Bebras Australia Computational Thinking Challenge

2023 Solutions Guide
Round 1

Secondary School
Grades 7–12

bebras.edu.au

Bebras Australia Computational Thinking Challenge

Bebras is an international initiative aiming to promote Computational Thinking skills among students.

Started in 2004 by Professor Valentina Dagiene from the University of Vilnius, 'Bebras' is Lithuanian for beaver. This refers to their collaborative nature and strong work ethic.

The International Bebras Committee meets annually to assess potential questions and share resources. Questions are submitted by member countries and undergo a vetting process.

The Bebras international community has now grown to 60 countries with over 3.29 million students participating worldwide!

Bebras Australia began in 2014 and is now administered through CSIRO Digital Careers.

In Australia, the Bebras Challenge takes place in May and August–September each year. As of 2020, two separate challenges are offered for each round.

To find out more and register for the next challenge, visit bebras.edu.au

Engaging young minds for Australia's digital future

518
Australian schools participated in Round 1 2023

31,837
Australian students participated in Round 1 2023

3.29 million
Students participate worldwide

CSIRO Digital Careers supports teachers and encourages students' understanding of digital technologies and the foundational skills they require in an ever-changing workforce. Growing demand for digital skills isn't just limited to the ICT sector. All jobs of the future will require them, from marketing and multimedia through to agriculture, finance and health. Digital Careers prepares students with the knowledge and skills they need to thrive in the workforce of tomorrow.

csiro.au/digital-careers

CSIRO digital careers

What is a Solutions Guide?

Computational Thinking skills underpin the careers of the future. Creating opportunities for students to engage in activities that utilise their critical and creative thinking along with problem solving skills is essential to further learning. The Bebras Challenge is an engaging way for students to learn and practice these skills.

Within this Solutions Guide you will find all of the questions and tasks from Round 1 of the Bebras Australia Computational Thinking Challenge 2023. On each page above the question you will find the age group, level of difficulty, country of origin and key Computational Thinking skills.

After each question you will find the answer, an explanation, the Computational Thinking skills most commonly used, and the Australian Digital Technologies curriculum key concepts featured.

Contents

What is Computational Thinking?

Computational Thinking is a set of skills that underpin learning within the Digital Technologies classroom. These skills allow students to engage with processes, techniques and digital systems to create improved solutions to address specific problems, opportunities or needs. Computational Thinking uses a number of skills, including:

DECOMPOSITION
Breaking down problems into smaller, easier parts.

PATTERN RECOGNITION
Using patterns in information to solve problems.

ABSTRACTION
Finding information that is useful and taking away any information that is unhelpful.

MODELLING AND SIMULATION
Trying out different solutions or tracing the path of information to solve problems.

ALGORITHMS
Creating a set of instructions for solving a problem or completing a task

EVALUATION
Assessing a solution to a problem and using that information again on new problems.

More Computational Thinking resources

Visit digitalcareers.csiro.au/CTIA to download the Computational Thinking in Action worksheets. These can be used as discussion prompts, extension activities or a framework to build a class project.

Each resource was designed to develop teamwork; critical and creative thinking; problem solving; and Computational Thinking skills.

Computational Thinking skills alignment

2023 Round 1 Questions	Grade level	Decomposition	Pattern Recognition	Abstraction	Modelling & Simulation	Algorithms	Evaluation
Years 7+8							
The Gift	Easy	■		■		■	■
Becky Bee	Easy	■			■	■	■
Flower Garden	Easy	■	■		■	■	■
Stone Factory	Easy	■	■	■	■	■	■
Strawberries	Easy	■		■	■	■	■
Classroom Seating	Medium	■	■	■	■	■	■
Colour the Frog!	Medium	■		■		■	■
Ohrid Pearls	Medium	■	■		■	■	■
Tic-Tac-Toe	Medium	■	■	■	■	■	■
Wheat Storage	Medium	■			■	■	■
Favourite Movie	Hard	■		■		■	■
Mysteria	Hard	■	■			■	■
Connection of Islands	Hard	■		■	■	■	■
Ordering	Hard	■	■		■	■	■
Underground Trains Network	Hard	■		■	■	■	■
Years 9+10							
Nuts and Bolts	Easy	■	■	■	■	■	■
Mary's Neighbours	Easy	■		■	■	■	■
Cipher 8	Easy	■	■	■	■	■	■
Rug Weaving	Easy	■	■	■		■	■
In Love	Easy	■		■		■	■
Super Security System	Medium	■	■		■	■	■
Hangar Carousel	Medium	■	■			■	■
Overlapping Villages	Medium	■	■	■	■	■	■
Beaver AI	Medium	■	■	■		■	
The Printer	Medium	■		■	■	■	■
Four Tiles	Hard	■	■	■			■
Favourite Gem	Hard	■				■	■
Collecting Stones	Hard	■		■			■
Maze	Hard			■	■		■
Tree Farming	Hard	■	■	■	■	■	■

Computational Thinking skills alignment

2023 Round 1 Questions	Grade level	Decomposition	Pattern Recognition	Abstraction	Modelling & Simulation	Algorithms	Evaluation
Years 11+12							
Lists	Easy						
Listen and Walk	Easy						
Colourful Candles	Easy						
Movie Night	Easy						
Words	Easy						
Beaver Dam	Medium						
Jumping Game	Medium						
Packing	Medium						
Floor Pattern	Medium						
Treasure Box	Medium						
Beaver Games	Hard						
The Beaver Database	Hard						
Ordering	Hard						
Virus	Hard						
Seashells and Pebbles	Hard						

Australian Digital Technologies curriculum key concepts

Abstraction

Hiding details of an idea, problem or solution that are not relevant, to focus on a manageable number of aspects.

Data Collection

Numerical, categorical, or structured values collected or calculated to create information, e.g. the Census.

Data Representation

How data is represented and structured symbolically for storage and communication, by people and in digital systems.

Data Interpretation

The process of extracting meaning from data. Methods include modelling, statistical analysis, and visualisation.

Specification

Defining a problem precisely and clearly, identifying the requirements, and breaking it down into manageable pieces.

Algorithms

The precise sequence of steps and decisions needed to solve a problem. They often involve iterative (repeated) processes.

Implementation

The automation of an algorithm, typically by writing a computer program (coding) or using appropriate software.

Digital Systems

A system that processes data in binary, made up of hardware, controlled by software, and connected to form networks.

Interactions

Human-Human Interactions: How users use digital systems to communicate and collaborate.
Human-Computer Interactions: How users experience and interface with digital systems.

Impact

Analysing and predicting how existing and created systems meet needs, affect people, and change society and the world.

For more information on the Digital Technologies curriculum, please visit the Australian Curriculum, Assessment and Reporting Authority (ACARA) website:
australiancurriculum.edu.au/f-10-curriculum/technologies/digital-technologies

Digital Technologies key concepts alignment

2023 Round 1 Questions	Grade level	Abstraction	Data Collection	Data Representation	Data Interpretation	Specification	Algorithms	Implementation	Digital Systems	Interactions	Impacts
Years 7+8											
The Gift	Easy	■				■	■		■		
Becky Bee	Easy		■	■	■	■	■	■			
Flower Garden	Easy				■	■	■	■			
Stone Factory	Easy	■				■	■	■	■		
Strawberries	Easy	■		■	■	■	■	■			
Classroom Seating	Medium	■				■	■				
Colour the Frog!	Medium	■				■	■	■	■		
Ohrid Pearls	Medium					■	■				
Tic-Tac-Toe	Medium	■			■	■	■				
Wheat Storage	Medium		■			■	■		■		
Favourite Movie	Hard	■	■		■	■	■	■			
Mysteria	Hard			■	■		■				
Connection of Islands	Hard	■	■		■	■	■			■	
Ordering	Hard				■	■	■				
Underground Trains Network	Hard	■	■		■	■	■			■	
Years 9+10											
Nuts and Bolts	Easy	■		■	■	■	■				
Mary's Neighbours	Easy	■		■	■	■	■			■	
Cipher 8	Easy	■		■	■	■	■		■		
Rug Weaving	Easy	■		■	■	■	■				
In Love	Easy	■			■	■	■				
Super Security System	Medium			■	■	■	■	■		■	
Hangar Carousel	Medium			■	■	■	■	■			
Overlapping Villages	Medium	■		■	■	■	■	■	■		
Beaver AI	Medium	■			■		■				■
The Printer	Medium	■	■		■	■	■		■		
Four Tiles	Hard	■		■	■	■			■		
Favourite Gem	Hard					■					
Collecting Stones	Hard	■		■	■	■	■		■	■	
Maze	Hard					■	■				
Tree Farming	Hard	■		■	■	■	■		■		

Digital Technologies key concepts alignment

2023 Round 1 Questions	Grade level	Abstraction	Data Collection	Data Representation	Data Interpretation	Specification	Algorithms	Implementation	Digital Systems	Interactions	Impacts
Years 11+12											
Lists	Easy	■		■	■	■	■	■			
Listen and Walk	Easy	■		■	■	■	■				■
Colourful Candles	Easy			■	■		■				
Movie Night	Easy		■			■	■				
Words	Easy			■	■	■	■		■		
Beaver Dam	Medium	■		■	■	■	■				
Jumping Game	Medium	■				■	■	■			
Packing	Medium					■	■	■			
Floor Pattern	Medium	■		■	■	■	■	■		■	
Treasure Box	Medium	■			■		■				
Beaver Games	Hard	■		■	■	■					
The Beaver Database	Hard	■		■	■				■	■	
Ordering	Hard	■		■	■	■	■	■	■		
Virus	Hard	■		■	■	■	■	■		■	■
Seashells and Pebbles	Hard	■		■	■	■					

Bebras Challenge 2023 Round 1

Years 7+8

The gift

Gaby's dad gives her four boxes containing four different gifts; a bracelet, a notebook, a ring, and perfume. There is only one gift per box. By applying logic, Gaby can keep the one she likes the most. Each box has a label that does not lie.

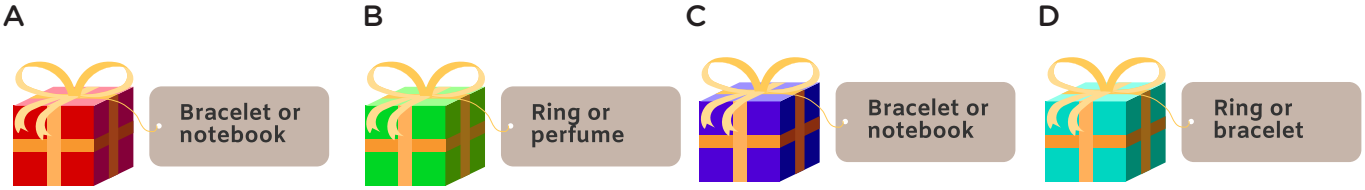
Question

Gaby wants to keep the ring. Which box has the ring in it?

