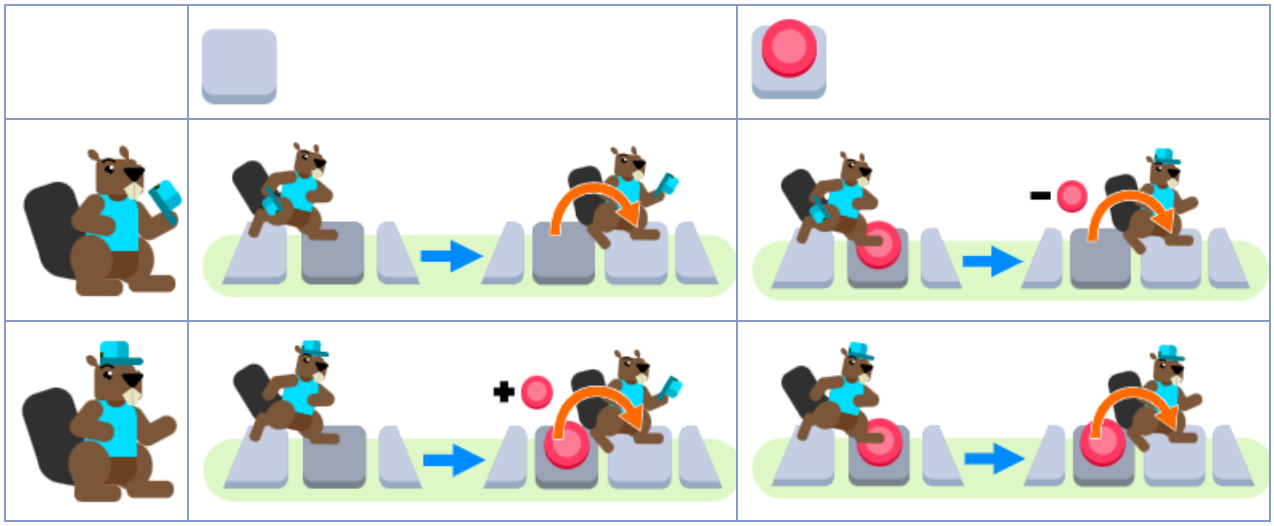
The problem of how to colour a graph using a minimal number of colours has many applications. Some examples include scheduling sports competitions, designing a seating plan, and even solving a Sudoku puzzle.

Playing with hats

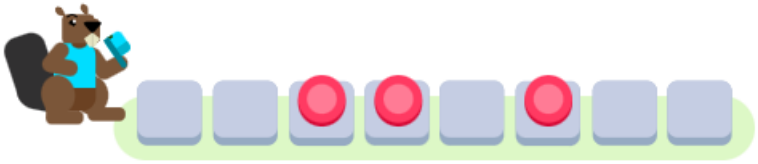
Beavers like to play a game by placing chips on squares.

The beaver moves from left to right from square to square .
The beaver has a hat and behaves differently depending on whether he has the hat on his head or in his hand on his hip. The pictures in the table below show the rules of the game. The changes for each situation are shown “before → after”.

For example: If the beaver has his hat on his head and steps on a square with no chip, he places a chip on the square and takes off his hat before moving to the next square.



In the beginning, the beaver has the hat in its hand and the chips are on the squares as in the picture below.



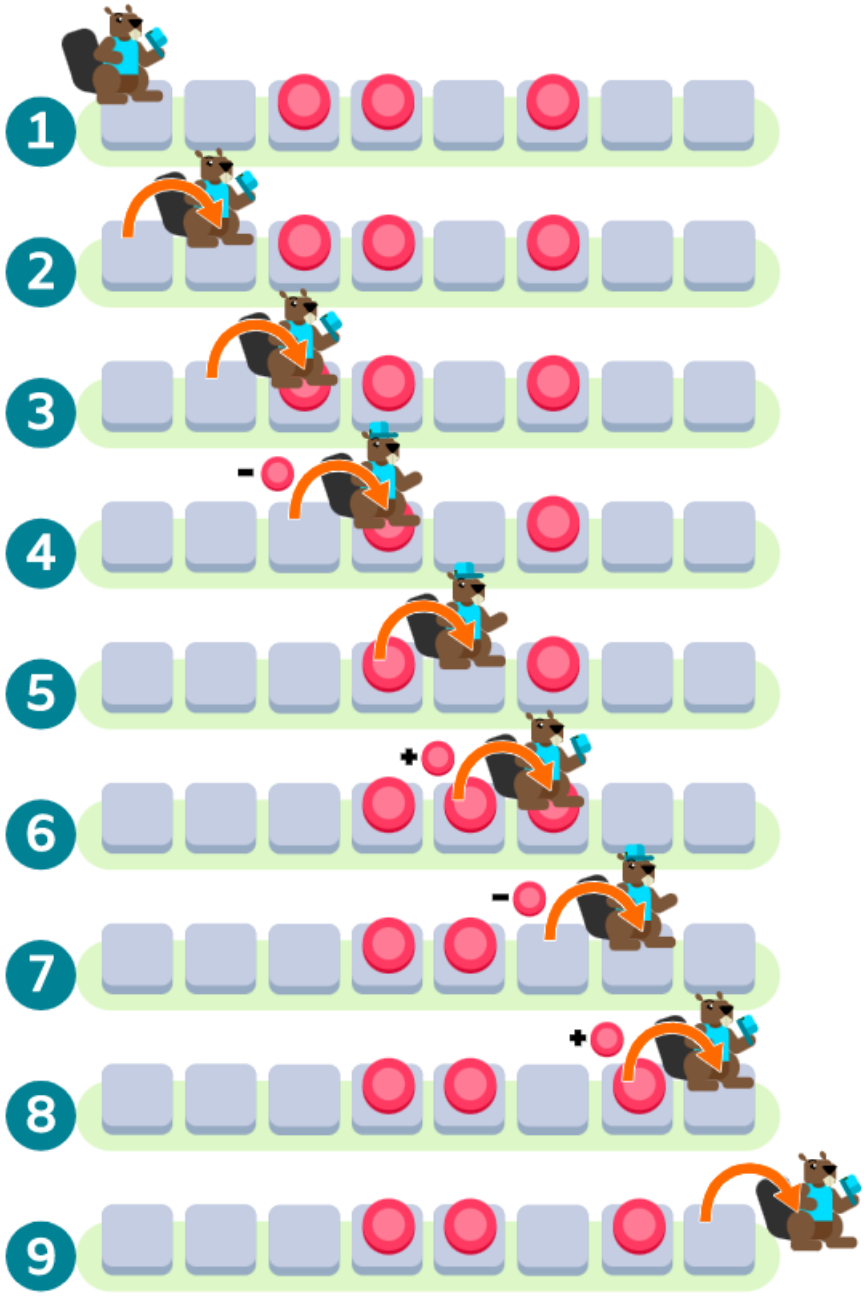
Question

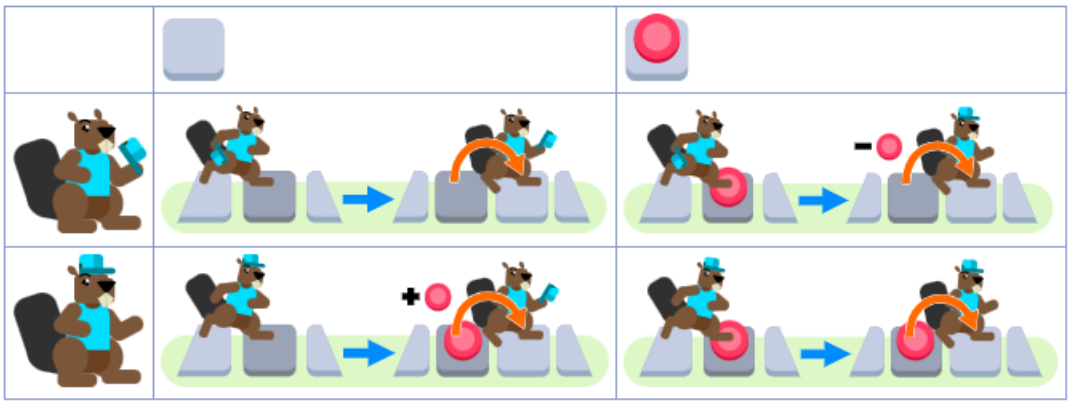
Which squares have chips after the beaver has moved from left to right, and he left the last square?

Click on the squares to place or remove a chip. Click on ‘Save” when you are done.

Explanation

The solution can be found by step-by-step analysis. We show this in a picture, using the rules below:





Background information

The beaver has two states: with the hat in the hand and with the hat on its head. Depending on its state, it behaves differently. The beaver with its rules and the fields behaves like a so-called Turing machine. A Turing machine is a useful model for computation in Computer Science. Although it is very simple, it is as powerful (and efficient!) as any programming language, meaning we can convert any software program into a Turing machine and, conversely, any Turing machine into a program. It was first described in 1936 by the English mathematician and computer scientist Alan Turing. Turing machines are one of the most important formal models in Computer Science.

A Turing machine has various necessary components:

A long tape is divided into squares. Normally it is said to be infinite.

- A finite alphabet of symbols, e.g., 0, 1. In our example, we used a chip and no chip.
- A read/write head: this would be able to look at a square and read its symbol. After reading and proceeding according to the rules the head would then move left or right one square at a time. In our case, the beaver represents the read/write head.
- A finite set of states: we used two states: hat in the hand and hat on the head.
- A set of rules (transition rules): to specify how the machine operates (see task description).

Fruit Stack

A family of father, mother, daughter, and son prepares breakfast for the next day.

They pile up four boxes, each filled with a different fruit:

Apple  , pear  , orange  , or strawberry  .

Being half-asleep in the morning, everyone just grabs the topmost box right after getting up. They do not know in which exact order they will get up, but the mother always does so before the daughter, and the father is always last.

Nobody likes all fruits; here is the list of preferences:

				
Father	✗	✗	✓	✗
Mother	✓	✗	✓	✓
Daughter	✓	✓	✓	✗
Son	✓	✓	✗	✓

Question

Drag the fruits into the boxes so that everyone is guaranteed to get a fruit they like.

When you have completed the task, click on Save.

Explanation

There is only one correct solution:

We can see this as follows: We first look at what the father wants. He only likes oranges and is getting up last. Therefore, we need to put oranges into the box at the bottom.

Because the mother has eaten when the daughter gets up, the mother is either the first or the second one to take a box.

For the same reason, the daughter is the second or third one to take a box. The son can be first, second, or third.

To summarize, the following three orders are possible:

1st	Mother	Mother	Son
2nd	Son	Daughter	Mother
3rd	Daughter	Son	Daughter
4th	Father	Father	Father

We see that the second one to get up can be either the son, daughter, or mother. This means that we need to put into the second box from top something they all like.

The only option is apple, as we can see from the table of preferences. (Second row in the table below.)

So we are left with two choices for the topmost box, pear and strawberry – The mother does not like pear, therefore we have to put strawberry into the first box, which the son also likes. (First row in the table below.)

We can now put pear into the third box, which son and daughter both like. (Third row in the table below.)

In summary, we had the following options for the order of the persons and the order of the fruits.

1st	Mother or son	
2nd	Daughter or son or mother	
3rd	Daughter or son	
4th	Father	

Background information

One of the first things computer scientists learn is how important it is to have everything correctly sequenced. They also need to understand the background information of the story. Without knowing exactly who will eat first, we need to organise the data to make the problem solvable. The actual order used in this task is stack order. In particular “Last in, First out” or LIFO. The pile of boxes in the fridge is what computer scientists would call a stack: A structure where only the item on top of the stack can be accessed. Only after removing the top item, does another one become available. Stacks are used very frequently in programming.

The task asks us to find a solution whereby the sorting of the fruits will work under multiple possible conditions. But there are some constraints, not all situations can occur. Solving such constraint problems can be very hard. Often, the best idea to do so is by writing and using a computer program to solve the problem for us.

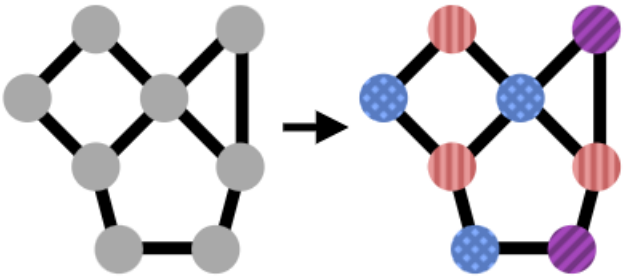
Logic is important in Computer Science and computer programming, which is why problems that help students understand logic lay a good foundation for when they start learning to create computer programs. Creating tables to display all possibilities (as shown in the explanation) is a good way to sort and sequence the given data. The use of Boolean logic may also be useful by using AND, OR and NOT to determine which data is useful in any given sequence. Students will start to understand conditionals in programming too, such as ELSEIF, by solving computational problems like the one shown in this question.

Once students have a good understanding of logic and how problems can be dealt with by following and sequencing commands logically, they will be better placed to write their own computer programs to solve problems with many variables. They will then be able to write programs to help deal with stacks.

Team Building

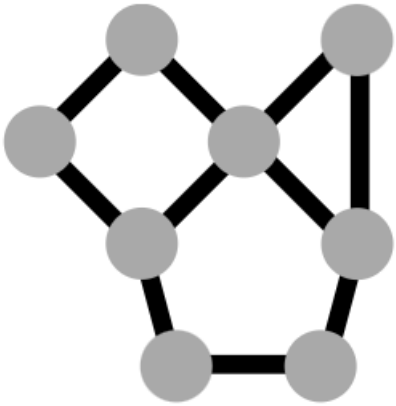
Eight people usually form three quiz teams.

In the graphs below, circles represent people. If there is a line between two people, this means they do NOT know each other. Knowing this, it is possible to assign teams by colouring in the circles, which can also help find the minimum number of teams possible.



Unfortunately, one table is broken tonight, so only two quiz teams can be formed. This becomes possible if two people are introduced to each other as this has the effect of removing one line from the graph. But, who to introduce?

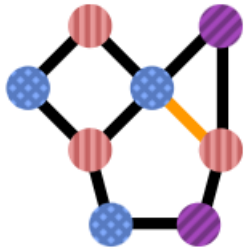
Task
Select one edge (line) from the graph that, when removed, allows two teams of four to be formed. Click on ‘Save” when you are done.



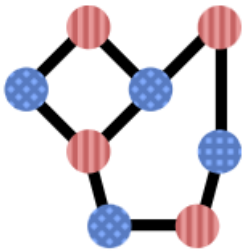
Explanation

Introducing two people means deleting an edge. We need to delete an edge so that two colours are enough to colour all the vertices (people) but no two vertices of the same colour are connected by an edge.

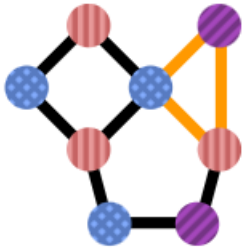
The only possible option is the edge marked in orange below.



After deleting this edge, we can colour the graph with two colours as shown below.

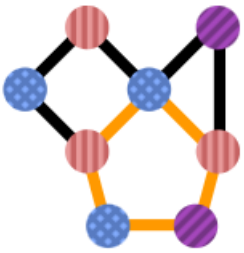


To test that deleting this edge is the only possible choice, we first consider the triangle in the upper right:



If any edge outside of this triangle is deleted, we still need three colours just for the three vertices of that triangle.