EXPLANATION

Answer

Explanation



We know for a fact that labels don't lie, so the bracelet and the notebook must be in either box A or B. Either the notebook is in box A and the bracelet is in box B or vice versa. That leaves Gaby with box C and box D. From the previous statement, we know that the bracelet is either in box A or box B, meaning it cannot be in box D. Therefore the ring must be in box D, which is what Gaby should choose.

For completeness, we now also know that the perfume is in box C.

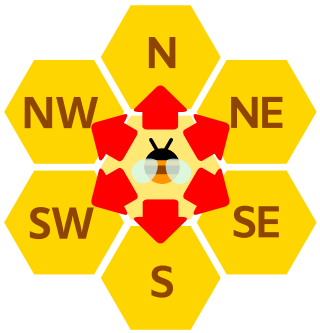
BACKGROUND INFORMATION

This task involves some logical thinking and computational thinking skills. The data for the problem is given by the gift tags. By *decomposing* and *abstracting* the problem we can take a look at the single labels and realise some boxes (in this case, A and B) are mutually exclusive. We could then start drawing some conclusions about part of the problem. After having assigned some of the gifts, we can find the solution by discarding irrelevant information (i.e., we already assigned the bracelet to either gift A or gift B, so it's irrelevant for gift D, immediately giving us the solution).

Logic plays a key role in computer science (databases, programming languages, artificial intelligence, hardware and software design and verification, etc.), and is undoubtedly one of the foundations of further computer science concepts, languages, and techniques. Logic is especially important to define the *control flow* of a program - that is, execute the correct codes at the right time depending on the data it is processing. To do so, programming makes use of *logical operators* such as "AND", "OR", and "NOT".

Becky bee

Becky Bee wants to collect as much honey as possible from her beehive by visiting different sections. Becky can only move in 6 directions:

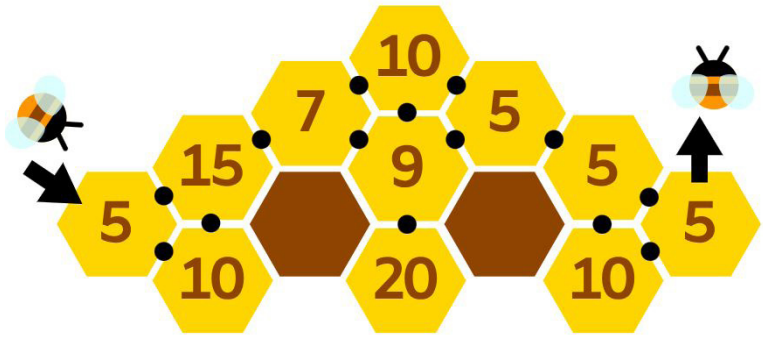


The hive below shows how much honey she can collect in each section as she flies through it. She follows these rules:

- Because of noise regulations, she can move through each section at most once.
- Some sections are closed off completely.
- She only has time to visit 9 sections on this trip.

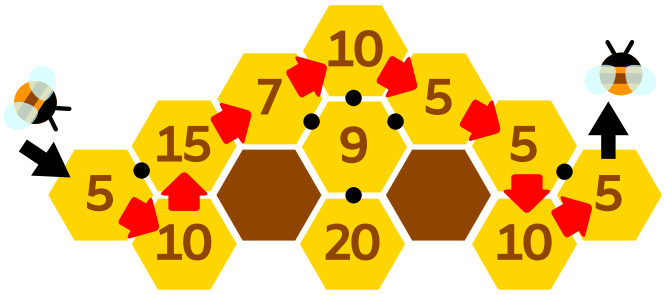
Question

Can you give Becky instructions on how to fly through the hive, to collect as much honey as is possible?



EXPLANATION

Answer



Continued on next page

This question comes from
Romania

Becky bee – continued

Explanation

The flight of Becky Bee is shown in the diagram above. Because you are not allowed to fly through a section more than once, you cannot fly through the section with '20'. That basically only leaves 2 possible paths that visit 9 of the sections (the difference is going through 9 or 10 in the center). Going through the center path labelled 10 allows Becky to gather the most honey.

BACKGROUND INFORMATION

One approach of solving this task would be to try out all different paths. Each path is a *permutation* of the 8 instructions.

If we didn't have a set of arrows and could pick our own directions, there would be 40320 possible paths in this problem! Since NE and SE both appear 3 times, there are a lot less, but this still leaves 1120 possible different paths.

A computer would be able to try out all these paths (or even all 40320 paths). It could check whether they start and end at the right section, visit sections more than once, visit blocked sections, or ever leave the hive.

Trying out every path is called an *exhaustive search*. Computers are very good at this.

Another way to try to solve this puzzle, is to start at the beginning and try the different arrows you still have left in each section. If you run into a section from where you cannot make a move, you go back to a previous section and try another arrow there. This type of exhaustive search is called backtracking.

Flower garden

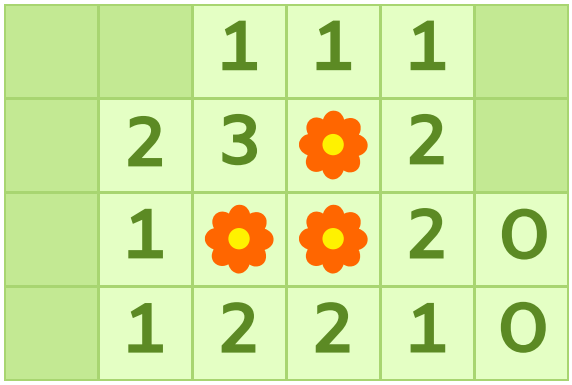
The Flower Garden is a video game where players try to find flowers without stepping on them.

Rules:

- Players can press the A button on a square where they do not think there is a hidden flower.
  This will either reveal on that same square
  - a number that shows how many neighbouring squares have flowers in them 1.
  - a hidden flower that they have now "stepped on", and have lost the game.
- Players can press the B button on a square when they know there is a flower there.
  This makes the flower "grow".

If the player "grows" all the hidden flowers before stepping on one, they win the game.

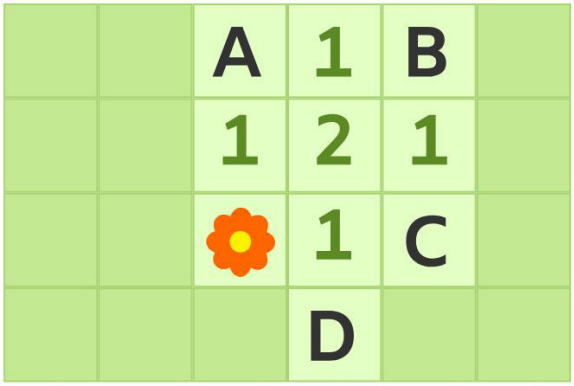
Ariana saved a game in progress where she had "grown" three flowers, but there were still more flowers in the remaining blank squares.



Ariana decides to start a new game below. She thinks that there is a flower in one of the squares marked A, B, C, or D but has not decided on which to click with button B yet.

Question

In which square marked A, B, C, or D is there another hidden flower?



Continued on next page

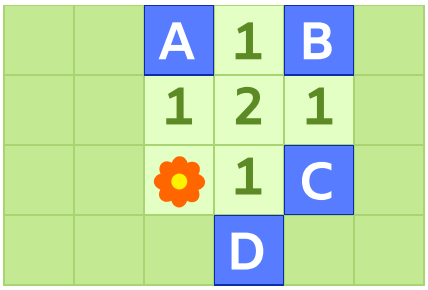
Flower garden – continued

EXPLANATION

Answer

B

Explanation



In the square below square A, there is a "1". We can see there is a flower below that. So the "1" indicates that flower and thus, square A cannot contain another flower, as the "1" flower is accounted for.

In the square to the left of square C, and above square D we have a "1". To the left of that "1" is the previously identified flower. So this "1" indicates that flower and, thus, both square C and square D cannot contain another flower.

Square B has two neighbouring squares with a "1". These "1"s could be referring to square B, but the "2" on the square to the bottom left of square B, proves that a flower needs to be in square B. This "2" is only touching one flower, and we have already identified that another flower cannot be in squares A or C. The only option left is square B.

BACKGROUND INFORMATION

Tasks like this sometimes require starting with an assumption. If you find an error, for example a "1" that refers to an item (and because of this you cannot have an additional item), you "go back" and amend the false assumption.

This relates to the very popular algorithmic technique called *backtracking*. The disadvantage of backtracking is that sometimes you need to "go back" a lot, which wastes time and resources. As well as this, you need to define a clear "ending", in the case of there being no solution.

This question comes from the Netherlands

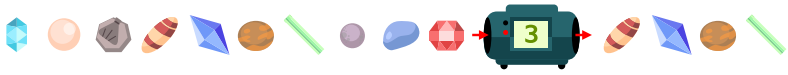
Years 3+4
Years 5+6
Years 7+8 Easy
Years 9+10
Years 11+12

Stone factory

A stone-sorting factory has two kinds of machines. Each machine will remove some of the stones and return the others as its output.

The pink machine is called an “odd machine”. It removes every second stone just like the picture below:

The grey machine with a number “n” on it is called a “trim machine”. It removes the first and last “n” stones, and outputs whichever stones are left. The picture below shows how a trim machine with the number “3” on it works:



These two types of machines can be combined, and the stones that come out of the first machine will be the input of the second machine.

Question

If the stones are arranged and put into the machine like the picture below, what will be the output?

EXPLANATION

Answer

Continued on next page

This question comes from the Netherlands

Years 3+4
Years 5+6
Years 7+8 Easy
Years 9+10
Years 11+12

Stone factory – continued

Explanation

The stones are originally arranged as follows:

They go through an odd machine first. Every second stone is removed, starting from the first stone, and the output is as follows:

Then, this output of stones goes through the trim machine with “2” on it, The machine removes the first and last 2 stones, and the output is:

The answer can be found efficiently without working out the entire sequence. We can see that the first stone for each answer is different - by working out this stone, we will know the solution. Given that the pink machine will only return the 1st, 3rd, 5th, 7th... stones, and the trim machine will remove the first two, we can quickly discover that the first output stone will be the 5th stone in the initial arrangement.

This quickly gives us that the output starts with  , and hence the correct answer..

BACKGROUND INFORMATION

Functions can manipulate and transform data in a computer program. They are among the most important tools for a programmer. Just like with mathematics, we use parentheses (or brackets) to determine the order of operations. The way the functions are used is like a recipe. Applying this in a program is like using the recipe to create something new - in this case a new string of data determined by the input string.

There are two major views on computer programs. On one hand you can look at a computer program as a set of instructions that are executed in the proper order. The other view is to regard a program as a machine that is able to process an input, and to produce output.

In this example image, we see a *nested function*, where there is a function within a function which means that the result of the first machine is passed to a second one.

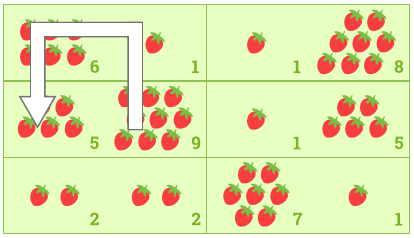
This question comes from Taiwan

Strawberries

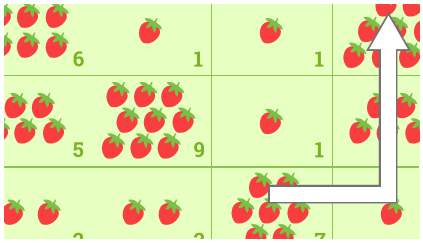
Beavers love to play a game called “Strawberry Hunt”! These are the game rules:

1. A different number of strawberries are placed on each area of a field.
2. A player starts from any area of the field and eats the strawberries in that area.
3. A player then can take only 3 total steps. Each step can be:
   - Move up one area
   - Move down one area
   - Move right one area
   - Move left one area
4. After each step, eat all the strawberries found in that area of the field.

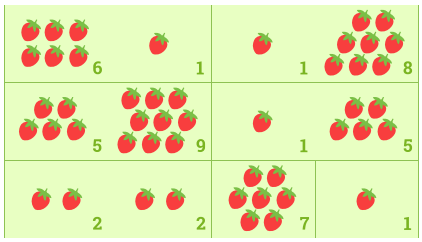
Example 1: A beaver will eat 9+1+6+5 = 21 strawberries in this path:



Example 2: A beaver will eat 7+1+5+8 = 21 strawberries as well.



Alessa the beaver is playing on this grid:



Question

How many strawberries can Alessa eat at most?

EXPLANATION

Answer

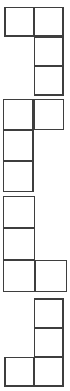
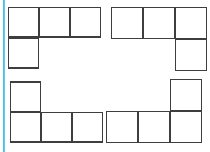
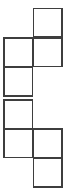
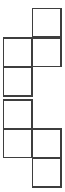
23

Explanation

There are 51 possible ways to finish the game (see table below), but we don’t have to check all 51 total sums of strawberries.

We know that each player can reach 4 areas in total. These paths can only make the following shapes:

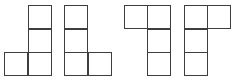
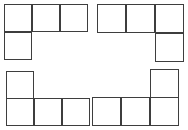
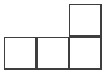
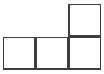
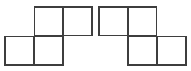
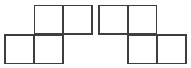
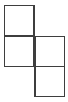
We can first check these shapes in any of the areas with 9 or 8 strawberries

Shape	Square	Line	L shape	Horizontal L shape	Z shape	Vertical Z shape
Form						
Possibilities	**6**	**3**	**12**	**16**	**8**	**6**

Continued on next page

Strawberries – continued

(including all shapes with both of these areas). Since checking all field combinations with 9 or 8 strawberries will also cover those fields with 5, 6 or 7 strawberries, all grids with larger numbers of strawberries will be covered. Then, we can find the largest total number of strawberries:

	Number of strawberries Little Beaver can eat on a path		
	Including the area '9'	**Including the area '8'**	**Largest of the two**
Square 	9+5+6+1=21 	8+5+1+1=15 	21
Line 	5+9+1+5=20 	6+1+1+8=16 	20
L shape 	7+2+9+1=19 	8+5+1+7=21 	21
Horizontal L shape 	9+1+5+8=23 	9+1+5+8=23 	23
Z shape 	5+9+2+7=23 	8+1+1+9=19 	23
Horizontal Z shape 	6+5+9+2=22 	8+5+1+7=21 	22

BACKGROUND INFORMATION

In computer science, many problems aim to find the maximum or minimum value. In these problems, there are often many options for how to proceed. Those that find these values are called the *optimal solution*. Some examples are; doing the most work, spending the least amount of money, or – in this Bebras task – eating the most strawberries.

Usually, there are many ways to find the optimal solution. One of them is by listing all the possible options, calculating their values one by one and choosing the options with the best values. We call this method the *brute-force search* or *exhaustive search*. However, this method may take tremendous computing time if there are many options. Therefore, it is helpful to analyse the problem. We can look for constraints which reduce the number of options, and help us find the optimal solution faster.

For many problems, computer scientists know methods to find optimal solutions efficiently, but for other problems it is known that they cannot be efficiently solved.

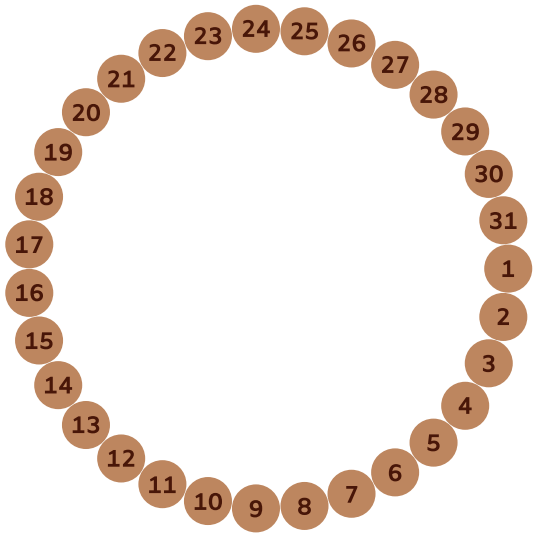
This question comes from India

Classroom seating

There are 31 chairs in a class at Beavaria Middle School. They are placed in a circle and numbered 1 to 31, as shown in the picture here.

All the students leave the classroom and then play this game:

- Enter the classroom one at a time.
- Sit on a chair that has the number of the day you were born.
- If the chair with this number is already taken, continue walking along the circle in a clockwise direction until you find a free chair, and sit down.



For example, suppose Geeta and Seeta are twins born on April 20, Arun is born on January 21 and Zubin is born on September 22.

- If they arrive in the order Geeta, Seeta, Zubin, Arun, then Geeta sits on chair 20, Seeta sits on chair 21, Zubin sits on chair 22, and Arun sits on chair 23.
- If they arrive in the order Seeta, Arun, Zubin, Geeta, then Seeta sits on chair 20, Arun sits on chair 21, Zubin sits on chairs 22, and Geeta sits on chair 23.

Question

Six children have entered the class and are seated as shown below:

Name	Birthday	Seated at
Abha	May 11	Chair 13
Byram	February 12	Chair 12
Chetan	September 14	Chair 14
David	August 11	Chair 11
Eesha	April 13	Chair 15
Fatima	July 12	Chair 16

Which of the following statements **cannot** be true?

Chetan was the first to sit down

Fatima the last to sit down

Eesha sat down before Abha

Byram sat down before David

Continued on next page

This question comes from India

Years 3+4
Years 5+6
Years 7+8 Medium
Years 9+10
Years 11+12

Classroom seating – continued

EXPLANATION

Answer

The correct answer is "Eesha sat down before Abha".

Explanation

Cheetan sits in the same chair as his birthday, so he can be the first to sit down. Option "Chetan was the first to sit down" can be a correct statement.

Fatima must have arrived after those seated in chairs 12 to 15 to land on chair 16. David sits in the chair 11, so he must have arrived before Abha, who has the same birthday as him. So Option "Fatima was the last to sit down" is a correct statement.

If Eesha arrived before Abha, chair 13 would have been available and she would not have needed to shift to chair 15. Option "Eesha sat down before Abha" cannot be a correct statement.

Byram and David are both seated in the chairs with the same number as their birthdays, so their order can be interchanged. Option "Byram sat down before David" can be a correct statement.

BACKGROUND INFORMATION

Hashing is a technique to assign a large set of possible input values to a fixed set of output values. Here the input set is the set of all birthdays and the output set is the set of chairs, 1 to 31. A hash function computes the output from the input. Since the input set is much larger than the output set, two inputs may get mapped to the same output.

We need to provide a mechanism to deal with such collisions. The task describes a solution called *open addressing*, by which collisions are mapped to higher values in the same output set. Hashing is used to implement data structures such as dictionaries.

This question comes from Italy

Colour the frog!

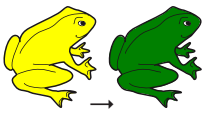
Lulu is learning to write programs that change the colour of images. She wants to turn her frog green.

She has written four different programs for changing the colour of the frog. Each program consists of a sequence of 'if-then-else' or 'if-then' statements. Each of the steps are carried out in order, one after the other. Each step will change a frog's colour once.

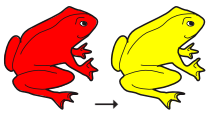
For example, for the following instruction:

IF the frog is **green** **THEN** colour it **yellow** **ELSE** colour it **red**

If the frog was green, this step would turn it yellow.



If the frog was yellow (or any other colour that isn't green), this step would turn it red.



She wants to be sure that the frog is green when the program is finished, regardless of the frog's colour beforehand.

Question

Three out of four of Lulu's programs will ensure the frog is ultimately coloured green.

Which program, when finished running, does not ensure the frog will be green?