Now consider the cycle of five vertices at the bottom.



If any edge outside of this pentagon is deleted, then the cycle stays intact, and it is impossible to colour it with only two colours.

If we try it and go through the cycle clockwise, then we need to alternate between the two colours. But when we reach the last vertex, it will have the same colour as the first vertex because the number of vertices in the cycle is odd.

Therefore, we need to delete an edge, that destroys both the triangle and the cycle of five in the bottom at the same time, leading us to the only possible answer.

Background information

Many real-world problems can be formulated as colouring a graph. An example is a graph where the vertices are students and an edge between two students shows that they refuse to work with each other in a group. If we colour the vertices with k colours, this can be seen as assigning every student to one of k groups. Such a colouring is *proper* if any two vertices directly connected by an edge have different colours. Often, we just say *colouring* when we mean a *proper colouring*. An edge is sometimes called *critical* if deleting it makes a colouring with fewer colours than possible. In the example, this means that if the corresponding two students change their mind and agree to work together, then having fewer groups becomes possible.

Pyramid of Coins

Emil has 6 coins:

He laid them on top of each other on the table to make a shape like a pyramid.

Question
Which was the fourth coin Emil laid on the table?
Click on the coin and then click on ‘Save” when you are done.

Explanation

The correct answer is 

Each coin overlaps at least one other, so you can start to find the solution from the last top laid coin.

The last coin which is not overlapped by the others is  .

The coin not overlapped by any other than  is coin  .

And 4th laid coin must not be overlapped by any other than  and 

coins.

This coin is  .

Background information

The coins in the picture are laid in a *sequence*. You can see the same effect if you're drawing a picture on the computer: if you draw a circle, then two dots, and then a curved line, you get a smiley. If you had drawn the circle last, the two dots and curved line would have been hidden behind the circle.

Computers internally usually also work sequentially. Most computer programs are written so that first one action and then another action happens. So a computer program for drawing a smiley could look like this:

draw circle at (5,5) with radius 5

draw dot at (2,7)

draw dot at (7,7)

draw left curved line from (2,2) to (7,2)

Of course, sequences are not all computers can. To program more complex programs, you need to be able to make *decisions*, *repetitions*, and it helps to put frequently used program parts into separate so-called *sub-programs*.

Counting by Nodding

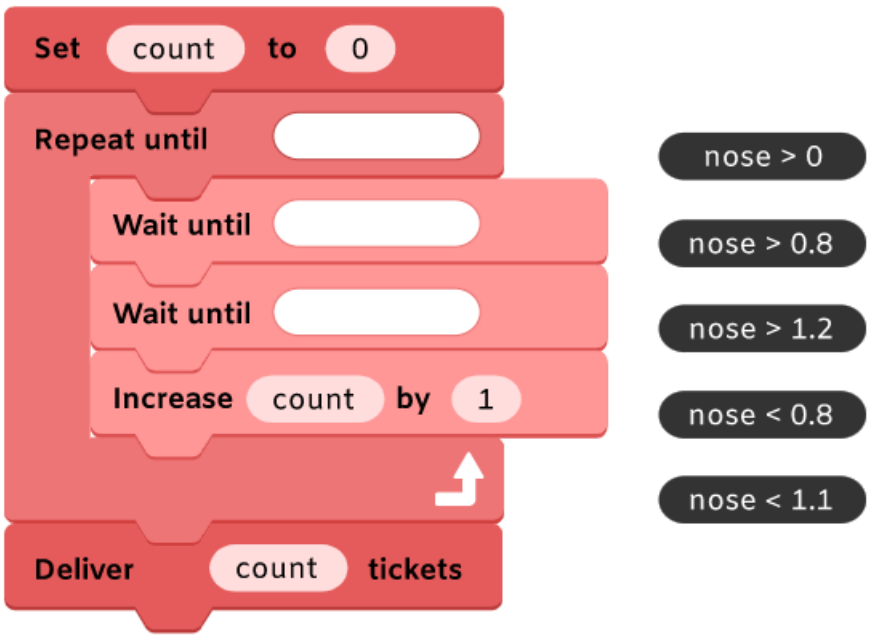
The ticket vending machine uses computer vision (CV) for communication. To purchase *n* tickets, the customer standing in front of the vending machine must nod *n* times and then raise the head once. The CV system constantly detects the vertical length of the bridge of the nose in the live camera image and assigns it to the variable nose.

If the value of nose is 1, the head is in its normal position.	
When the customer nods and the head goes down, the value of nose gets greater than 1, because the nose appears to be longer.	
When the head is raised, the value of nose gets lower than 1.	

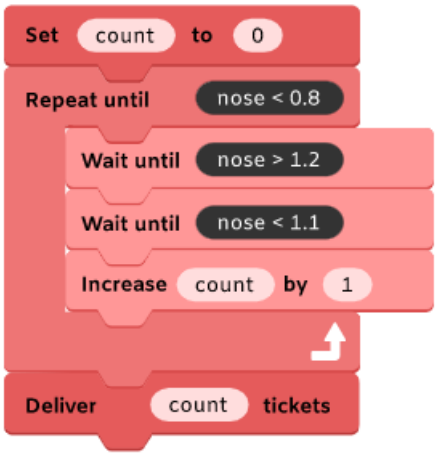
The control program is started when a customer stands in front of the vending machine and the head is in its normal position.

Question

Below you see a skeleton of the control program.
Complete the program by dragging appropriate condition blocks from the right to the gaps. Click on ‘Save” when you are done.



Explanation



The program uses two variables named count and nose. The variable count contains the number of nods, and nose represents the visible nose bridge (see task) and is automatically updated by the CV system.

A sequence of three commands is repeated in a loop until the head is raised and therefore nose gets a value smaller than 0.8. These repeated commands manage the counting: First, the system waits until the head goes down (nose > 1.2) and then waits until it goes up again (nose < 1.1). This is one complete nod. The value of the variable count is increased by 1.

When the loop is finished (because the person has raised the head and nose is smaller than 0.8), the variable count contains the number of nods and count tickets are delivered.

The program has to use inequalities, as in real-life applications, it will be difficult for the user to get the exact value of nose = 1 to signify the end of each nod. Similarly, the values used to trigger an action should be significant enough that it can be considered a

deliberate action of the user. For example, if nose > 1 is used to measure a nod, minuscule head movements can be counted as one nod even though it is not the intention of the user.

Background information

Computer vision (CV) makes it possible to communicate with a machine by gestures. An e-book reader with a clever CV control system enables handicapped persons, who cannot use their hands, to turn pages by head movements. For programming languages there exist special libraries like OpenCV supporting CV. These libraries contain special commands that make it possible to detect parts of a face like the eyes or the bridge of the nose in a camera image.

Beaver Sort

There is a bunch of trunks in the river, all of different sizes. Beaver Hamid's task is to sort the trunks by size. Hamid moves along the riverbank, always taking a position between two trunks. He compares these two trunks by size and swaps them if necessary.

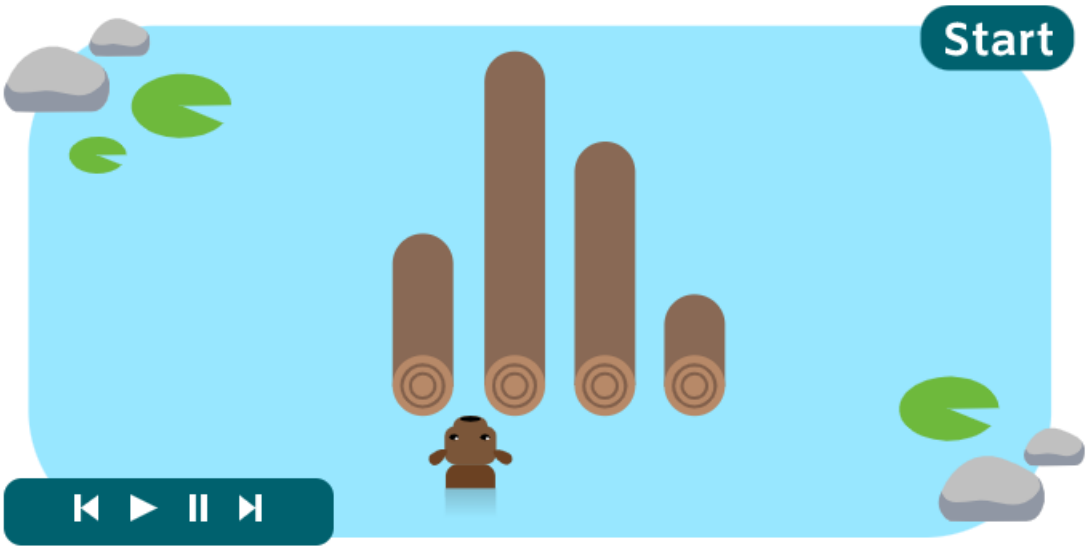
Hamid knows he can sort trunks in the following way, no matter how the trunks come in initially:

Start at the position on the right of the leftmost trunk.

Repeat until you are on the right of the rightmost trunk:

- If the trunk on the left is smaller than the right trunk: move to the right by one trunk.
- If the trunk on the right is smaller than the left trunk:
  - swap these trunks;
  - unless you are in the starting position: move to the left by one trunk.

See how Hamid sorts 4 trunks this way. In this example, he has to make 9 moves.



The number of moves Hamid has to make to sort a bunch of trunks depends on how the trunks come in initially. In the worst case, Hamid would have to make 30 moves for sorting 6 trunks.

Question

Now Hamid must sort 60 trunks. The number of moves he has to make can be expected in which range? Click on the correct answer.

Explanation

A. [0...30] is wrong. Even in the best case, if the trunks are sorted from the beginning, Hamid has to make 59 moves, which is already greater than 30.
B. [6...70] and [59..300] are also wrong. To prove it, we need to explore the worst case, when trunks are sorted in the opposite order. In this case, Hamid reaches a trunk on the k-th position and moves it to the first (leftmost) position, then goes for a trunk on the (k+1)-th position. So for the trunk on the k-th position, we need to move k-2 times right to reach it and k-2 times left to put it in the beginning. Thus, we obtain the sum 2(1+2+...+58) and we need to add 59 moves from the leftmost to the rightmost position at the end of the algorithm. The sum is exactly 592=3481. This number does not belong to any of these two ranges.
C. [59...3600] is correct. To see it, we need to prove that the worst case is really the worst one. When Hamid reaches the trunk on the k-th position, all previous trunks are already sorted properly, so he needs only to put this new trunk in the correct position among the previous ones. Then he goes to the trunk on the (k+1)-th position. So, the smaller is the trunk in k-th position, the greater number of moves it requires.

Background information

In Computer Science, sorting algorithms are used to put a sequence of objects in a certain order. The most frequently used orders are numerical order (for numbers) and lexicographical order (for all kinds of data based on an ordered alphabet). Efficient sorting is important for optimizing the efficiency of other algorithms, such as search algorithms that require input data to be sorted. Also, sorting can be useful for canonicalizing data and for producing human-readable output.

One of the known sorting algorithms is the gnome sort. It is conceptually simple, and it works with one item at a time but gets the item to the proper place by a series of swaps. If the list initially is almost sorted, it works with n swaps for n objects. Gnome sort (dubbed "stupid sort") was originally proposed by the Iranian computer scientist Hamid Sarbazi-Azad (professor of Computer Science and Engineering at Sharif University of Technology) in the year 2000.

While speaking of an algorithm we always need to think how fast it is, i.e. which number of operations it requires in the worst case, depending on the number of elements. For this algorithm, if we have n trunks, we need approximately n^2 objects. We call it quadratical relation. Another answer in this task represents another possible relation: constant (independent) relation for [0..30], linear relation for [6..70] and log-linear relation for [59..300].

This relation is called *algorithm complexity* and is studied by computational complexity theory.

Preferences

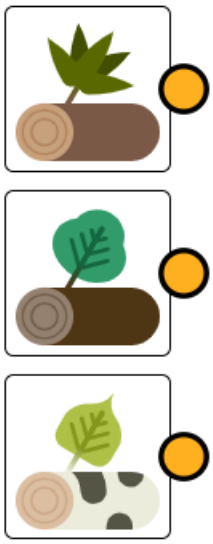
The beaver family has three gifts for their young beavers, one gift for each.
Each young beaver indicates which gift it prefers first and which one it prefers second.

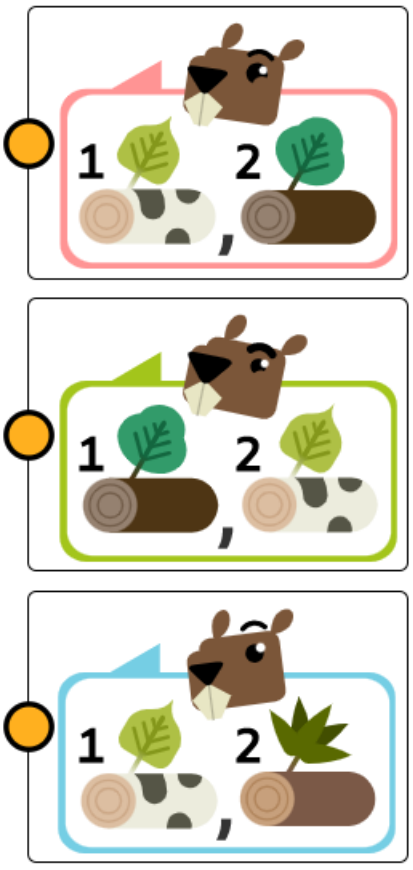
The family wants to assign the gifts in the best way:
They want to satisfy as many first-place preferences as possible,
and then as many second-place preferences as possible.

Question

See the gifts on the left and the beavers with their preferences on the right.
Assign the gifts in the best way, by dragging a line between the gifts and beavers.

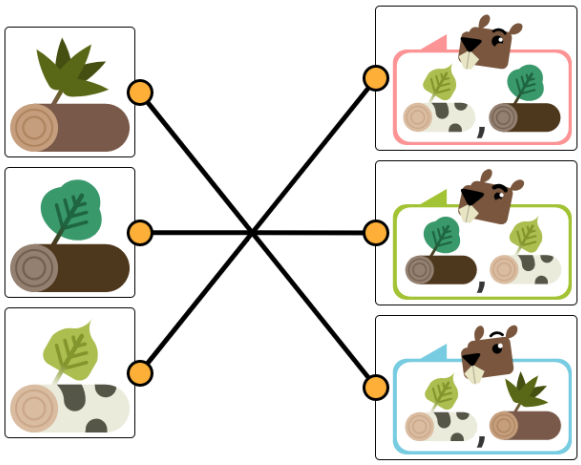
Click on ‘Save” when you are done.