IF the frog is **red** **THEN** colour it **yellow** **ELSE** colour it **green**
IF the frog is **yellow** **THEN** colour it **red**
IF the frog is **NOT** **yellow** **THEN** colour it **green**

IF the frog is **red** **THEN** colour it **yellow**
IF the frog is **NOT** **red** **THEN** colour it **green**
IF the frog is **yellow** **THEN** colour it **red** **ELSE** colour it **green**

IF the frog is **yellow** **THEN** colour it **green**
IF the frog is **NOT** **yellow** **THEN** colour it **red**
IF the frog is **red** **THEN** colour it **green** **ELSE** colour it **yellow**

IF the frog is **yellow** **THEN** colour it **green** **ELSE** colour it **red**
IF the frog is **red** **THEN** colour it **green** **ELSE** colour it **yellow**

EXPLANATION

Answer

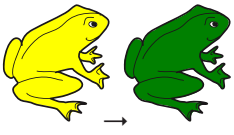
IF the frog is **yellow** **THEN** colour it **green** **ELSE** colour it **red**
IF the frog is **red** **THEN** colour it **green** **ELSE** colour it **yellow**

Continued on next page

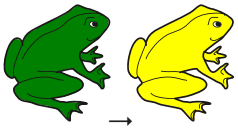
Colour the frog! – continued

Explanation:

If at the beginning the frog is yellow, following the first if-then-else instruction colours it green



after which the condition "the frog is red" is false and it will be coloured yellow again.



If you run this program:

After the first **if-then-else** instruction, the frog is either yellow or green. After the second instruction it is either red or green. Therefore the last condition "the frog is not yellow" is true and it will therefore be coloured green.

If you run this program:

When the second instruction is done, the condition "the frog is not red" will be true, so the frog will be coloured green. After this, the third instruction (if-then-else) colours it green again. We don't have to worry about a frog being yellow going into the third instruction because second instruction all yellow frogs have already been turned green.

If you run this program:

After the second instruction, the condition "the frog is not yellow" will be true, so the frog will be coloured red. Following this, the third instruction (if-then-else) colours it green.

BACKGROUND INFORMATION

Imperative programming is a way of providing instructions to a computer on how to deal with situations without demanding a particular outcome. This can be achieved by writing a sequence of *conditional statements*. These statements change a program's state, and must be actioned in the given order.

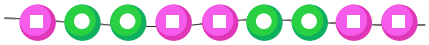
A conditional statement has the form if **C then A else B** or the form **if C then A**.

It should be noted that the two scripts if
C then A else B and
if C then A;
if not C then B
do not always have the same meaning.

In the second case (where the condition is evaluated twice), the actioning of instruction A could change the state of the system in such a way as to make condition C false. This would cause – immediately afterwards – the actioning of instruction B too. However, in the first case, one and only one of the two instructions is actioned.

Ohrid pearls

During their vacation in Ohrid, Monika and Veronika bought Ohrid pearls and made necklaces from them. Their necklaces are shown below.

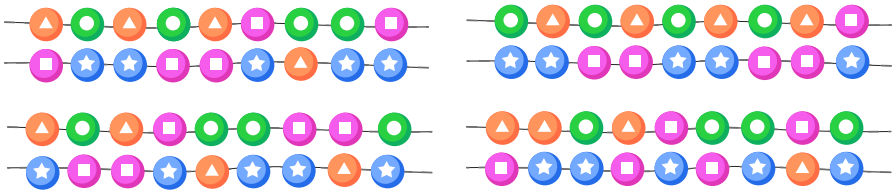
	Left	Right
Monika		
Veronika		

They decided to change the pearls in their necklaces using the following steps.

1. Monika and Veronika each take a pearl from the right of their own necklace.
2. If the pearl is pink (square) or orange (triangle) they add it to the left of the other's necklace. If it is one of the other colours, they add it to the left of their own necklace. Monika goes first and then Veronika.
3. If each has given three pearls to the other, they stop, otherwise they go to step 1.

Question

What do Monika's and Veronika's necklaces look like after the steps have been completed?



EXPLANATION

Answer

Explanation

Option  is not correct, because according to the rules, Monika and Veronika will only excange three pearls each. But in this example they have exchanged four.

Option  is not correct, because the steps have not finished yet.

Option  is not correct because, although the numbers of different coloured pearls is correct, the pearls are in the wrong order.

To see why option  is the correct answer, we can follow the steps one by one, as follows:

We start with the initial state of the two necklaces:

	Left	Right
Monika		
Veronika		

Continued on next page

Ohrid pearls – continued

We will follow the steps to remove pearls from the right side of each necklace, one at time, alternating between each necklace, starting with Monika. In the explanatory text below, we use the term "iteration" to refer to one pass through of the steps 1 through 3. A picture showing the result of each iteration can also be seen below. In this picture, the first pair of coloured rows represents the starting state of the necklaces, the subsequent pairs of coloured rows represent the necklaces after each iteration.

Iteration 1.

Since both of the pearls on the right sides are pink or orange, they are exchanged and added to the other's necklace on the left side. They have each given one pearl so far.

Iteration 2.

Since Monika has a pink pearl at the right side of her necklace, she adds it to the left side of Veronika's necklace. Veronika has a blue pearl (that will not be exchanged) so it's added on the left side of her own necklace. Monika has given two and Veronika has given one.

Iteration 3.

Monika's next pearl is green, so she adds it to the left side of her own necklace. Veronika's next pearl is orange, so she adds it to the left side of Monika's necklace. Each has given two.

Iteration 4.

Monika's next pearl is green, so she adds it to the left side of her own necklace. Veronika's next pearl is blue, so she adds it to the left side of her own necklace. No change: each has given two.

Iteration 5.

Both next pearls are pink and orange, so they are exchanged and added to the other's necklace on the left side. Each has given three.

Since they have each given three pearls to the other, they stop, and this gives the final answer.

Monika's necklace initial state Veronika's necklace initial state	
Monika Veronika	
Monika Veronika	
Monika Veronika	
Monika Veronika	
Monika Veronika	

BACKGROUND INFORMATION

A *queue* is a list in which elements are added only at one end and taken away only from the other end. It can be referred to as a *FIFO* (first in – first out) structure for holding data. It is a useful data structure because it automatically keeps track of the order that items were added to the queue.

The necklaces of Monika and Veronika can be regarded as queues. Although we don't explain how the pearls were put on the necklace in the first place, the rules explain that there is a specific end on which to add new pearls, and the other end is only for removing pearls.

Tic-tac-toe

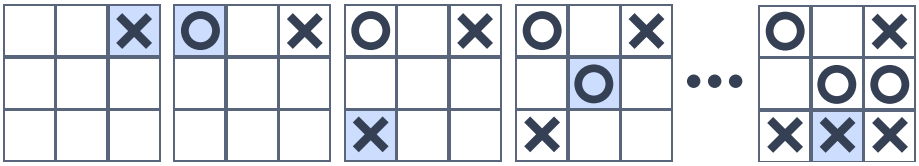
Tic-tac-toe is a paper-and-pencil game for two players.

Rules:

- Players choose who uses X as their mark and who uses *O*.
- One player starts, then both players take turns marking the spaces in a three-by-three grid with an *X* or an *O*.
- The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner.
- If no one succeeds and all nine boxes are filled, the game ends in a draw.

Example:

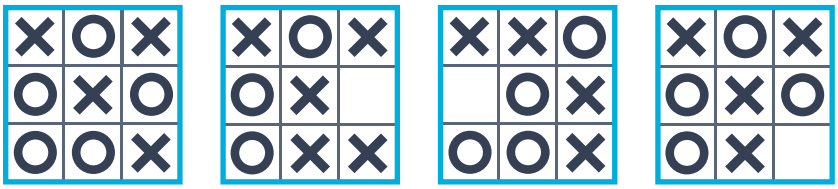
The following images show the first two moves each player made, and the last move they made in a game. (The moves made on each turn are highlighted.)



The image on the right, is what we call the 'result sheet' of a completed game. Not all sheets, that are filled randomly with *X* or *O*, are valid result sheets according to the rules above.

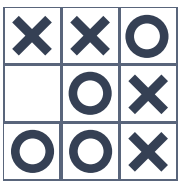
Question

Which of the following images is the only valid result sheet, of a completed game, according to the rules above?



EXPLANATION

Answer



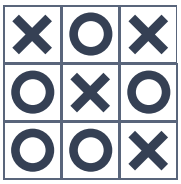
Continued on next page

Tic-tac-toe – continued

Explanation

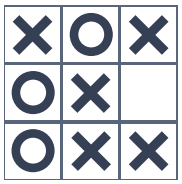
This is a valid game because it was won by the player using O, and then stopped.

Answer



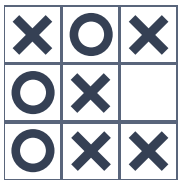
is not correct.

Answer



is not correct because there are 5 X-marks but only 3 O-marks. That is not possible. The difference between the numbers can only be 0 or 1.

Answer



is not correct because a winner had not been determined, and the fields were not all filled wcompletely.

BACKGROUND INFORMATION

Rules are very important in computer systems that process data. For example, there are rules that define the format of image files, credit card numbers or even telephone numbers. When you try to open a file with an image editor, the software first checks whether the content of the file is valid data, according to the rules of an image format.

Rules-based systems are used in computer science to not just store information, but also help interpret the results in a meaningful way. This is done by mimicking human intelligence, for example by using *IF/ THEN* statements.

This question comes from Russia

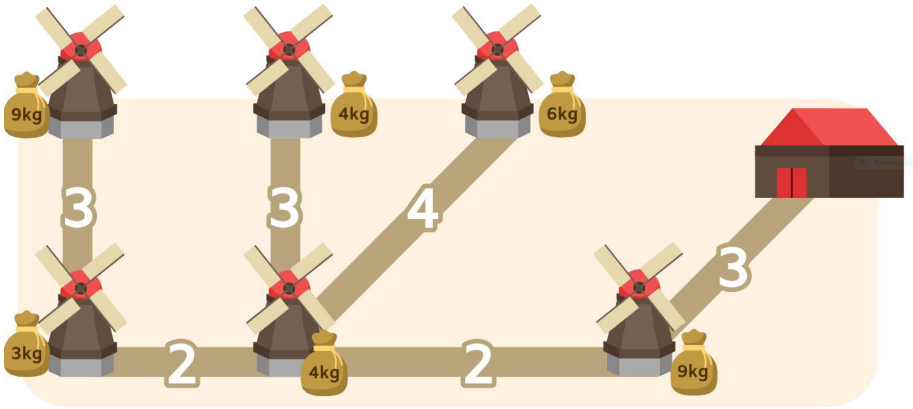
Years 3+4
Years 5+6
Years 7+8 Medium
Years 9+10
Years 11+12

Wheat storage

The picture shows several windmills and the roads connecting them to the storage shed. Every evening, the windmills produce sacks of flour and place them out the front.

Willie the beaver is supposed to pick up all the sacks and bring them to the storage shed before sunset. Willie can carry several sacks at the same time, but the total weight cannot be heavier than 15kg.

The time it takes Willie to travel from one windmill to another and to the storage shed is shown in the picture.



Question

Starting from the storage shed, Willie the beaver wants to bring in all the sacks to the storage shed as fast as possible. How many minutes does it take him to carry out this task?

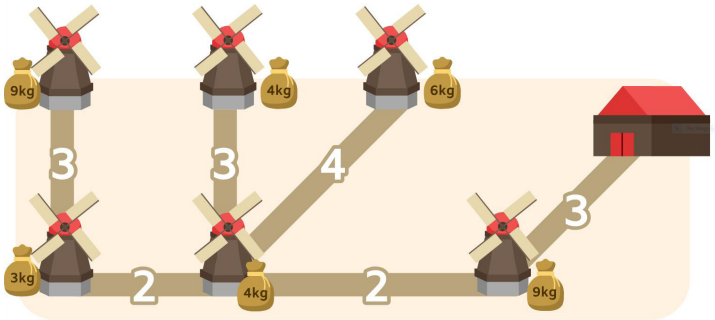
50 minutes | 31 minutes | 44 minutes | 54 minutes

EXPLANATION

Answer

50 minutes

Explanation

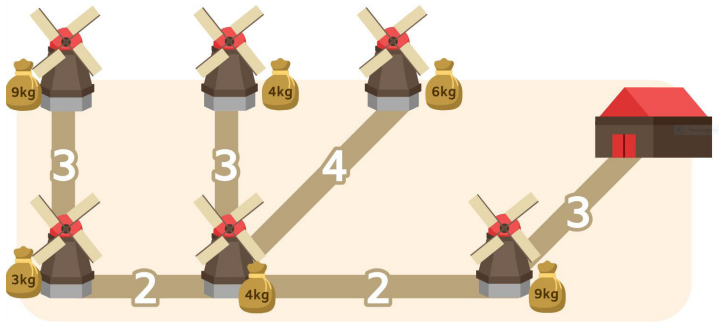


The total kilos of flour to be picked up is 35 kg. Since Willie the beaver can carry a maximum of 15 kg per trip, then he would have to take at least 3 trips to bring in all the sacks. To minimise his time, he would have to pick up sacks in neighboring mills.

For example, in picking up the 9kg sack from the upper left mill, Willie can also pick up the 3 kg sack in its neighboring mill (Total weight: 12kg). All other sacks weigh more than 3 kilos, so he cannot pick up any other sack. The first trip would take 20 min (3+2+2+3+3+2+2+3).

Continued on next page

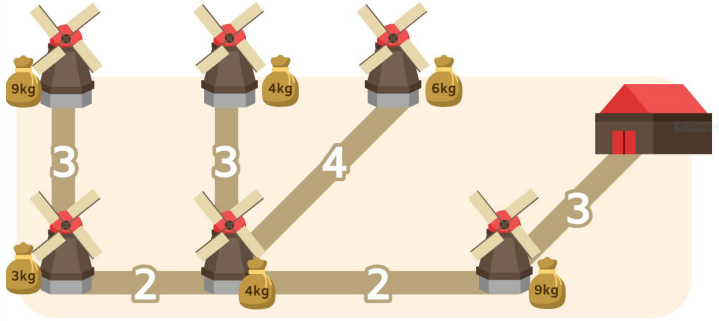
Wheat storage – continued



To pick up the remaining sacks in the four remaining mills, the second trip would have to bring in the three farthest sacks, the 6kg in the upper right mill and two 4 kg in the middle mills (Total weight: 14 kg). The second trip would take 24 min (3+2+4+4+3+3+2+3).

The third trip will pick up the remaining 9 kg from the mill closest to the storage and would take 6 min (3+3). So the total trip would take 50 min.

The order of the first, second, and third trips above can be interchanged; what is important is the grouping of the sacks to be picked up.



Option “31 minutes” cannot be the answer because to pick up the 9 kg sack from the upper left mill would already take 20 min (see first trip above). The remaining 10 min is only enough to go to the second mill and back, so Willie will not be able to pick up the two sacks in the upper mills.

Option “44 minutes” cannot be the answer because 44 min is just enough time for the first and second trip, so the 9 kg in the nearest mill would not have been picked up.

Option “54 minutes” cannot be the answer since this is longer time than Option “50 minutes”.

BACKGROUND INFORMATION

In computer science, *optimisation* problems are problems where we seek the best solution among all possible solutions. Usually this is related to finding the largest or smallest value of a function. In this task, for example, we are asked to find the smallest total time.

Solving a problem using a computer is often related to optimisation. For example, we want our computer to be able to complete tasks in a short time, and using minimal resources such as memory.

In this task, we (or the computer) can find the best solution by trying all possible routes, then choosing the route with the smallest total time as the solution. This method is known as the *brute-force* technique. This technique is suitable for problems with small sizes (in this case, the number of windmills and paths that can be passed). If the size of the problem is large, then the brute force technique takes a long time, because there are so many possible solutions to examine.

Therefore, there are other techniques that can find solutions to problems more efficiently, such as *greedy algorithms* and *dynamic programming*.

Favourite movie

A group of friends want to choose which of seven movies to watch.

Each friend rates each movie by choosing one of these three symbols, shown here from highest to lowest:

✓ Best

– Neutral

✗ Worst

The friends' ratings are shown below. A movie is called a "favourite movie" if all the friends give it their highest or equal-highest rating.

At the moment, there are no "favourite movies". For example, Movie 1 is not a favourite movie because Niklaus gave a higher rating to Movie 4.

Question

Make changes to as few people's ratings as possible, so there is a favourite movie amongst the friends.

	1	2	3	4	5	6	7
Ada	✓	✓	✓	✓	✓	✓	✓
Nancy	–	✓	✓	–	–	✓	✓
Niklaus	✗	✗	✗	–	✗	✗	✗
Grace	✗	–	–	–	✗	–	✗
Edsger	✓	–	✗	✗	–	✓	✓
Rozsa	–	✗	–	✗	✓	–	–

Continued on next page

Favourite movie - continued

EXPLANATION

Answer:

Two people need to change their ratings so that a favourite movie can be found.

One possible solution is to change Niklaus and Rosza's ratings for movie 6.

Explanation:

There is no way to create a favourite movie with only one change to the ratings. For each movie rated, another movie is rated as better by at least two friends:

Movie	Amount of friends who rated other movies better
1	4: Nancy, Niklaus, Grace, and Rozsa
2	3: Niklaus, Edsger, and Rozsa
3	3: Niklaus, Edsger, and Rozsa
4	3: Nancy, Edsger, and Rozsa
5	3: Nancy, Grace, and Edsger
6	2: Niklaus and Rozsa
7	3: Niklaus, Grace, and Rozsa

If Niklaus and Rozsa change one rating each, movie 6 will become the favourite movie. Rosza needs to change her rating to 'best', and Niklaus needs to change his rating to at least 'neutral'.

Alternatively, they could downgrade their ratings for the movies they rated better than 6. For Niklaus that is movie 4, and for Rozsa movie 5.

BACKGROUND INFORMATION

One way to solve this problem is to check each movie, for each person, one by one, to see which movies have better ratings. In the end, you will arrive at a table like the one in the question. You will then know which ratings need changed, in order to find a favourite movie with the smallest number of changes. This is the *brute-force* approach. We can use this algorithm to effectively solve the problem.

But is the algorithm efficient? Could we find a favourite movie faster?

One way to optimise our approach is to search for specific conditions, such as the overall best ratings for each person. This greatly minimises the amount of data we must consider to find a solution. We can still solve the problem effectively, but much more efficiently than the first method allows.

For computer scientists, one of the core considerations is to solve a problem not only effectively, but also as efficiently as possible. Faster hardware makes computers solve problems faster, but if there is no efficient algorithm for a problem, fast hardware does not make much of a difference.

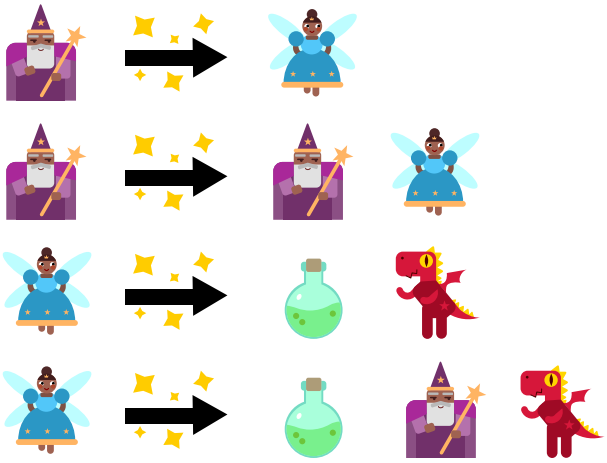
Mysteria

In a land called Mysteria, there lives a wizard.

The wizard can transform into a fairy, or create a fairy to their right.

The fairy can transform into a potion with a dragon on its right, or transform into a wizard with a potion on their left and a dragon on their right.

Before After



These magical transformations can happen any number of times, in any order. That is, any wizard and any fairy can transform at any time. In this way a long arrangement of magical things can be created from a single wizard or fairy.

Question

Starting with a single wizard, which arrangement is **not** possible to obtain?

(A) 

(B) 

(C) 

(D) 

EXPLANATION

Answer

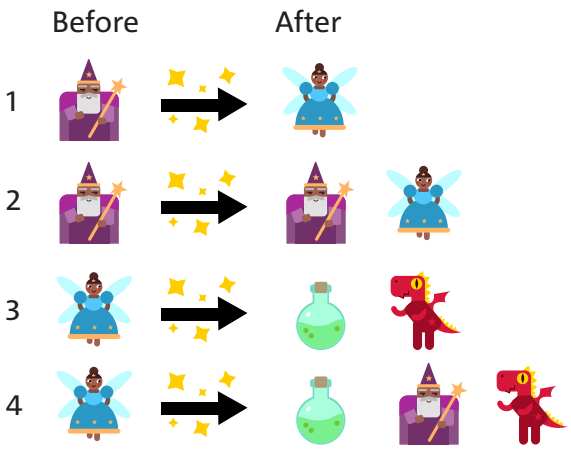
(B) 

Continued on next page

Mysteria – continued

Explanation

Suppose the magical transformations are numbered 1 to 4 as follows:



(A) 

can be obtained by starting with a single wizard and applying transformations, 1, 4, 2 and 3 in that order.

(C) 

can be obtained by starting with a single wizard and applying transformations 2, 2, 3, 4, and 1 in that order.

(D) 

can be obtained by starting with the single wizard and applying transformations 2, 1, 3 and 3 in that order.