Explanation

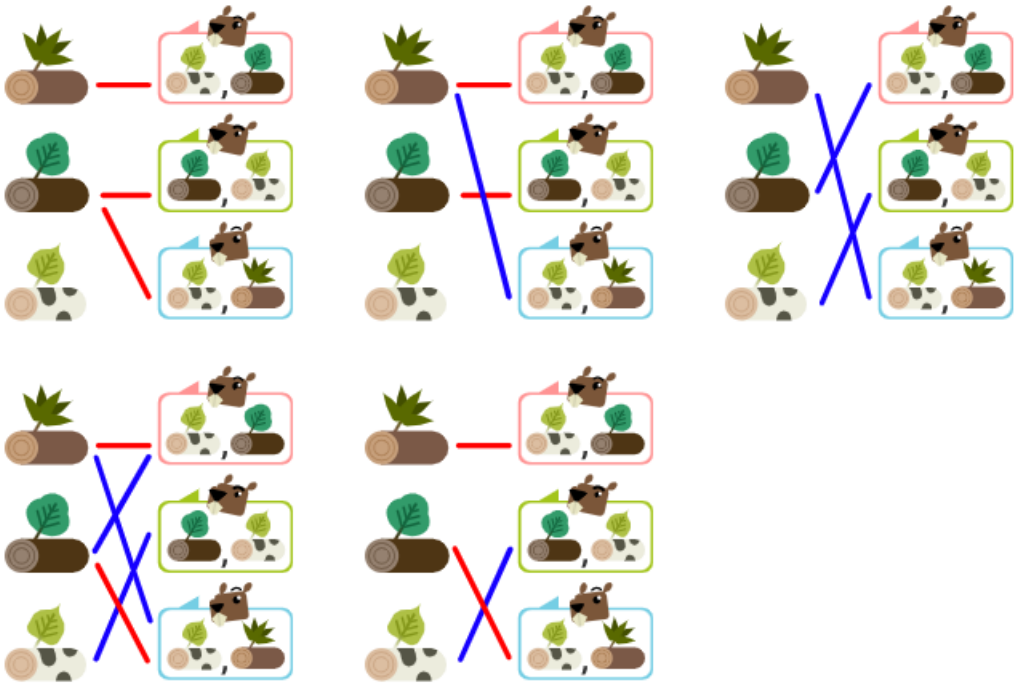
See below for an assignment of gifts in the best way.



We cannot satisfy all first-place preferences, because two young beavers have the same first-place preference. The assignment shown above satisfies two first-place preferences and one second-place preference. For this case, a better assignment is impossible. Note that if going from top to bottom, you assigned the second gift to the second beaver (just like the first gift to the first beaver), you could not assign a preferred gift to the third beaver anymore. That is, in this problem it is not sufficient to be "greedy" and take the next best thing one by one.

Background information

What algorithms are used to solve such problems?
You can try all methods of distributing gifts and choose the best one. To do this, you need to consider 6 options: some of them will leave some beavers without gifts, and from the rest choose the best one. But if there were 10 beavers and 10 gifts, then one would have to go through more than three million options.. It would take humans several months.
Informatics specialists are often faced with the task of correctly organizing the enumeration of options. For example, in our problem, we can first consider only those gifts that the beavers want to receive in the first place. In the picture on the left, they are marked with red links. It is immediately obvious that the second and third beavers want to receive the same gift.



Since we can satisfy the desire of only one of these beavers, consider two options:
1) when the second beaver receives the second gift and
2) when the second beaver does not receive the second gift.
For each of the cases, a picture is shown, in which the gifts that the beavers want to receive in the second turn are indicated in blue. Moreover, in the first case, the blue links can be removed from the second beaver and the second gift, since the gift has already been assigned to the second beaver. In the second case, there are still many options, but we can do the same, breaking all the options into those in which the third beaver receives the first gift and those where he does not receive it. By splitting the options into two parts, we will quickly find all possible distributions of gifts, so that later we can choose the best one. This algorithm in Computer Science is called the Divide-and-conquer algorithm.

Error Detection

In Beaver City, electricity is produced by windmills on the hills and it gets carried to the houses through the network below.

Some links are faulty, though: the two houses with the lights off don't have electricity anymore! All others do. Electricity can be carried from house to house in any direction.

Question

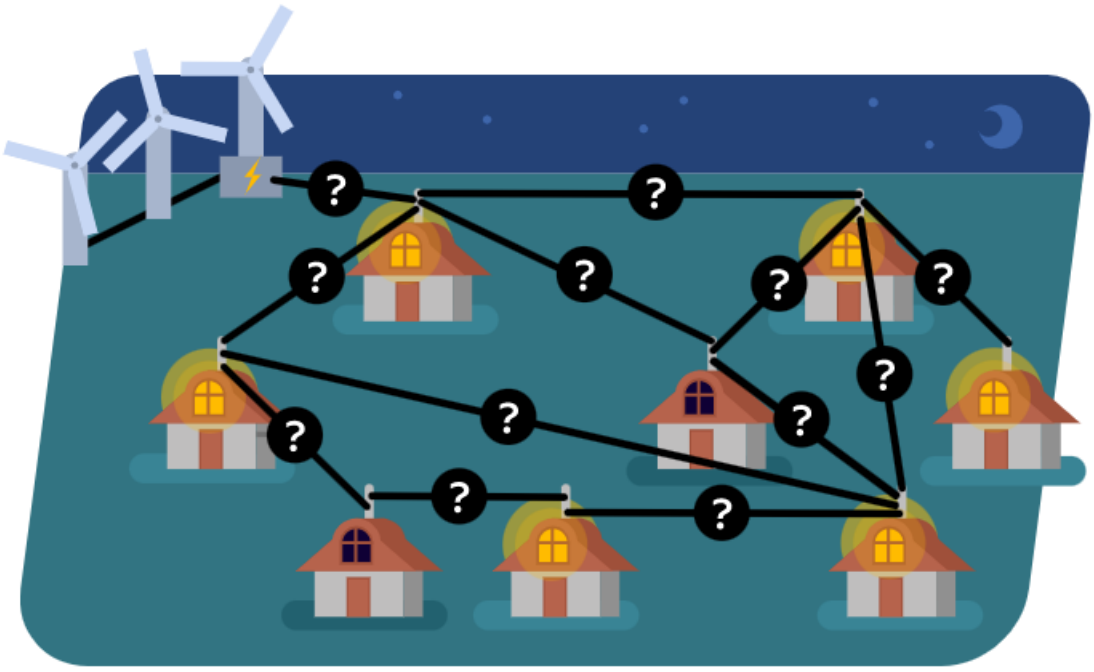
According to the current state of the houses, indicate for each link in this electricity-distribution network if

(1) it is known to be faulty ( ),

(2) it is known to be working ( ), or

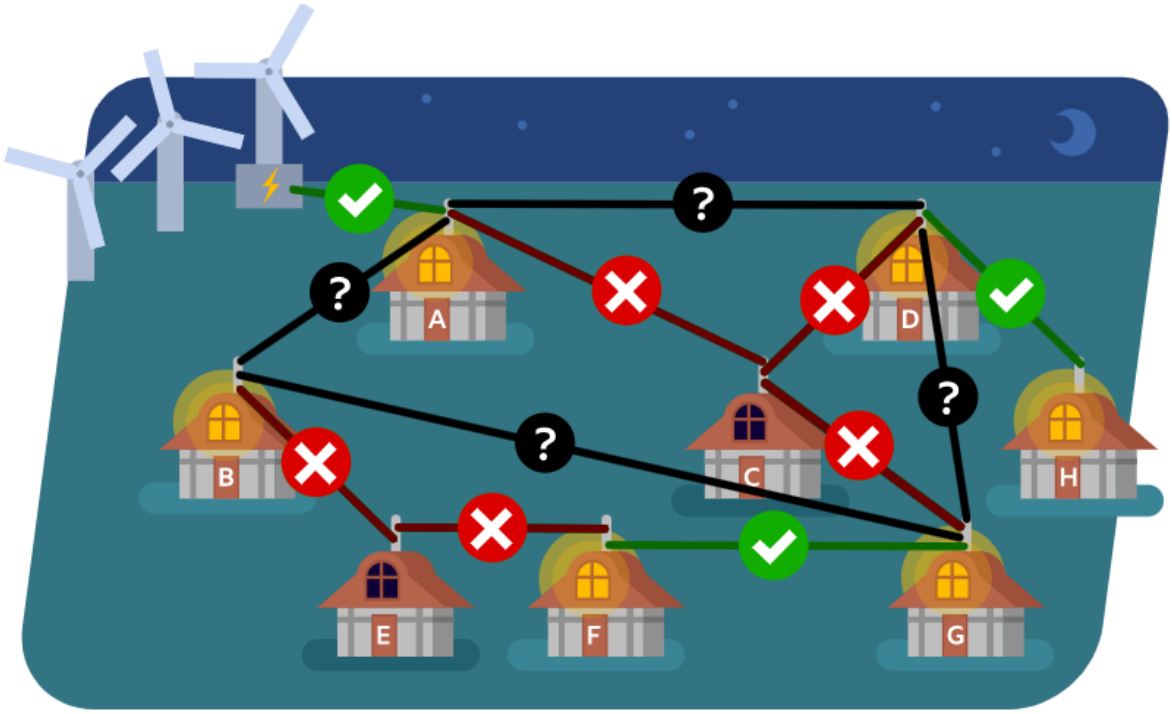
(3) we cannot tell, without more information, if it is faulty or working ( ).

Click on each link (several times if needed) to determine its state. Click on 'Save" when you are done.



Explanation

Here is the map showing what we know about the links in the electricity-distribution network:



The first thing we know is that the 2 direct links to house E and the 3 direct links to house C are all faulty. As all neighbouring houses have electricity, a working link would have brought electricity to houses C and E as well.

Next, links that are alone in providing electricity to houses where the lights are on cannot be faulty, otherwise, no electricity could arrive there. This is the case for the link leading to house H and the link from house G to F. The link from the windmills to house A must also be working, otherwise, no one would have electricity at all.

The remaining houses, B, G, and D, are multiply connected to house A. For instance, B can get its electricity directly from A, but it could also get it from G if the link to A was faulty. The same can be said about D. Finally, G can get its electricity either from B or from D. One of the links in the A – B – G – D – A cycle could thus be faulty and these 4 houses would still get electricity.

Background information

In computer networks, just like in electricity-distribution networks, some links can be faulty— slow, overloaded, or outright broken. Having redundancy in the structure of a network ensures its continuous availability in case of faults (provided there are not too many faults at the same time).

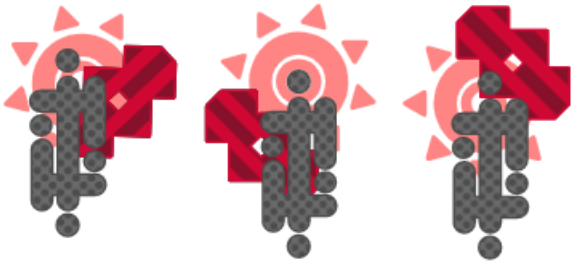
To represent network structures, computer scientists use the notation of graphs. A lot of algorithms exist to work with graphs in order to, for instance, determine a faulty link as efficiently as possible given the network structure.

Fixing errors in a system is a task computer scientists very often have to do, not only in computer networks but also in software development. To fix an error, one has to identify its precise source, and this process is usually done gradually in several steps. Some programmers believe you never can find all errors and bugs in a program.

Stamping

Casey has 3 stamps:

Each stamp makes a different coloured pattern and can be stamped in any direction.
Casey makes the three patterns below by always using the stamps in the same order.



Question:
Which stamp did Casey always use first? Click on the correct answer.

Explanation

Sun	Stripes	Blocks
		

The sun stamp is “under” the blocks stamp and the stripes stamp. The stripes stamp is at the top: it overlaps the blocks stamp and the blocks stamp overlaps the sun stamp. So the sun was stamped first, and the stripes last in each pattern.

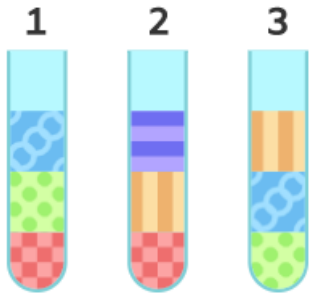
Background information

There are many computer programs and apps where you can draw or manipulate pictures. Most of them use layers. With layers, you can define the order of image parts. The pictures on the bottom layer will be overlapped by the pictures on other layers. Of course, the layers can be in different sizes and they can have a transparent background.

In animation, where you use more frames (pictures) after each other, the bottom layer can contain the background – and it can be copied to the next frame. You can manipulate the layers separately. So it is easier to modify, copy or delete a smaller part of the picture.

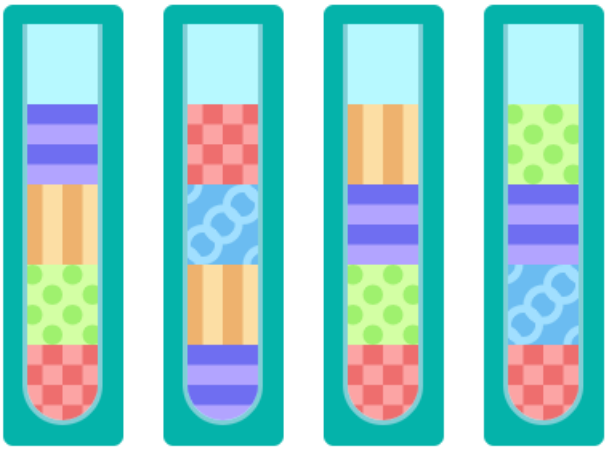
Colourful Tube

Beaver Mark wants to fill a tube with various types of liquids. His Science teacher explained that a liquid with a higher density will be under a liquid that has a lower density. The teacher gives Beaver Mark 3 examples of tubes filled with all available liquids in the laboratory:



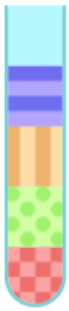
Question

Beaver Mark intends to fill a tube filled with 4 different liquids from the laboratory. Which tube could Beaver Mark have filled? Click on the correct answer.



Explanation

The correct answer is:



The order of all liquids according to their density is as follows:

Background information

The questions show a repeating pattern that requires students to identify similarities in the component parts of a problem (Pattern Recognition). Pattern recognition is simply looking for patterns in the problem and using any of the problems or solutions that have been encountered in the past to apply in the new situation. What we have learned in the past could help us to solve the new problem.

Ordering is putting things into their correct place following some rule. In the questions, the rule for ordering liquids is determined by the density of each liquid.

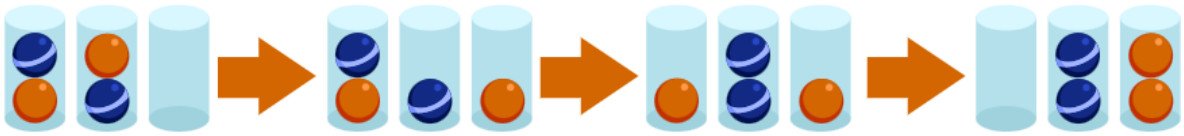
A constraint is a condition of a problem that the solution must satisfy. In the questions, the density of the liquids becomes a constraint for ordering the liquids.

Moving the Ball

Beavers are playing a game. The aim of the game is to move the balls so that the same balls are in the same tubes.

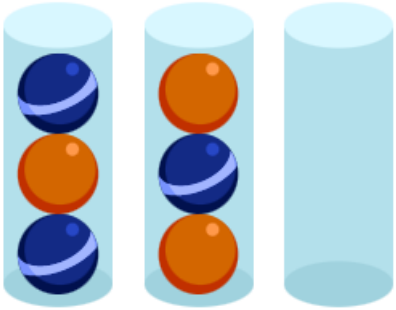
Game rules:

- Rule 1: A ball can be moved to an empty tube.
- Rule 2: When there is a space in a tube, a ball can only be moved on top of a ball of the same colour.
- Rule 3: Only one ball at the top of a tube can be moved at a time.