While the above is helpful to demonstrate that these three combinations are possible to create, it is much stronger to show that the combination in option (B) is not possible:

(B) 

can be quickly shown as an impossible combination by noticing that the transformation rules always create a potion and a dragon at the same time. Therefore, the number of potions in Mysteria will always equal the number of dragons, which is not the case in this answer option.

BACKGROUND INFORMATION

The magical transformations can be thought of as a set of rules used to generate patterns of objects in Mysteria.

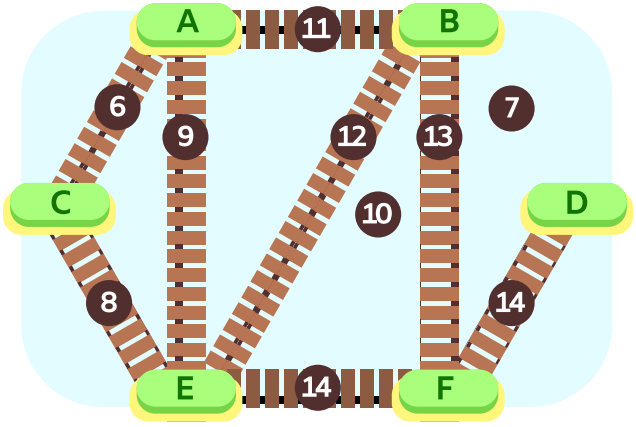
In computer science, a *context-free grammar* is one tool that can be used to describe rules that generate patterns. Context-free grammars can describe languages (both formal and natural), and by repeatedly applying the rules of the grammar you can generate words (or strings) that make up the language.

By using this tool, the question can be thought of as asking which of the 'words' are not part of the Mysteria language.

Connection of islands

A jungle community, living on six islands, wants to connect these islands by building a network of canopy bridges.
A plan of all possible bridges was created. The numbers show the cost to build each bridge, for each possible connection. The bridges do not intersect with each other.
The community wants to link all the islands so that it is possible to travel from any island to any other island. Travel between islands can either be done directly, or by going indirectly through one or more islands.

At the same time, the community wants to build the bridges as cheap as possible.



Question

What is the cheapest whole number that can be spent to link up all six islands?

EXPLANATION

Answer

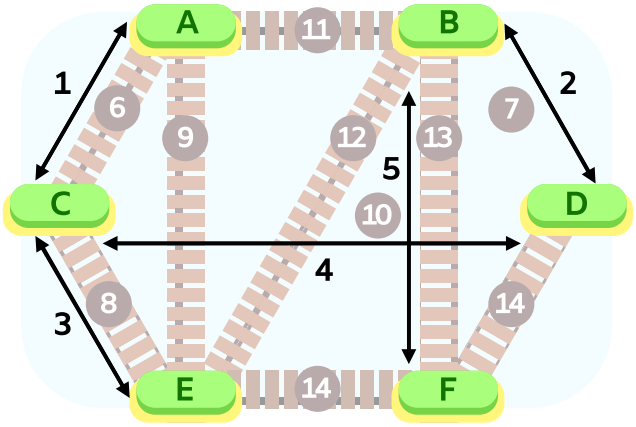
44

Continued on next page

Connection of islands – continued

Explanation

Counting: 6 + 7 + 8 + 10 + 13 = 44



The algorithm used to solve this problem is called Kruskal's algorithm:

1. Start from the cheapest bridge AC – with a cost of 6.
2. Then we choose the second cheapest bridge – BD (7).
3. The third cheapest bridge is CE (8).
4. The next cheapest bridge has a cost of 9. However if we build it, we get a circuit consisting of AECA. It means that we do not need this bridge at all – we can reach islands A, C, and E without building this bridge.
5. The fourth bridge to build is CD (10).
6. The next bridge which costs (11) is a circuit again: ABDCA. Therefore we do not need to build this bridge.
7. In similar way, the bridge BE (12) also makes a circuit, and is unnecessary: ACDBEA.
8. The fifth bridge that we need to build is BF (13).

Now we have all six islands connected. At each step we have chosen the cheapest possible bridge.

BACKGROUND INFORMATION

The given problem requires us to find a minimum *spanning tree* for a given graph. A spanning tree is a subgraph of a given graph which has exactly the same vertices and some of the edges. In this problem, the subgraph is the selection of bridges from the total bridges possible - the islands are the vertices and the bridges are the edges.

Kruskal's algorithm always finds a minimum spanning tree (or shortest path) for a connected weighted graph. The algorithm is based on selecting edges one by one, with the smallest weight, avoiding circuits with already chosen edges.

Ordering

April and Sandy challenged each other to a number game, but they both got different results!

They are trying to form a six-digit number, using each of these digits exactly once:
$(1, 2, 3, 4, 5, 6)$.

The number must be the smallest number that satisfies all these conditions:

$$[4 \to 1, 1 \to 2, 4 \to 5, 2 \to 3, 5 \to 2, 3 \to 6]$$

Condition $a \to b$ means that, in the number, digit a must be to the left of digit b.

Question:

What is the smallest number that meets all of these conditions?

EXPLANATION

Answer:

4 1 5 2 3 6

Explanation:

This number satisfies all conditions, and it is the smallest such number.

The number 4 5 1 2 3 6 also satisfies all conditions, but it is not the smallest such number, because it is larger than 4 1 5 2 3 6.

Here is one way to reach the final number: we consider each condition one by one, and write the relevant digits into a sequence, which satisfies all conditions considered so far. If it is unclear where to put a new digit, we can write it onto a separate line.

4→1	**4 1**
4→1, 1→2	**4 1 2**
4→1, 1→2, 4→5	**4 1 2 5**
4→1, 1→2, 4→5, 2→3	**4 1 2 3 5**
4→1, 1→2, 4→5, 2→3, 5→2	**4 1 2 3 5**
4→1, 1→2, 4→5, 2→3, 5→2, 3→6	**4 1 2 3 6 5**

In the end, we see that there are two ways of ordering 1 and 5, which both satisfy all conditions: 5 1 and 1 5. The second option will result in the smaller number.

BACKGROUND INFORMATION

In this Bebras task, we were only given partial information as to how the digits should be ordered. Yet we were asked to form the complete number as a fully ordered sequence of digits. In computer science, a fully ordered sequence of objects that sorts this type of partial ordering information is called a *topological sorting*.

Computer scientists use methods to compute a topological sorting from partial information – or to find out if it is impossible. For instance, there is no number that will satisfy all these conditions: 2→4, 4→3, 3→2.

This question comes from
South Africa

Years 3+4
Years 5+6
Years 7+8 Hard
Years 9+10
Years 11+12

Underground train network

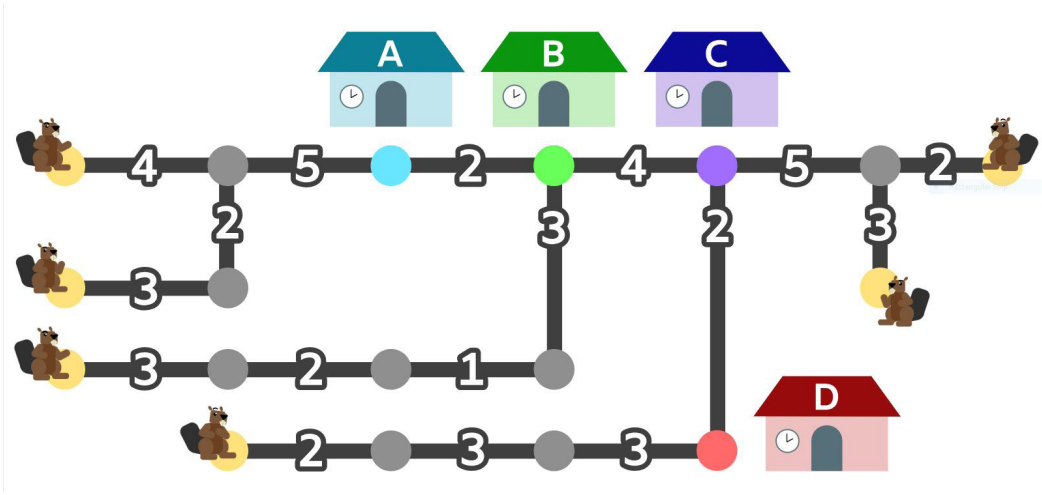
Six beaver friends live in different parts of Beaver City. They use the underground train network to meet up in one of the main stations A, B, C, or D. They want to arrive at the same station in the shortest amount of time.

All beaver friends board the train at different stations, at the same time. The map of trains shows stations (coloured nodes in front of the buildings) on each route and the time it takes to travel between the stations.

It takes one minute to load and unload passengers at each station.

Question:

What station should the friends meet at so that they can see each other as soon as possible?



EXPLANATION

Answer:

Station B.

Explanation:

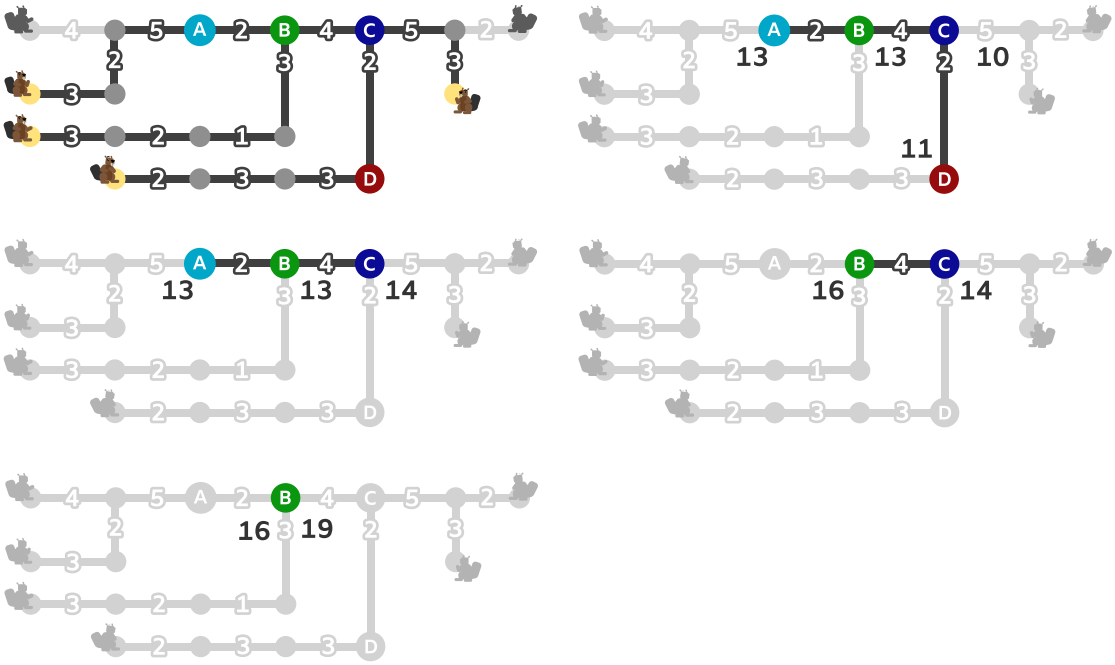
To solve this problem, we must calculate the time it takes each friend to travel, starting from the station where they get on, and ending with each one of the given stations (A, B, C or D). This could of course be solved by calculating every possibility for each friend, to see how long it takes for the last friend to arrive at each station. This would be very tedious and take a lot of time.