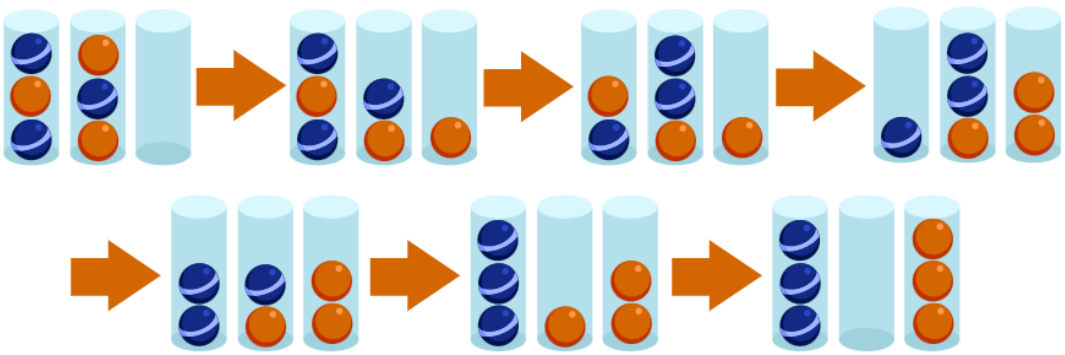
Question

Drag the balls from tube to tube, obeying the rules of the game, until the same balls are in the same tubes. Do so in the minimum number of moves. Click on ‘Save” when you are done.



Explanation

The balls are classified in the following order. Although the order is different, there may be other ways to classify the balls in just 6 steps.



Background information

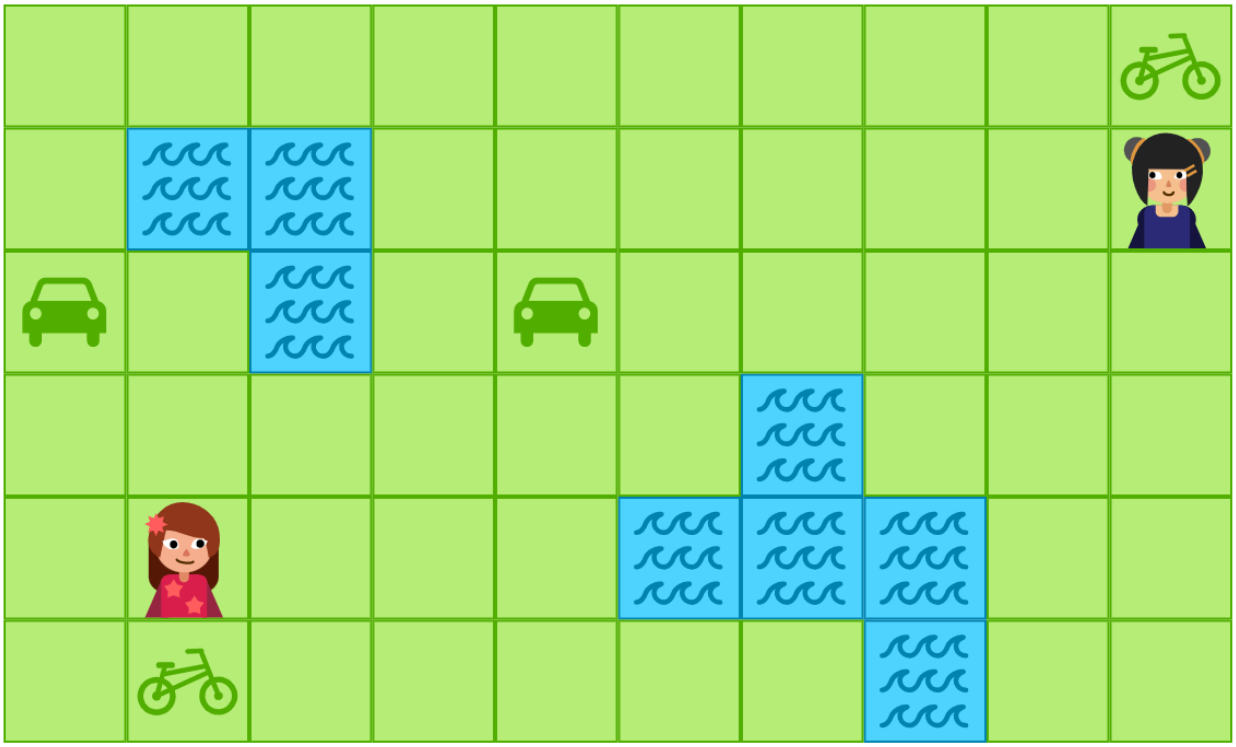
In Informatics so-called stacks can be used for storing data. The special characteristic of stacks is that you can only access and remove the last added data.

The cylinders in our example behave like stacks: You can only add a ball on the top of a cylinder. And you can only remove the ball from the top. The other balls cannot be accessed.

Meeting race

Two friends need to meet urgently - see the map below. They can walk from a square to a horizontally or vertically adjacent square in exactly one minute. If they reach a bike or car they can use it to travel faster – 2 squares in one minute with a bike, 5 squares with a car. They cannot travel over water.

You can use the picture below to help you solve the question. Click on the squares to toggle between the two friends.



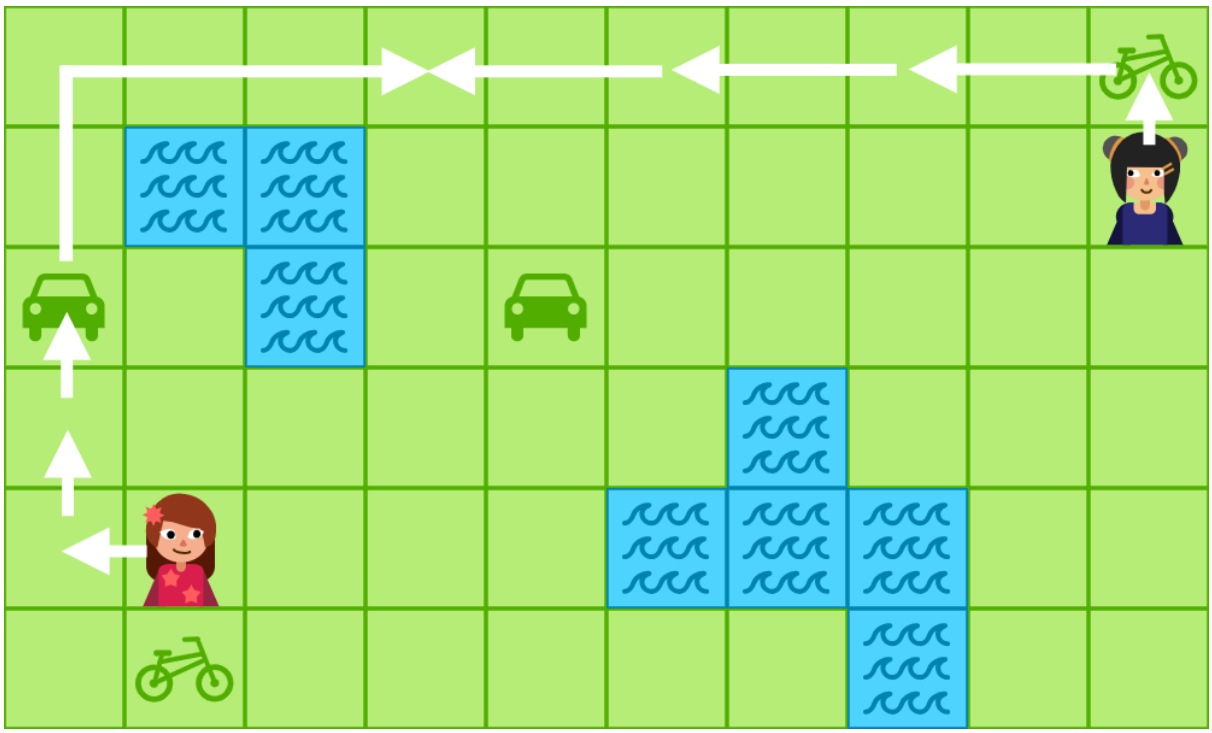
Question:

What is the minimum number of minutes they need to end up on the same square? Fill in the number below and click on ‘Save” when you are done.

Answer:

Explanation

The correct answer is 4. This can be achieved by the route shown below:



(Another option is to take the leftmost bike and cycle to the leftmost car and then continue as above.)

To see why 3 minutes are not sufficient, you can reason as follows.

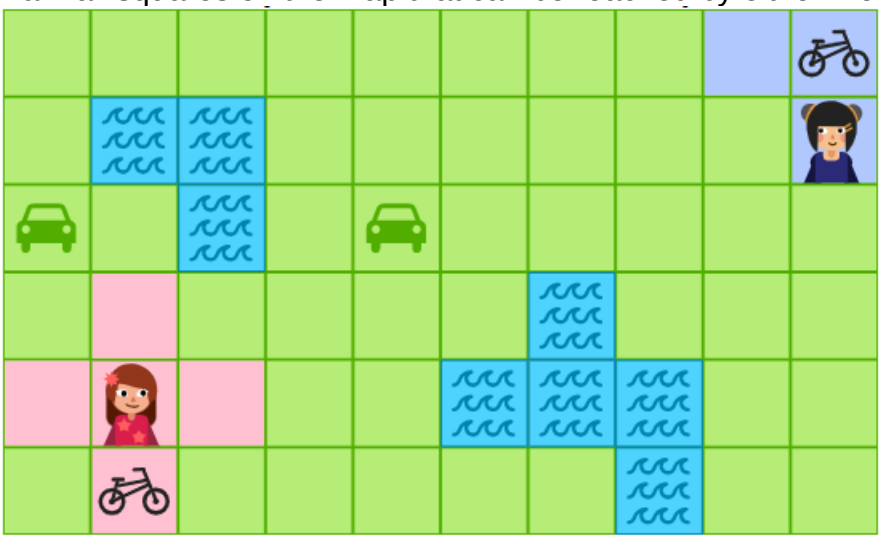
- Although in 3 minutes you can reach the car on the left, there is no time left to drive it anywhere. And that position cannot be reached in three minutes by the other person. So the cars are of no use and we may as well remove them from the map.
- The two friends are more than 5 minutes away from each other on foot, so they need a bike. In fact, both need a bike because they are separated by more than 9 positions. But finding that bike costs one minute and with only two minutes left they cannot reach each other, even by bike.

Background information

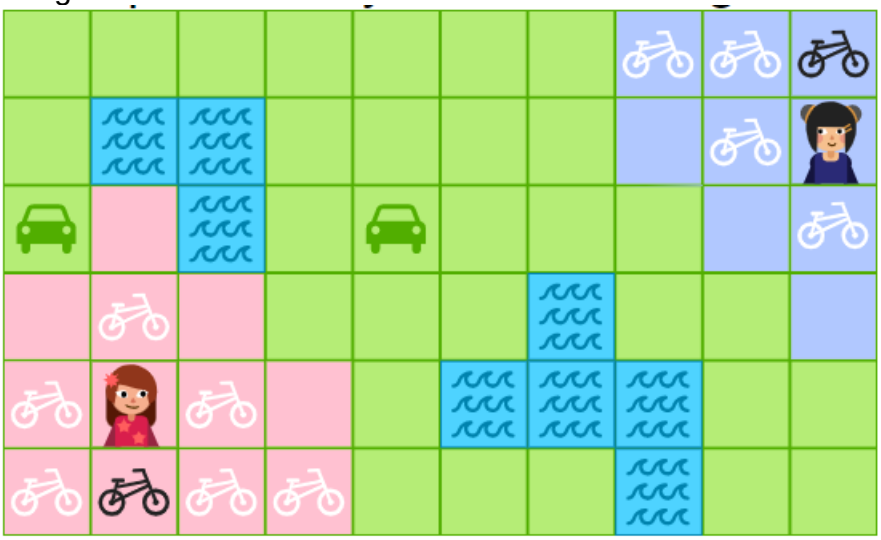
It's Informatics

How did you go about solving this task? Did you find a short route by accident and hoped that no shorter route could be found, or did you try out dozens of different possibilities and remember the shortest time? A computer program that is designed for this kind of task would use a systematic approach, most probably using an algorithm called '**breadth-first search'**. For this task, this would be done as follows:

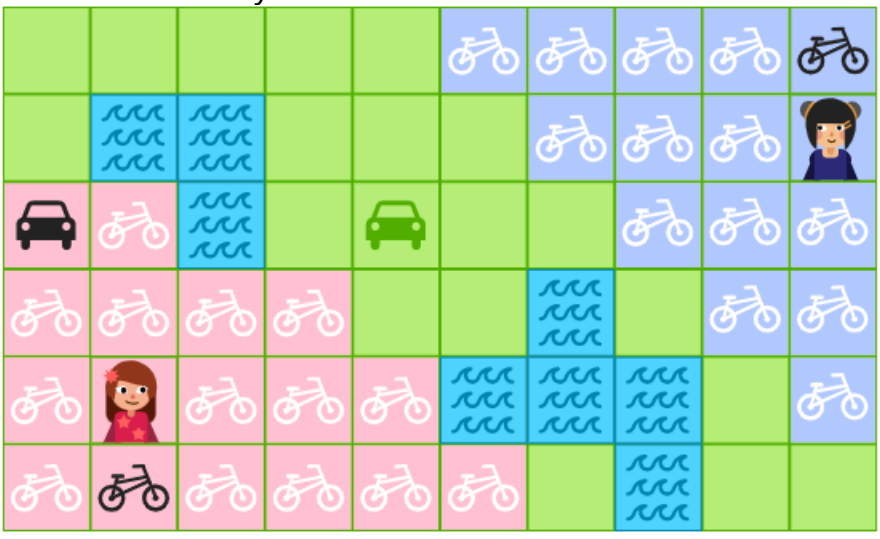
1. Mark all squares on the map that can be reached by either friend in one minute.



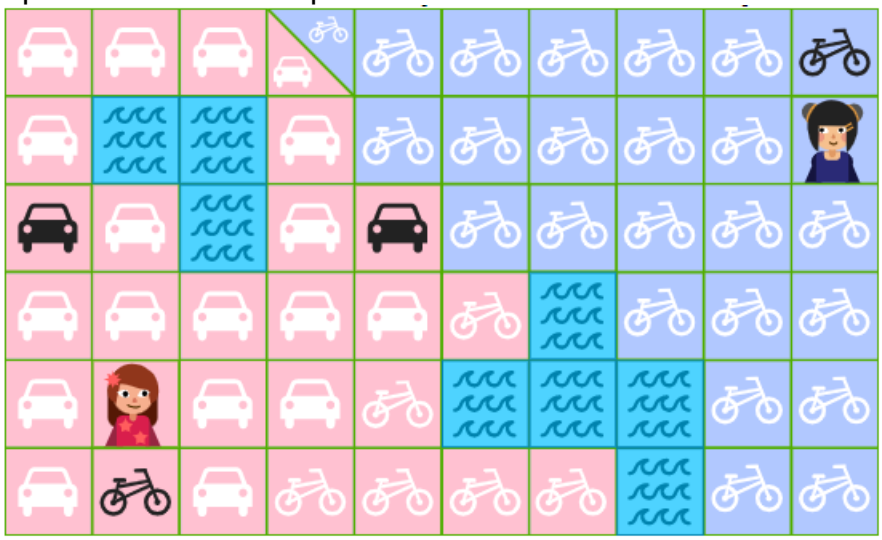
2. Mark all squares that can be reached in (at most) one minute from the positions you marked in step 1 and keep track of which kind of transportation you were using.