To solve this task more promptly, we can use decomposition (break the problem into smaller pieces) and abstraction (decide what routes are most important to calculate and ignore the others). We only have to consider the longest travel times on each route, because we only need to know how long it takes for the last friend to arrive.

Continued on next page

Underground train network – cont'd

Pictured below is one possible sequence of steps we can use to solve this task:



Looking at the two friends who will arrive at station A from the left, we find that for the first friend it takes 11 minutes (1 + 4 + 1 + 5) and for the second friend it takes 13 minutes (1 + 3 + 1 + 2 + 1 + 5). We only have to consider the slower arrival, 13 minutes, from this point onwards. Similarly, two friends on the right side will meet at station C after 10 minutes (1 + 3 + 1 + 5), but since it takes 14 minutes (1 + 2 + 1 + 3 + 1 + 3 + 1 + 2) for the third friend to arrive at station C, we only have to consider this friends route from this point onwards.

We continue in the same way from one station to the next, as shown in the image above, by only considering the slowest arrival time, until we reach the final station (B). This is the best station for all friends to arrive at in the shortest possible time: 19 minutes.

This result is verified in the table below. The number of minutes needed for the friends to arrive at each station, from each direction, is listed. We can see that it takes 22 minutes for the last friend to arrive at station A, 19 minutes for station B, 21 minutes for station C and 24 minutes for station D. This confirms that station B is the best station for the friends to meet at.

Travel time to A	Travel time to B	Travel time to C	Travel time to D
From left: 13	From left: 16	From left: 21	From left: 24
From right: 22	From right: 19	From right: 10	From right: 11

BACKGROUND INFORMATION

In computer science, *dynamic programming* means simplifying a complicated problem by breaking it down into simpler sub-problems. It is used heavily in optimisation problems (finding the maximum and/or minimum of something).

The type of diagram used in this task is called a *graph*. The stations are the nodes and the rails are the edges of the graph. More specifically, this is an example of a *weighted* graph, since the edges have particular values (travel duration) assigned to them. This means that the graph has weighted edges. Weights in graphs may represent things such as costs, lengths, weights or capacities.

Bebras Challenge 2023 Round 1

Years 9+10

Nuts and bolts

At the Beaver Construction factory, Benoit works on the nut  and bolt  assembly line

His job description is as follows:

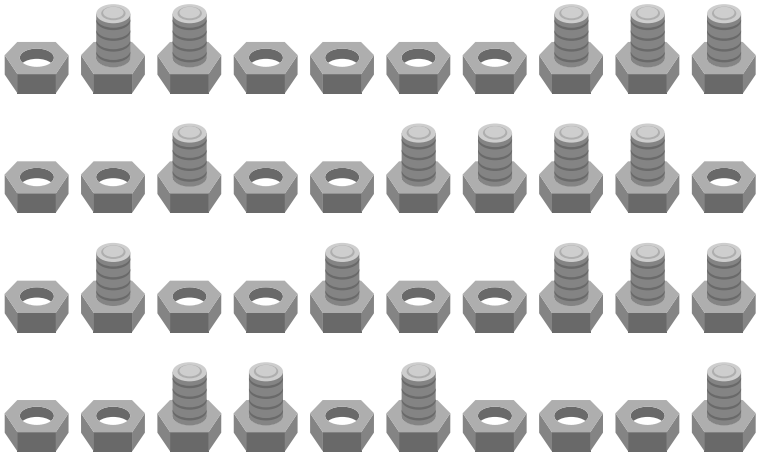
- Stand at one end of a conveyor belt, which brings him a line of nuts and bolts.
- Take each item, either a nut or a bolt, off the conveyor belt as they arrive.
- Put the nuts from the conveyor belt in the red bucket.
- Attach any bolt taken from the conveyor belt, to a nut from the red bucket, and place the assembled part onto the table behind him.

Things can go wrong for Benoit in two different ways:

1. When Benoit takes a bolt from the conveyor belt there may not be a nut in the bucket.
2. There may be no more nuts or bolts on the conveyor belt, even though there are still nuts in the bucket.

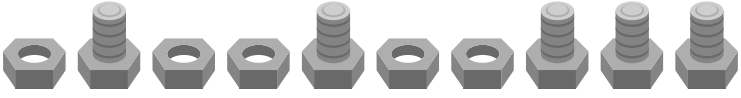
Question:

Which sequence of nuts and bolts, when processed from left-to-right, will not cause things to go wrong for Benoit?



EXPLANATION

Answer



Continued on next page

Nuts and bolts

Explanation

We can keep track of the state of the bucket and conveyor belt, from left-to-right:

N = 

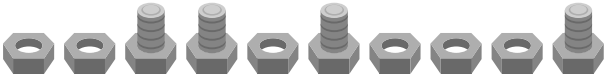
B = 

Bucket	Conveyor Belt
empty	N B N N B N N B B B
N	B N N B N N B B B
empty	N N B N N B B B
N	N B N N B B B
N N	B N N B B B
N	N N B B B
N N	N B B B
N N N	B B B
N N	B B
N	B
empty	empty

Looking at the other answers:

will go wrong after N B B, since there will be no nut in the bucket when the second B is encountered.

will go wrong after N N B N N B B B B, since there will be no nut in the bucket when the fifth B is encountered: notice there are only 4 N's before this B.



will go wrong after the entire sequence is processed, as there will be two nuts in the bucket, since there are 6 N's and 4 B's.

BACKGROUND INFORMATION

This task highlights the use of a *push-down automaton* (PDA). A PDA is a way of describing an algorithm that relies on the current state, but also has an unlimited amount of memory in the form of a stack. In this task, the state is either having a nut or having a bolt on the conveyor belt, and the stack is the bucket which holds the nuts.

A PDA can be used to recognise, or *parse*, *context-free languages*.

To recognise or parse a language means to determine if a given sequence of symbols belongs to the language. In this case, we can think of the nuts and bolts as a representation of *balanced parentheses,* where N = (an opening bracket, and B =) a closing bracket. That is, balanced parentheses are valid arrangements of parentheses in arithmetic expressions.

Examples of a sequence of parentheses which are not balanced are (((() or ())(. Detecting balanced parentheses is important in *compilers*, since many programming languages rely on parentheses to indicate *nested scopes* as well as arithmetic expressions.

This question comes from
Cyprus

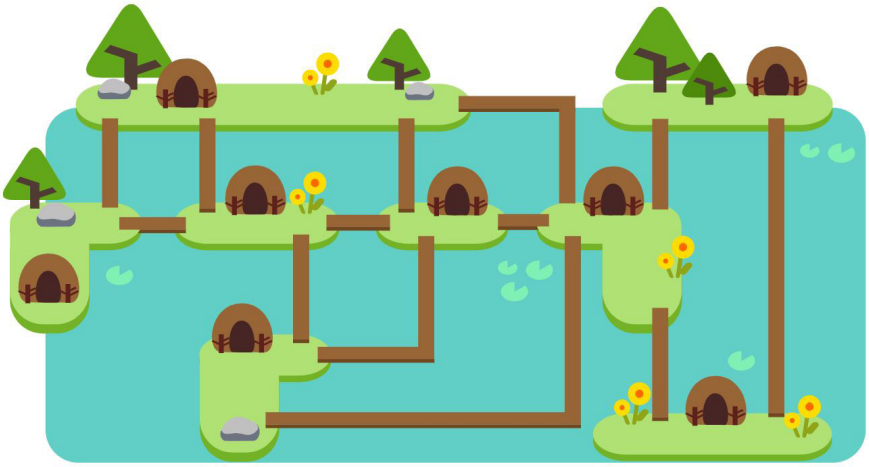
Mary's neighbours

John wants to visit his friend Mary, but he doesn't know where she lives. Fortunately, he has a map and some information:

- People think of themselves as neighbours if they can visit each other by crossing only one wooden walkway.
- Mary, Zac, and Pan, have four neighbours.
- Zac and Pan are neighbours with Niki.
- Niki has no other neighbours.

Question

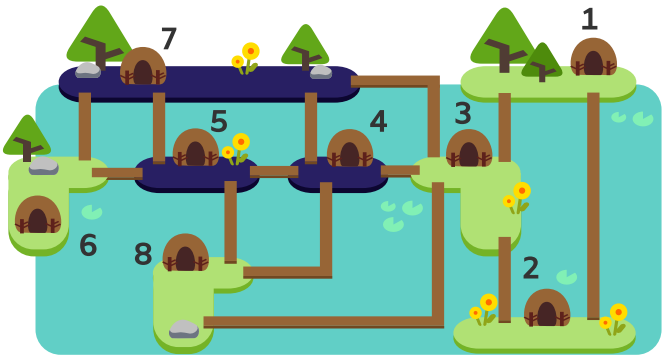
Using the information above, find out which house is Mary's.