3. Mark all squares that can be reached in one minute from the squares marked in step 2. Because the two areas that we have marked do not overlap, you see that the friends cannot yet reach each other in 3 minutes.



4. Do one more step: mark all squares that can be reached in one minute from the squares marked in step 3.



Now the two regions that we have marked overlap (in one square) indicating that after 4 minutes the two friends are able to meet.

You are probably familiar with software that finds the fastest route between two places on a map taking care only to follow roads and not drive through mountains and rivers. This task is very similar but now two persons move towards each other instead of one person towards a fixed position.

Because of the systematic way in which the search for a solution is done, a computer will often find solutions that are not obvious at first – sometimes a detour with fewer traffic lights can be a better option than a direct road, or a public transport route with several quick transfers turns out to be faster than a direct bus.

In Computer Science various methods are known for finding the best solution to a problem like this. Apart from the depth-first search method described elsewhere, there is also the **branch and bound technique**, which is quite similar but uses well-reasoned shortcuts to speed up the search: if we have already found a pretty good solution then we may discard options that we know cannot produce a better solution than the best one found so far.

When a problem becomes too complex, going through all possible solutions to find the best one will take too long even for a fast computer. And in practice, it is often sufficient to find a very good answer even if it is not the best possible. (If you can reach your destination in 78 minutes you are probably not much bothered by the fact that there is another route that could have taken you there in 77.)

One technique that is used in that case, is the more intuitive **greedy algorithm**, which at each step chooses what seems optimal at that time and does not look ahead at what could happen in further steps. In our task that would mean that the two friends always take a step that brings them closer together, which in this circumstance is not a good strategy because then they would go on foot most of the way. There are however other types of problems in which this greedy strategy produces reasonably good solutions. And it is much faster than the other methods.

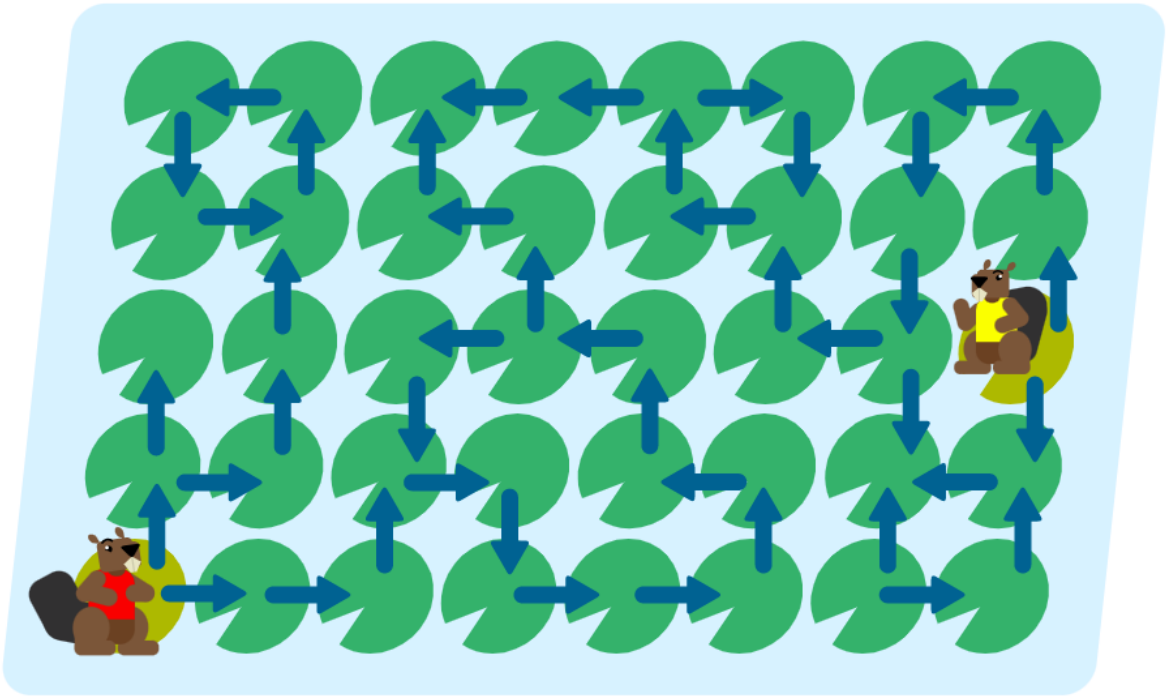
Do they meet?

On the lake, beavers can go from one lily pad to another but only in the direction the arrows indicate. Bob (in a red vest) and Nora (in a yellow vest) would like to meet on one of the lily pads. They start on different lily pads as you can see below.

Question

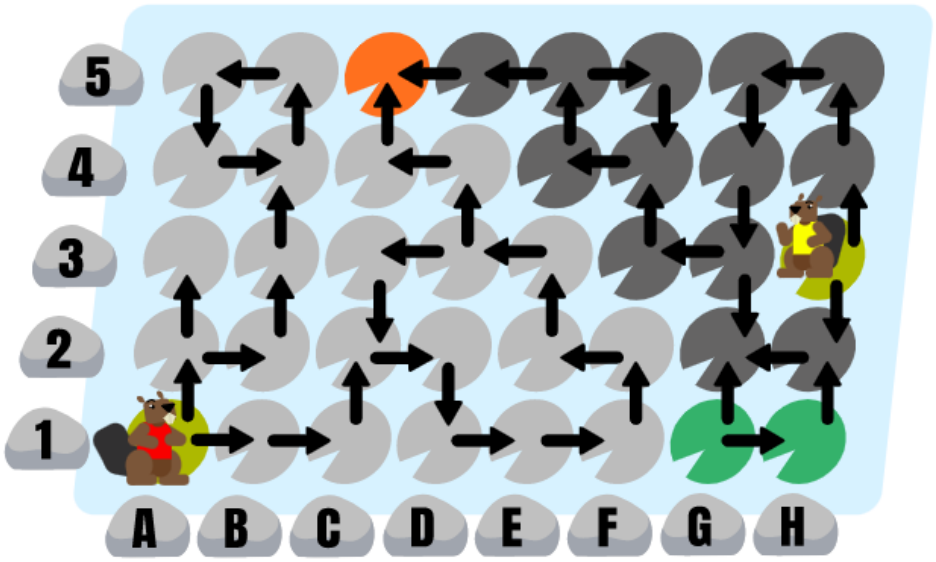
Click on the lily pad where they can meet if they follow the arrows.

Click on ‘Save” when you are done.



Explanation

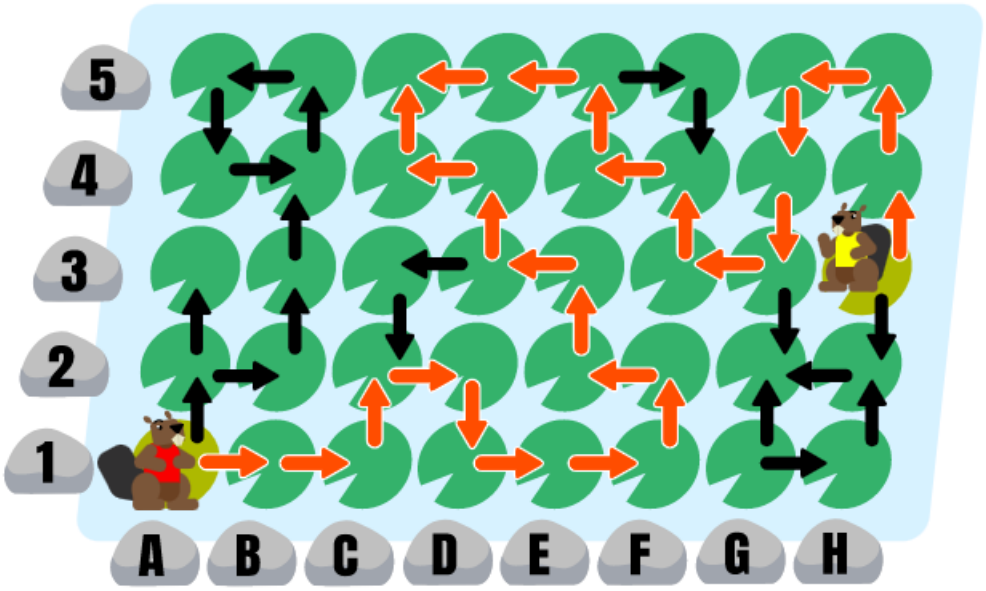
The beavers may meet on C5.



At his starting position, Bob has two options: If he goes “up”, then he may either run into the dead-end at A3 or get stuck in the loop that begins at B4. If he goes “right” (to B1), he can continue to D3. At D3 he may either go “left” into a loop that will take him back to D3 eventually or go “up”, which makes him end up at C5, another dead-end.

Nora also has two options at the start. If she goes “down”, she will run into the dead-end at G2. If she goes “up”, she will reach G3. From there she may either run into the G2 dead-end again or go “left” and reach E5 eventually. There she may either go into a loop that will take her back to E5 again or reach another dead-end at C5.

As we already know, Bob may reach C5 as well, so we can see that they may meet at C5. The picture shows the ways along which they both can reach C5.



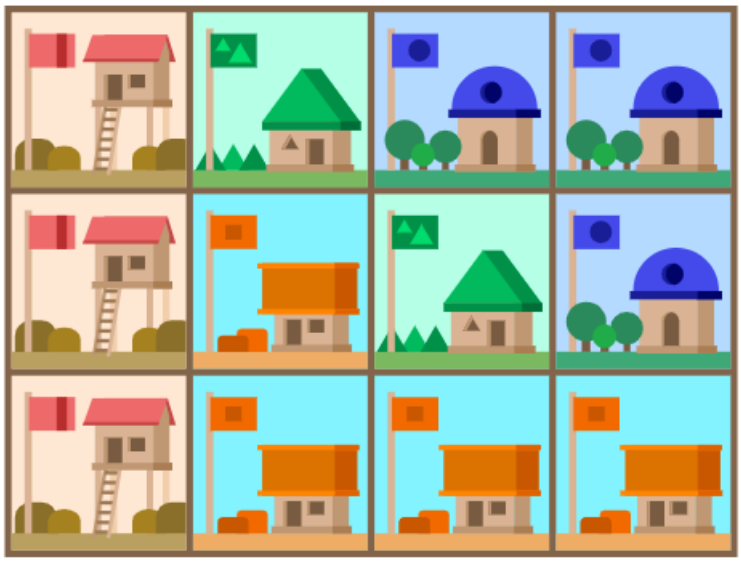
But this does not yet fully guarantee that they cannot meet at F4 or C5 either. The next picture shows the set of pads that Bob (white) and Nora (dark grey) may reach by following the arrows in any possible way. We can see that C5 is the only pad common to both sets.

Background information

Let us take a closer look at how we created the last picture. For each beaver, we followed the arrows. If there was a dead-end, or a loop was detected, we went back to the most recent fork, in order to choose another option to proceed. Thus, we followed all possible ways. A very similar procedure is often applied by computer scientists when solving difficult problems. The procedure tries to construct a solution step by step. Often, there are several options to choose the next step. Then, the procedure will choose one option but keep track of the other options as well. When it runs into a dead-end (and a loop, if detected, is a dead-end as well), it goes back to the most recent choice and tries another option. This algorithmic approach to problem-solving is called *backtracking*; this name gives a pretty good indication of how the approach works.

Unification

In the far land of Beavaria, there lived four tribes in several villages Each tribe has its own flag as shown in the picture: red stripes, green triangles, orange squares and blue circles. One day, they all decided to unify. However, in order not to cause chaos in the land of Beavaria, it was decided that only two tribes can be unified at the same time. The time needed to unify two tribes in months is equal to the number of villages of these two tribes. After this, the two tribes become one single tribe, and the unification process is repeated until there is only one, unified, tribe remaining.



Question

What is the minimum number of months needed for the tribes to unify? Click on the correct answer.

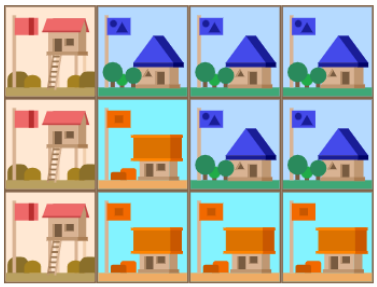
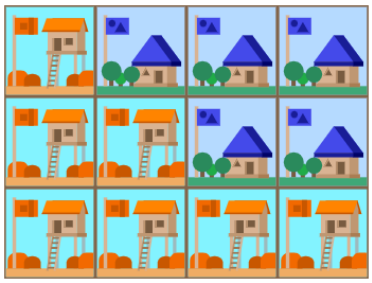
23	24	25	26	27

Explanation

The correct answer is 24.

The optimal strategy to minimize the total number of months needed to unify all the tribes is to minimize the number of times each village is included in the unification process. Hence, we can see that the largest tribes should be added last, as then the largest number of villages will only be added the least number of times. In order to do this, each unification step should happen with the two tribes having the fewest villages.

This is illustrated in the table below:

1. green triangles have the least number of villages, so they will be chosen for unification first. Since two tribes have 3 villages, we can choose to unite either one, for example, green triangles and blue circles. After unification, they can be called blue triangles and circles.	2. Now the ones having the least number of villages are orange squares (4) and red stripes (3). After unification, we can call them orange stripes and squares.	3. Lastly, 5 blue triangles and circles and 7 orange stripes and squares villages unite into one large orange shapes tribe.
		
This takes 5 months and results in 3 red stripes, 4 orange squares and 5 blue triangles and circles villages.	This takes 7 months and results in 5 blue triangles and circles and 7 orange stripes and squares.	This takes 12 months.

Therefore, the minimum number of months to unify all four tribes from the land of Beavaria is 5+7+12=24.

Background information

This challenge is an example of an *optimization problem*, a task whose goal is to come up with a strategy that maximizes or minimizes a certain quantity, subject to some constraints. Optimization problems are ubiquitous in our everyday lives: finding the shortest route to a destination, creating a schedule that accommodates the greatest number of non-overlapping activities, and so on. There are several ways to attack an optimization problem, and these include *greedy algorithms*.

Greedy algorithms rest on the assumption that making the best choice at each stage (*local optimum*) will result in the best final outcome (*global optimum*). In this problem, this assumption is satisfied: tribes have to minimize the number of months for each unification in order to minimize the number of months for the entire unification process. It must be emphasized, however, that the greedy paradigm is not a universal solution to all types of optimization problems. Nevertheless, it usually provides a decent approximation within a reasonable time.

Bridge Construction

Beaver Mark is a bridge builder. He needs a hammer  , nails  , planks  and ropes  to build one. In his village, there is a river that has no bridge, so he decided to build one so that all beavers can cross the river safely.

After checking his warehouse, he realized he already has a hammer  and ropes  , but he doesn't have any of the other items he needs for the bridge.

Luckily, there are 3 stores in his village where he can get the things he wants:

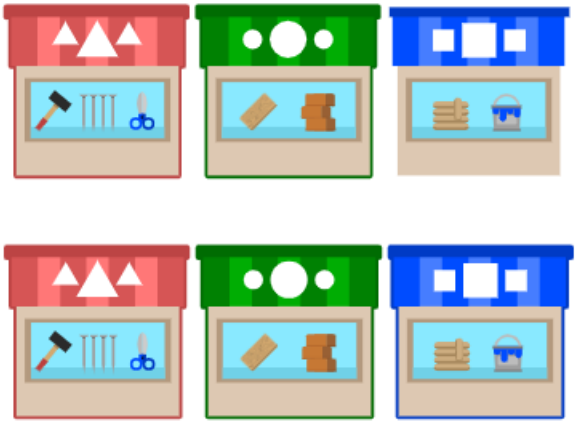
- Triangle's sells hammers, nails and scissors
- Circle's sells planks and bricks
- Square's sells ropes and paint

Question

If Mark wants to visit the fewest number of stores, which stores does Mark need to visit in order to be able to build the bridge? Click on the stores to select them. Click on 'Save" when you are done.

Explanation

The correct answer is triangle's and circle's:



is wrong because Mark doesn’t need to visit Square's because he already has ropes and he doesn’t need paint.

is wrong because Mark needs to visit triangle's as well because he needs nails but doesn’t have them.

is wrong because he needs to visit triangle's as well, and he also doesn’t need to visit square's, for the reasons stated in the paragraphs above

is correct because the only items he needs but doesn’t have are nails and planks: he needs to get nails from triangle’s store and planks from circle’s store.

Background information

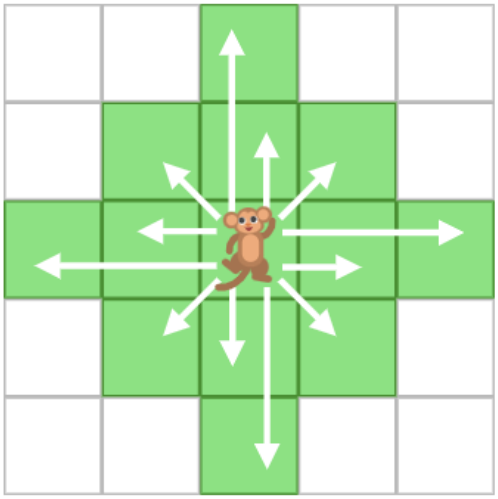
This problem conceals how a microservice architecture works: a big application (the village) may offer a lot of resources (procedures), but these resources are grouped in smaller entities called microservices. When a process (Mark) needs some procedures in its logic, it doesn’t have to implement them itself (Mark doesn’t need to craft his own

items), but it requests them from other services (in this case, Mark buys the resources from other stores).

This problem asks what the stores are that Mark depends on. In real life, it is a good practice to check which services a process depends on so that when something doesn’t work you can trace all affected processes in one way and trace the root cause in another way so that you can mitigate the damage caused by unavailable services.

Jumping Jack

Jack is a monkey living in a park. From one tree, he can jump to another tree if it's either up to two cells away horizontally or up to two cells away vertically, or one cell away diagonally, as shown in the diagram below.

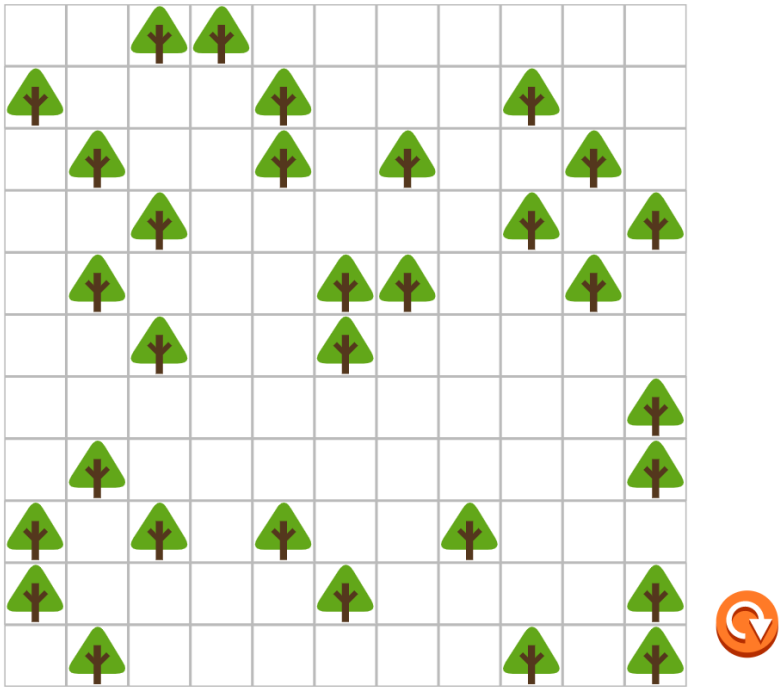


Jack plays a game in which he jumps to as many different trees as possible without touching the ground. He can start from any tree in the park. Map of the park: (insert the map of the park here?)

Question:

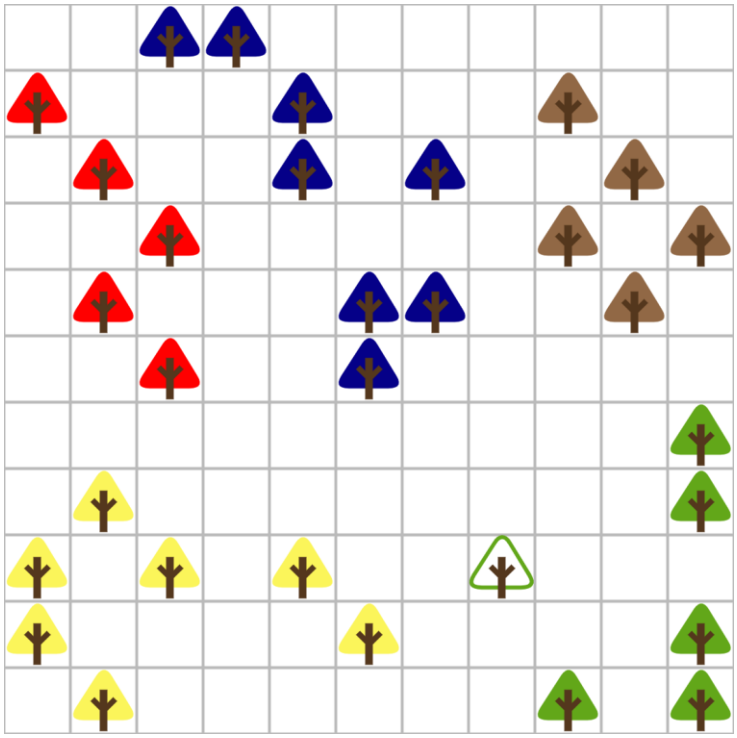
Colour in orange (square), the biggest number of different trees Jack can visit in one go without touching the ground. Click on a tree to change its colour. It will flip between green (triangle), orange (square) and red (round). You can use green or red to help you solve the task, but only orange (square) trees are part of the answer.

Click on ‘Save” when you are done.



Explanation

In this diagram, Jack can visit the 8 trees that are coloured in dark blue. We coloured other trees with multiple colours, to show the different groups of trees.

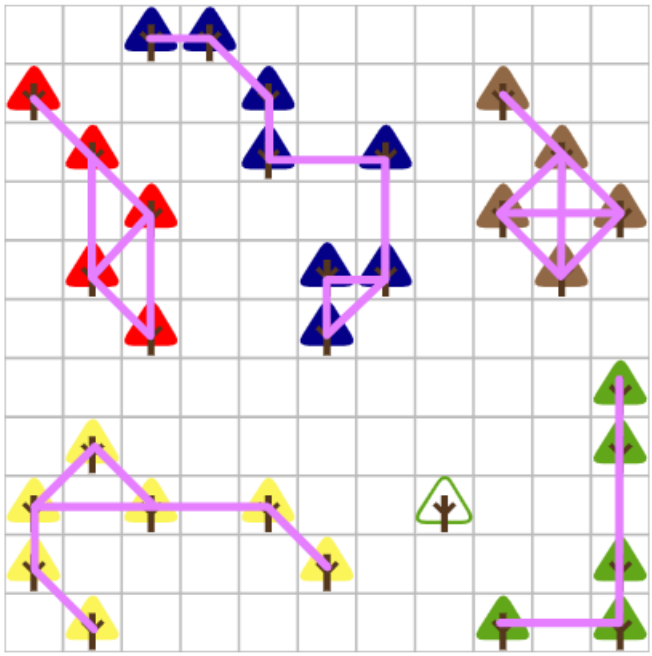


There are essentially six groups of trees in the park. If Jack starts on a tree we painted yellow, he can reach all yellow trees and no trees of other colours. How do we find such groups? Pick a random tree and colour it as you wish. Then use the same colour for all trees that are reachable from it. And all trees that are reachable from those trees, too. And so on, until you cannot reach any other trees. If there are any trees you haven't

coloured yet, take another colour and start again from a random uncoloured tree. Your colouring simulates Jack's exploring.

Background information

From the point of view of Informatics, we are manipulating a graph: the trees are called vertices, and we connect two trees with an edge when Jack can jump between these two trees. In the diagram below, we represent edges as black segments between trees.

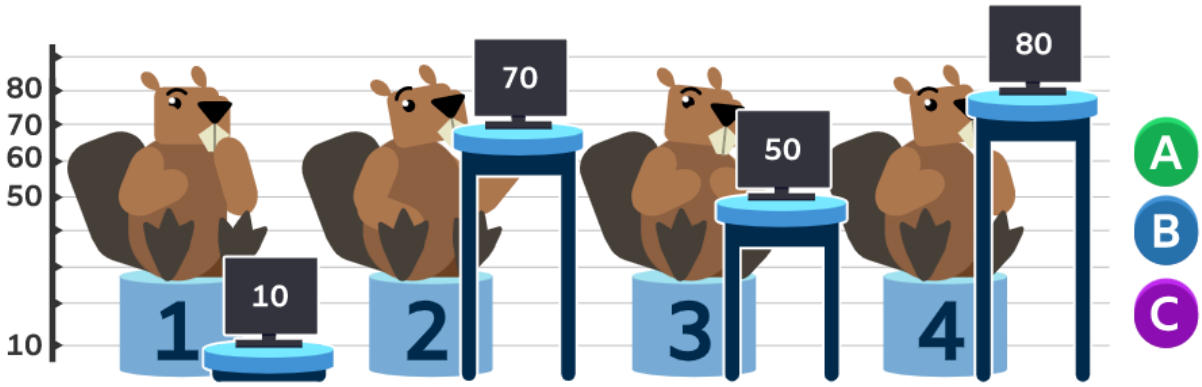


If there is a path, using these edges, that allows Jack to go from one tree to another, then these two trees belong to the same group. We call such groups the connected components of the graph. Here we used a different colour to represent each connected component.

The procedure for colouring is similar to many different graph algorithms that deal with searching. Their fancy names are breadth-first and depth-first search.

A Desk Trouble

At Miss Beaver's programming class, students adjust the heights of their desks by using an electrical system.
The recommended height of the students' desk is 60 beaver units.
Unfortunately, a naughty little beaver spilt water on the controls, causing a change in the heights of all the desks, as shown in the image below.



After the control buttons failed, these are how they function:

- The "A" button can raise desktops 1, 2 and 3 by 10 beaver units, each time it is pressed.
- The "B" button can lower desktops 2, 3 and 4 by 10 beaver units, each time it is pressed.
- The "C" button can raise desktops 1, 3 and 4 by 10 beaver units, each time it is pressed.

Question

Could you help Miss Beaver choose the correct option that could place all desks at the recommended height of 60 beaver units? Click on the correct answer.

Press "A" 3 times, "B" 4 times and "C" 2 times.

Press "A" 4 times, "B" 5 times and "C" 1 time.

Press "A" 5 times, "B" 1 time and "C" 0 times.

Press "A" 2 times, "B" 4 times and "C" 6 times.

Explanation

a. **Press "A" 3 times, "B" 4 times and "C" 2 times.**
b. Press "A" 4 times, "B" 5 times and "C" 1 time.
c. Press "A" 5 times, "B" 1 time and "C" 0 times.
d. Press "A" 2 times, "B" 4 times and "C" 6 times.

If Miss Beaver presses the "A" button 3 times, desks 1, 2 and 3 will have heights of 40, 100 and 80 beaver units, respectively. Now, if she presses the "B" button 4 times, desks 2, 3 and 4 will have heights of 60, 40 and 40 beaver units, respectively. Finally, if she presses the "C" button 2 times, the heights of desks 1, 3 and 4 will all be 60 beaver units. As all the desks are at the recommended height of 60 beaver units, the goal has been achieved.

Option b is wrong because if Miss Beaver follows these instructions, the final heights of desks 1, 2, 3 and 4 will be 60, 60, 50 and 40 beaver units respectively. Just two of the desks are at the recommended height.

Option c is wrong because if Miss Beaver follows these instructions, the final heights of desks 1, 2, 3 and 4 will be 60, 110, 90 and 70 beaver units respectively. Only one desk is of the recommended height.

Option d is wrong because if Miss Beaver follows these instructions, the final heights of desks 1, 2, 3 and 4 will be 90, 50, 90 and 100 beaver units respectively. None of the desks has the recommended height.

Background information

In programming, we look for computers that solve specific tasks through the instructions we give them.

Often, we need computers that perform particular actions in some cases and other actions in other cases.

Conditional sentences allow the computer to choose which processes it needs to act based on different parameters indicated.

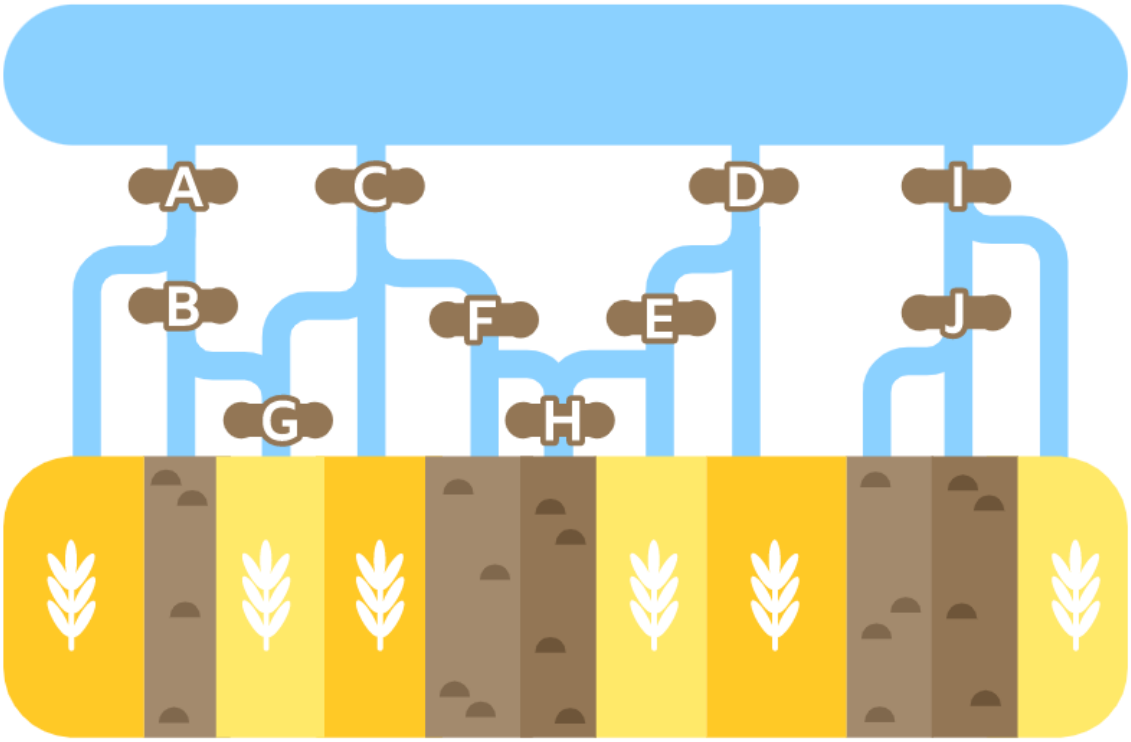
Farmer Beaver

Beaver Mert grows wheat in the fields shown in yellow on the map below. He also has fields where nothing grows shown in brown . To save water, Mert only wants to water the wheat fields . He can block the water channels coming from the lake at the spots marked with the letters A to J.

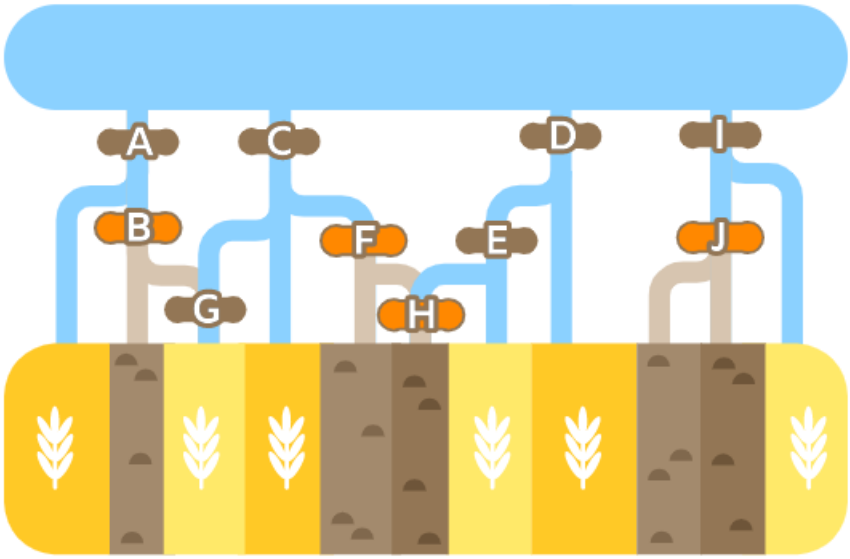
The water will only flow downwards towards the fields and will never flow back towards the lake.

Question

At which spots must Mert block the channels so that all wheat fields are watered, but none of the empty fields is?
Click on the letters to select them. Click on ‘Save” when you are done.



Explanation



If the gates there are not closed, the water will reach the fields with weeds. If more gates are closed, then some wheat fields will get no water. If we review all spots:

- A must be open to irrigate field 1
- B has to be closed to avoid irrigating field 2. It also brings water to field 3 – but this field can also get it through C
- C must be open both for field 4 and for field 3, which cannot be watered though A since B is closed
- D must be open for fields 7 and 8
- E must also be open field 7
- F must be closed to prevent the irrigation of field 5
- G, if closed, would only prevent field 3 from being watered and thus must be open as well
- H must also be closed even if F is also closed, as water can flow from the open D and E
- I must be open for field 11
- J, finally, prevents fields 9 and 10 from getting irrigated

Background information

In this task, water flows to the fields based on a number of *conditions*. For instance, water flows to field 7 if both D and E are open. Water flows to field 3 if G is open and either of these conditions hold: (1) C is open; or (2) both A and B are open. These types of compound conditions are formed with the *boolean operator* AND if two gates are one after the other on the same channel, and with operator OR if water can flow to the same destination from two separate channel segments. Such a condition is always either true or false. It is known as a *boolean value*, or simply as a *boolean*.

In programming, booleans are ubiquitous. Common usages include the *if statement*, found in virtually all programming languages, used to check that a certain condition is true before executing a series of instructions.

Orange juice

The beavers are playing a logic game to drink orange juice. John can drink from a bottle when both rules are met:

A) there is a bottle with less juice immediately to the left of this bottle, and
B) there is a bottle with more juice immediately to the right of this bottle.

Question

Click on all the bottles that John can drink from. Click on ‘Save” when you are done.

Explanation

The correct answer is 2 bottles.

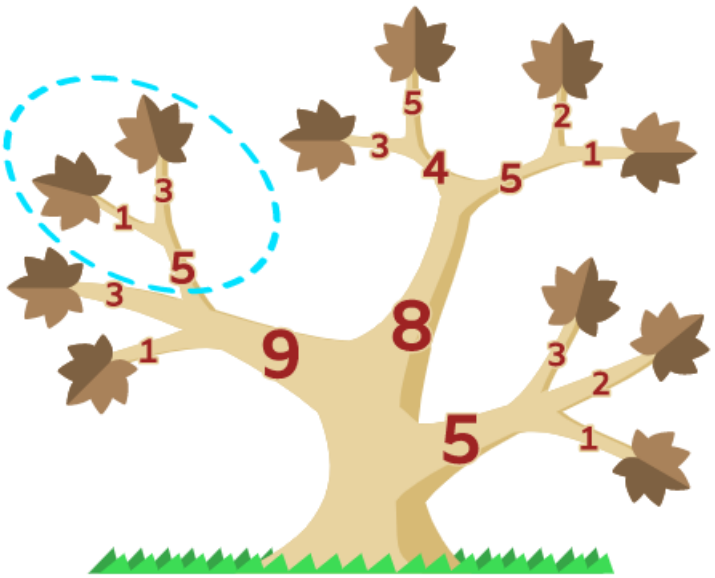
Only the bottle in fourth place and the one in seventh place meet the given conditions. Less juice on the left AND more juice on the right.

Background information

Computer science often involves solving problems that are specified by a set of logical constraints. The task is to find a solution that satisfies all of these restrictions. More complex tasks can be considered where the constraints are combined by logical operators. As conjunction (A and B means that both constraints A and B must be satisfied, as in this task) or disjunction (A or B means satisfying only one of them is sufficient). The process that the beaver is running here has close ties to scheduling, that is, a method of deciding how the bottles are chosen. The task is also to understand an algorithm and run it to predict a future state. Given an initial situation (the bottles on the shelf) and an algorithm, executing it, step by step, to know the result produced is an important activity that programmers must carry out. It is known as debugging when they have to find out a possible error, that is, something that goes wrong in the program.

Hercules and Hydra

Beaver Bruno has a tree in his garden. Unfortunately, the tree has a disease and all its leaves died and turned brown. Now Bruno needs to cut off all the branches with the dead leaves. Then the tree can grow new branches with healthy leaves.

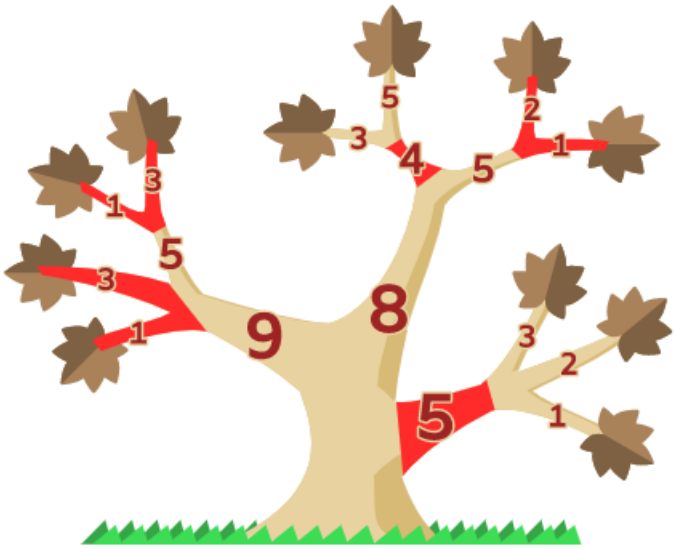


In the picture, the numbers show the time needed to cut each branch. When Bruno cuts a larger branch (for example, the branch marked with 5 inside the blue ellipse), all leaves attached to it fall down and then Bruno does not need to cut all the smaller branches (the ones marked with 1 and 3 inside the ellipse) one by one.

Question
What is the shortest time Bruno needs to cut off all branches with dead leaves? Click on the correct answer.

19 | 20 | 22 | 25

Explanation



We can view the tree (the plant) as a data structure that is also called a tree. Tree as a data structure is a type of another data structure called a graph. With this interpretation, the task is to find the minimal weight of edges separating all the leaves from the root. This is a standard problem in graph theory known as the minimal cut (or min-cut for short) and can be solved by various standard algorithms (Ford-Fulkerson algorithm, for example).

However, there is a much simpler special algorithm for finding min-cut between the leaves and the root of a tree. We can start moving from leaves towards the root and recalculate for each edge whether it's optimal to cut this edge or not.

Let's illustrate the process. Initially, we start from leaves. For each leaf we cut the branch it is on, thus the initial estimate is

1 + 3 + 1 + 3 + 3 + 5 + 2 + 1 + 3 + 2 + 1.

Now we move towards the root. In each move, we can either leave the "old" cuts or replace them by cutting a single larger branch.
The second iteration:

1 + 3 + min(5, 1 + 3) + min(4, 3 + 5) + min(5, 2 + 1) + min(5, 3 + 2 + 1) =
1 + 3 + 4 + 4 + 3 + 5.

The third iteration:

min(9, 1 + 3 + 4) + min(8, 4 + 3) + 5 = 8 + 7 + 5.

Now we reached the root of the tree, hence the final answer is 8 + 7 + 5 = 20.

Background information

Trees are important data structures in Computer Science. Moreover, some contemporary algorithms use trees, like decision trees in a random forest.

Finding the minimal cut of a directed graph allows us to also find the maximum flow in this graph, which is widely used in logistics. It helps to determine the maximum weight of goods that can be transported from the factories to other countries taking into account all means of transportation between major cities and their capabilities.

Go to the Market

Ana, Barnie and Chloe are planning a party for their family.

They need to buy the following 6 packages:

- 5 kg Rice
- 3 kg Apples
- 3 kg Sugar
- 3 kg Willow
- 2 kg Spinach
- 2 kg Corn

Each beaver can only carry two baskets at a time — one in each hand. Each basket must have a package in it. Barnie can carry 8kg in total. Ana and Chloe can each carry 5kg.

All the food must be carried home in a single trip.

Task:
Which items from the shopping list should each beaver carry? Drag the items to their baskets. (Click ‘Save’ when you are done.)

Explanation

Barnie (8kg): 5kg rice **+**
3kg Sugar, 3kg Apple **or** 3kg Willow

Ana and Chloe (5kg): 2kg Corn **or** 2kg Spinach **+**
3kg Sugar, 3kg Apple **or** 3kg Willow

Barnie can carry 8kg so he should carry 5kg of rice and either 3kg of sugar, 3kg of apple or 3kg of willow. Chloe and Ana can carry 5kg so they should carry 2kg of spinach or 2kg of corn and 3kg of sugar, 3kg of apple or 3kg of willow.

We know that:

1. one beaver can only carry two baskets
2. each food item is in its own individual package
3. we have eight different food items

So all four beavers will need to carry two different food items home.

This means the 5kg of rice can only be carried by Barnie. Even though both Ana and Derrick can carry 5kg, this would mean they are unable to carry any other food. To make sure Barnie is carrying the maximum possible, he needs to carry one of the 3kg items (willow, apples or sugar) in his other hand.

This leaves the two beavers that can carry 5kg. They both carry one 2kg and one 3kg item.

Background information

This task requires the satisfying of constraints to determine which beavers can carry which food items. While there are a number of options, there are some important constraints that must be satisfied. The main constraints are that all the food must be carried in one trip, the maximum weight each beaver can carry, and that each beaver can only carry two items.

In a computer system, some processes can only be performed if a given set of constraints are satisfied.

Necklaces

Beavers Anna, Bella and Lena made necklaces to spell out their names.

They used different patterns of just two beads for each letter:

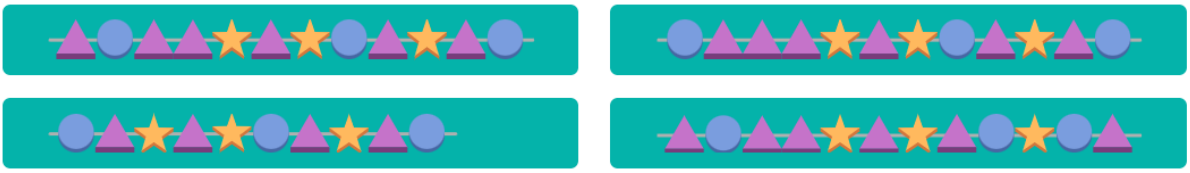
 and  .

To separate the letters in the necklaces they used  beads.

The finished necklaces:

Anna	
Bella	

Question:
Which necklace did Lena make? Click on the correct answer.



Explanation

The correct answer is 

The letter L is in the name BELLA and is the third in a row, so it is made of beads

.

We see that
options  or
 may be correct. Then we find out which beads represent the letter E. From Bella's necklace, we see that E is  .

Options  and

 may still be correct. Now we want to find out which beads are the letter N. From Anna's necklace we see that N is  .

And that is in 

Another way to find a solution can be to determine how beads are threaded for each letter
and then determine what name each of the four answer options represents. So we know from Anna's necklace that the letter A is made of beads  and the letter N is  .S

From Bella's necklace, in turn, B is made of beads, E is and L is. So we will create LENA from  , among which we will insert  .

So we get a sequence that is in answer  .
When we decode the other answers, we find that  is BENA,  is NENA and  is LEAN.

Background information

We commonly encode information, mainly to make communication easier or to write information more economically. Or we can come up with a secret code. Here the encoding of letters is based on the Morse code, where the dot(●) of Morse code is replaced by  and the dash(_) is replaced by  . Therefore, 'A' in ANNA's name is encoded in  (●_).

If we don't know which encoding is used, then we can't create necklaces for names with other letters. We would have to agree on how to encode each letter from the alphabet. However, we can also encode non-text information - pictures, sounds or videos.

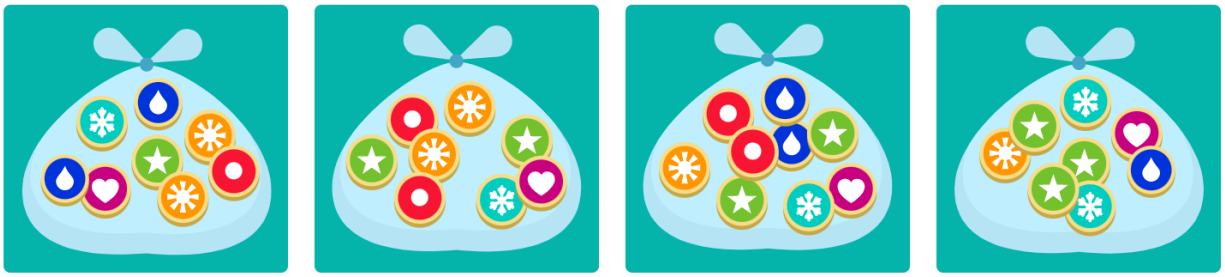
Coin Bag

This is Saoirse's bag of coins.

In Saoirse's country, there are only four types of coins. The image below shows both sides of each type of coin:

Her bag of coins has been shaken and placed next to three other bags.

Question:
Which is Saoirse's bag of coins? Click on the correct answer.



Explanation

Saoirse’s bag has 4 coins of type one (green/yellow), 2 of type 2 two (red/blue) coins, 1 of type three (orange) coin, and 1 of type four (purple) coin, as illustrated in the table below.

				
Saorise's bag	**4**	**2**	**1**	**1**
Bag A	3	3	1	1
Bag B	4	2	1	1
Bag C	**3**	**4**	**1**	**1**
Bag D	4	1	2	1

 is not correct because it has 3 star/sun coins but Saoirse's bag has 4 star/sun coins.

 is not correct because it has 2 snow coins but Saoirse's bag has only 1 snow coin.

 is correct because it has the correct number of each type of coin.

 is not correct because it has 3 star/sun coins but Saoirse's bag has 4 star/sun coins.

Background information

Some of the world's information (stories, conversations, messages, shopping lists) can have different lengths and different orderings (referred to as unstructured information). Computer scientists often have to invent a structure for things when processing information. Sometimes, certain features have to be ignored, and things that look different have to be treated as being equivalent. This is an example of abstraction.

In this task, a bag (or multi-set) is used as an example of very slightly unstructured data (the bag has no particular ordering for its coins and can have multiple coins of the same type). But there are more complicated real-world examples too. Extracting meaning from human language is one very difficult but important task for computers that deal with unstructured data. For example, imagine if people were asked "What did you like best about this movie?" and various people responded with these answers:

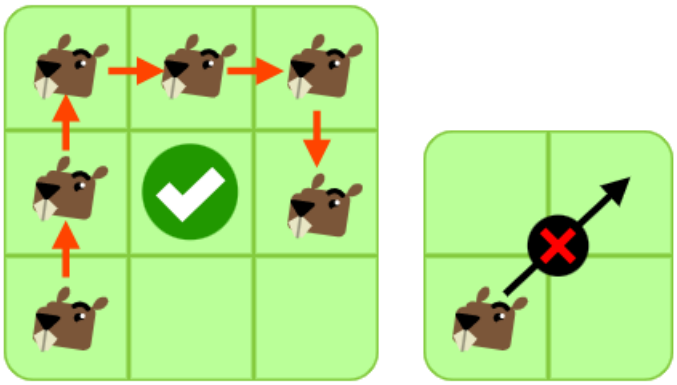
- "I loved the score"
- "The music in this movie"
- "The complete audio experience"
- "I recognised my favourite song"

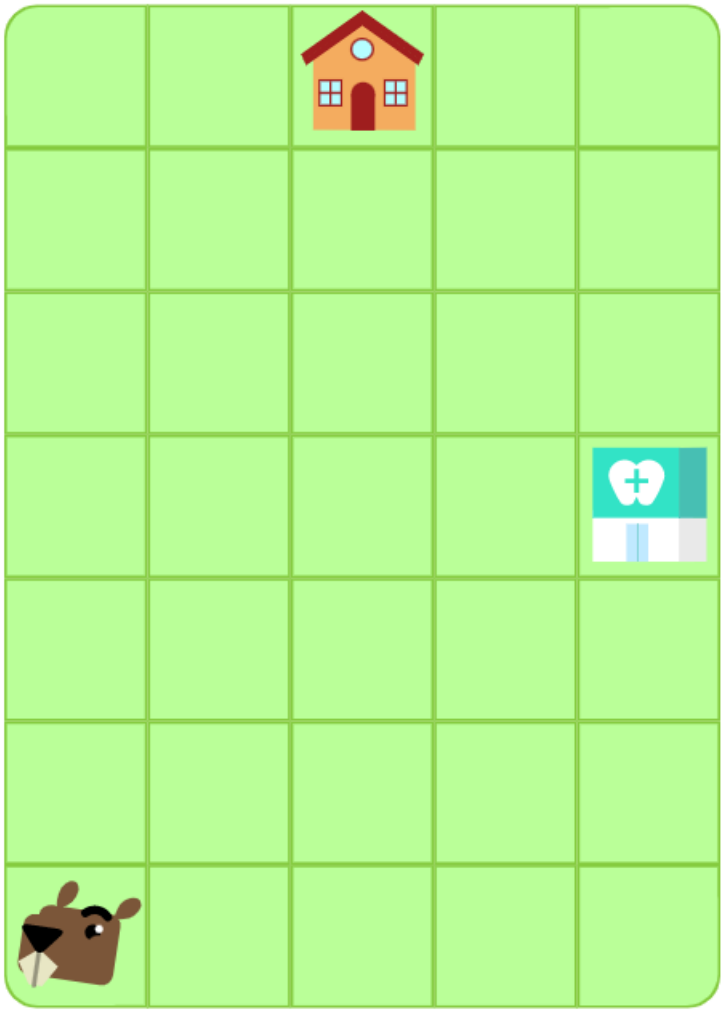
A computer program analysing these peoples' responses would have to recognise that, in this context, all of these statements should be represented as being equivalent even though they all use different words.

Dentist

Beaver is heading home from school, but first, he must make an appointment at the dentist.

Beaver plays a game where he tries to complete the journey by only going straight ahead and turning right.
He can do these two rules as many times as he wants and, in any combination, but he must NOT go diagonally.





Question

Is it possible to reach first the dentist and then home if Beaver follows his rules?

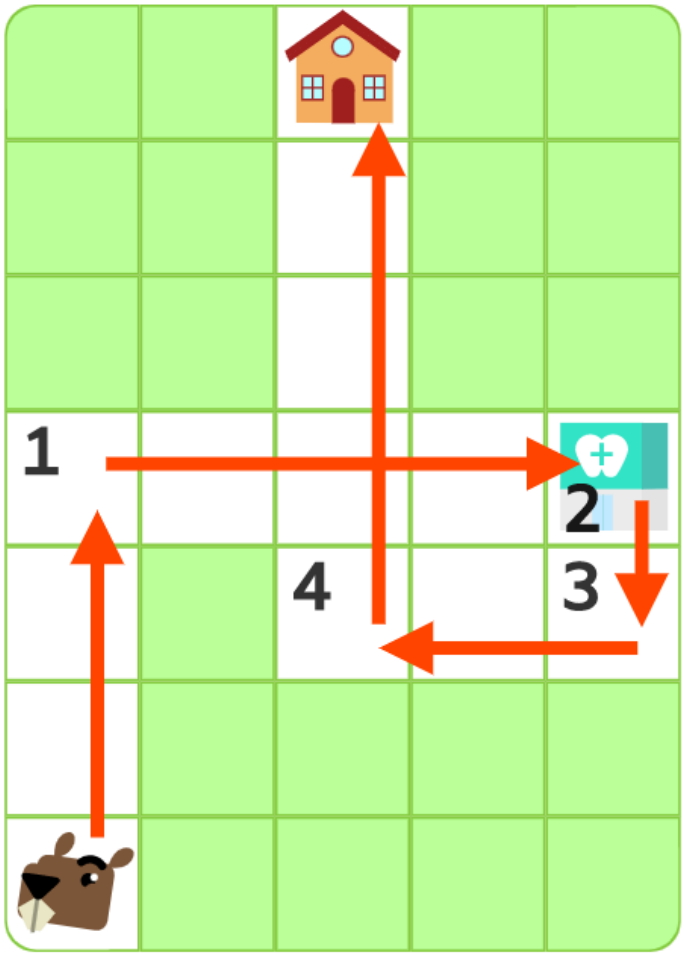
Click on the correct answer.

- It is not possible to reach both places in this way
- It is possible, if he turns right exactly 4 times
- It is possible, if he turns right exactly 2 times
- It is possible, if he turns right exactly 6 times

Explanation

Correct answer: It is possible if he turns right exactly 4 times

The Beaver can get to his home after visiting the dentist if he follows the movement rules: straight and right according to the following picture:



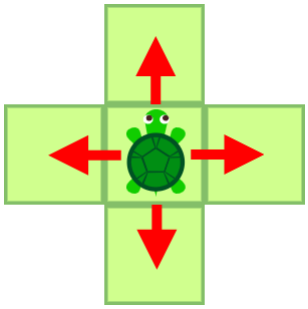
Background information

One of the main tasks of Computer Science is to search for possible solutions, solutions that follow certain conditions.
The question that is often asked is whether there is at least one possible solution.

In this task, it is necessary to write a program for the movement of the Beaver according to the given rules and with certain restrictions.

Turtle Path

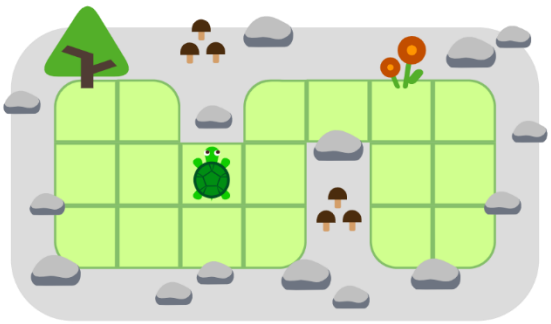
Turtles live in small gardens. Each garden is divided into squares, covered with either grass or stones. The turtles cannot cross stony areas. But they can move from one grass square to the next, as shown in the picture.

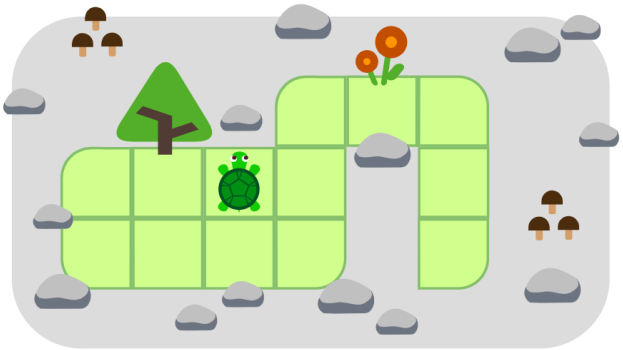


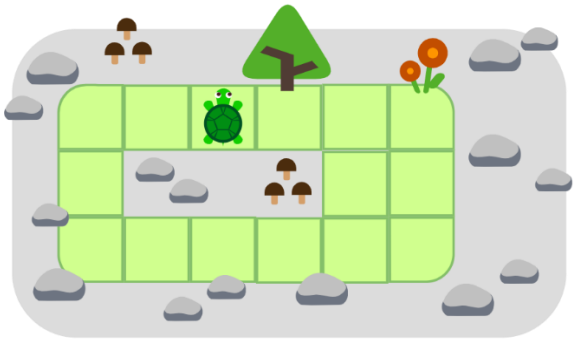
Each turtle needs to take a feeding path in its garden: It needs to move to all grass squares while visiting each of them only once.

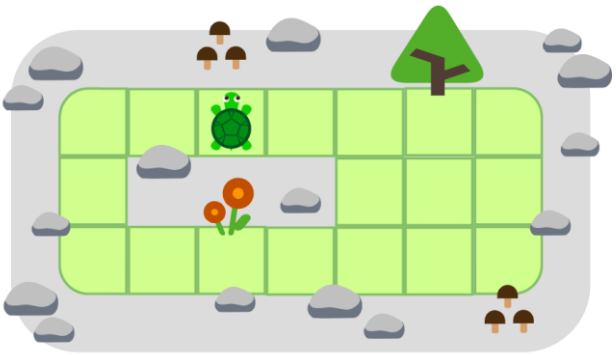
Question:

Unfortunately, one turtle cannot take a feeding path in its garden. Which one? Select its garden below.

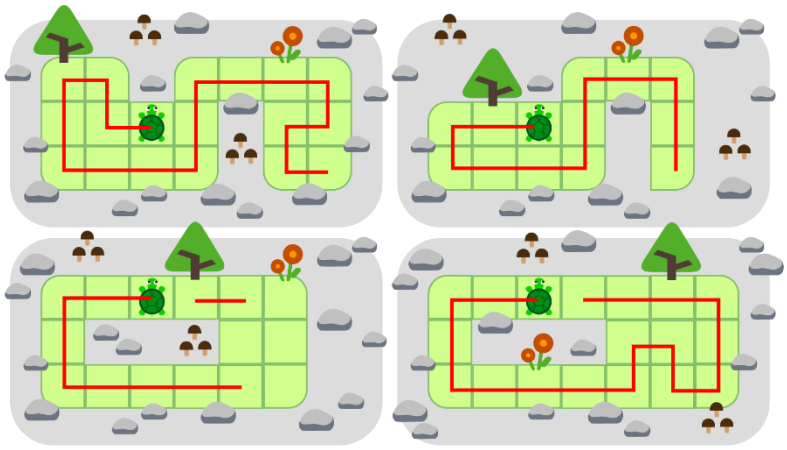






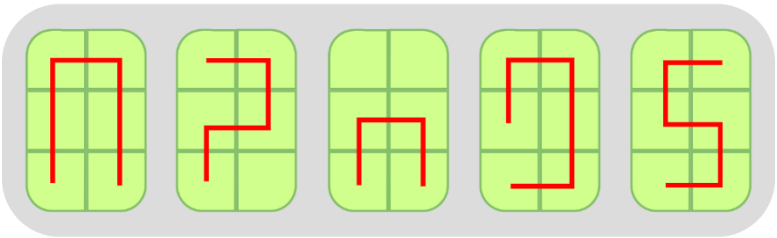


Explanation



For each of the gardens a feeding path is shown above.

There is no feeding path in the bottom left garden, though. Such a path would have to include the parts shown above. Now, the turtle still needs to find a path across the six remaining grass squares, starting at one end of the shown part and ending at the other, while visiting all six squares exactly once. However, there is no such path. Let the turtle start on the lower left of the six squares; it is sufficient to consider this case, due to symmetry. See below all paths the turtle can take on the six squares, visiting as many squares as possible, but each square at most once. None of these paths ends at the upper left square.



Background information

The turtles want to find a path through their garden, visiting each grass square exactly once. This problem is quite well known in informatics – not as the turtle problem, but as "Hamiltonian path problem". A turtle garden (the grass squares, that is) can be regarded this way: Each grass square is a node (depicted as dot), and each possible way from one square to the next is an edge (depicted as line). For such kinds of structures (called graphs by computer scientists, and by mathematicians either), in the 19th century, Sir William Rowan Hamilton wondered whether there would be a cycle path along the edges, visiting each node (other than the starting node) exactly once. This "Hamiltonian cycle problem" is very hard to solve in general; the same holds for the "Hamiltonian path problem", which does not require the path visiting all nodes exactly once to be a cycle. In informatics, there is no known algorithm that, for arbitrary graphs, can efficiently (i.e. within a reasonable time) decide whether there is such a "Hamiltonian path" for the graph or not.