EXPLANATION

Answer

House 4, as shown below

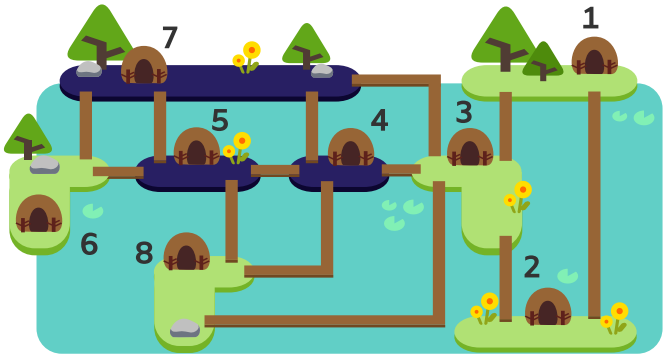


Continued on next page

Mary's neighbours – continued

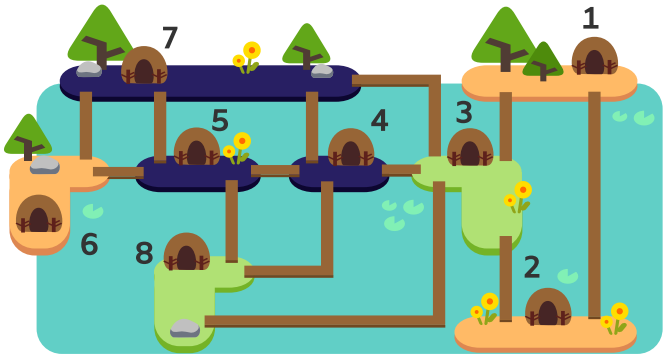
Explanation

To solve this problem, we must focus on the paths that go between each house. We first need to identify the houses with four paths. There are three such houses: 4, 5 and 7.



Mary, Zac and Pan each live in one of these three houses, but we still need to find out exactly where Mary lives.

The other two pieces of information describe Niki's house. We can deduce that there are only two paths attached to Niki's house. Therefore, Niki must live in one of the houses labelled 1, 2 or 6.



As Niki is neighbours with both Pan and Zac, who each have four neighbours, we can further deduce that:

- Niki lives in house number 6
- Zac and Pan live at numbers 5 and 7 (or the other way around)

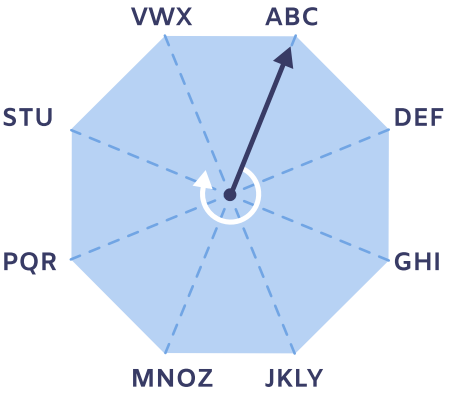
There is only one other house with four paths, so this must be Mary's house!

BACKGROUND INFORMATION

Graph theory is the study of graphs used to model pairwise relations between objects. A graph may be seen as a set of *nodes* (also called vertices or points) connected with edges (also called links or lines). In this task, the houses represent nodes, and the paths represent edges.

Graphs can be helpful when describing and solving networking problems. For example, finding a good spot for a router in a building, or making sure that each of the houses in a neighbourhood has a strong wi-fi signal.

Cipher 8



An octagon has three or four letters written at the corners. An arrow points from the centre of the octagon to a letter group. The arrow can rotate clockwise.

Messages can be encrypted using the octagon and arrow.

At the beginning of the encryption of a new message, the arrow always points to the letters ABC.

Each letter of the message is encrypted so that:

- The **first number** indicates how many corners of the octagon the arrow should be rotated, from its current position.
- The **second number** indicates the position of the encrypted letter in the letter group to which the arrow points.

Example:

TREE is encrypted with the sequence 62-73-42-02.

Question

What is the encrypted message for WATER?

72-11-26-32-53 | 62-11-62-22-43 | 62-11-26-22-53 | 72-11-62-32-43

EXPLANATION

Answer

72-11-62-32-43

Continued on next page

Cipher 8 – cont'd

Explanation

We can find the correct answer by creating an encrypted text for the message WATER ourselves, using the octagon and the rules provided.

The encrypted text will contain five codes, each made of two digits:

- At the start of the encryption, the arrow points to the letter group ABC. The letter W is located within the letter group that the arrow will point to after rotating 7 vertices. The letter W is the second letter in the letter group VWX, so the first code is 72.
- The second letter of the encrypted message is A. This letter is in the letter group ABC, which the arrow will point to after rotating once from its current position at letter group VWX. A is in the first position, so the second code is 11.
- We get to the letter group containing T by rotating a further 6 vertices. The letter T is the second letter in this set, so the third code is 62.
- The letter E is in the letter group that the arrow will point to after the arrow is rotated another 3 vertices. E is the second letter in the group, so the code is 32.
- The last letter is R, found in the letter group that can be reached after rotating 4 vertices from this point. R is the third letter, so the last code is 43.

In full, the final encryption is 72-11-62-32-53.

BACKGROUND INFORMATION

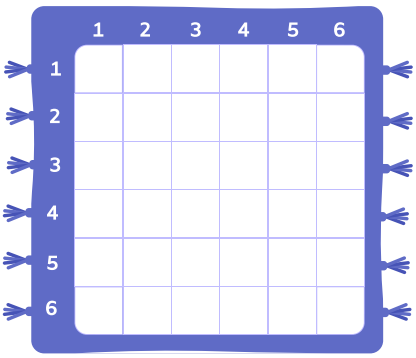
One method of protecting data from unauthorised people is *encryption*. Cryptography began as early as 3500 years ago. One of the simplest methods of encryption is to replace each letter with a different letter.

In this Bebras problem, by introducing the rotating arrow and by grouping the letters in sets the encryption method becomes more complicated. By introducing these factors it may be more difficult to crack the code if you don't understand the rules behind it. However the cipher is simple enough that once a person understands how to solve it, it is quite easy to do.

This system of encrypting messages is reliable is because it follows an algorithm. This means we could write a program to encrypt messages this way. It also means we could just as easily write a program to decode such messages - if we know how the system works.

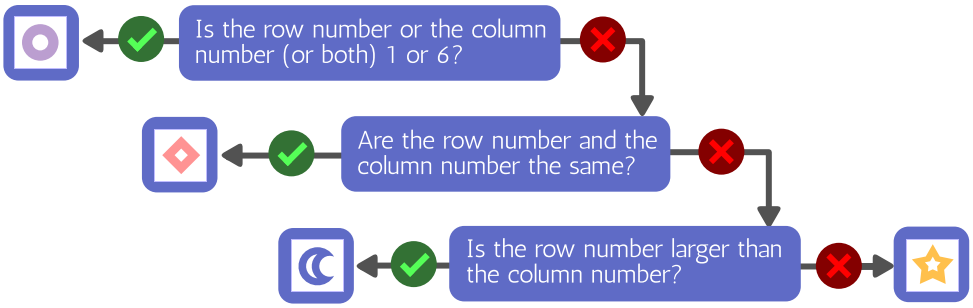
If we altered the letters in each section, or if we put them in different groups of two or three, we would get other means of encryption. Cryptanalysis experts, who try to crack codes, might have more trouble unraveling our encryption scheme this way.

Rug weaving



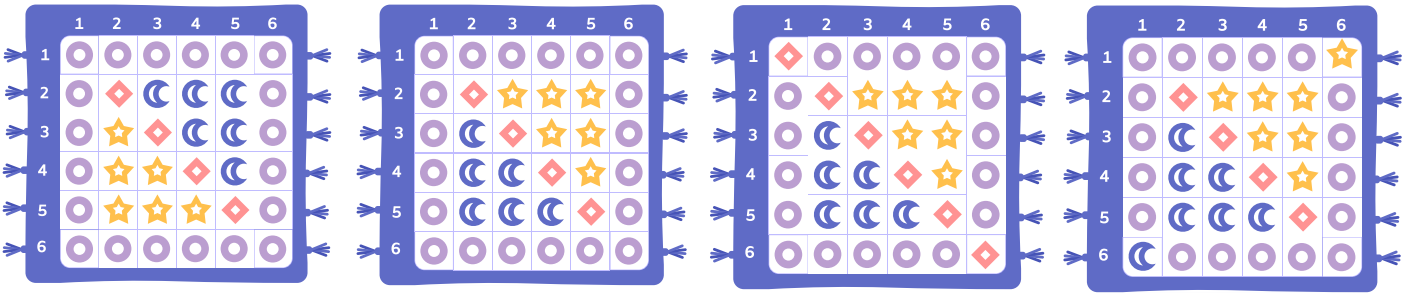
Hale is a Turkish weaving artist.
She is making a plan for a square rug with 6 rows and 6 columns.

Hale puts a symbol in each square on the grid of the rug, using the following questions:



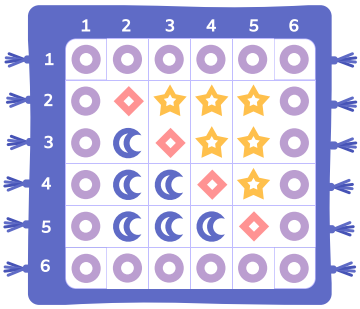
Question

Using this method, which of the four rug plans below will Hale make?



EXPLANATION

Answer

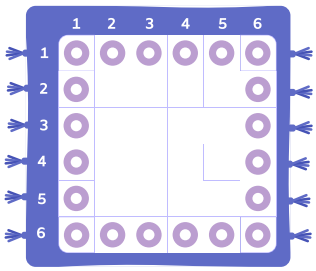


Continued on next page

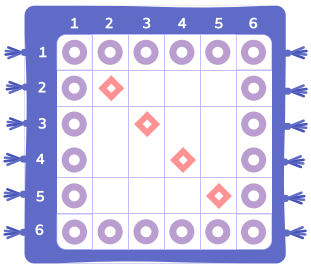
Rug weaving – continued

Explanation

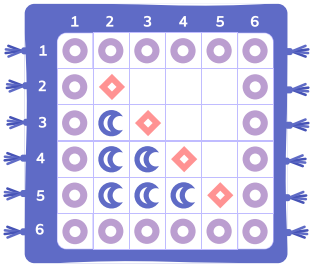
The first question that Hale asks indicates that the 1st and 6th rows, as well as the 1st and 6th columns, should have purple circles in them.



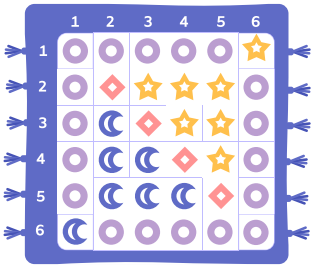
The second question leads to all remaining squares where the row and column numbers are the same, having a red diamond symbol. This results in a diagonal line in the middle of the rug.



Next, we find that the row number is larger than the column number in all the squares to the left of the diagonal line we just discovered. In these places, a blue crescent moon symbol is placed:



Finally, the remaining squares to the right of the diagonal line all have row numbers that are not larger than the column numbers. Therefore, they are filled in with the yellow star symbol, giving the answer seen in option B.



BACKGROUND INFORMATION

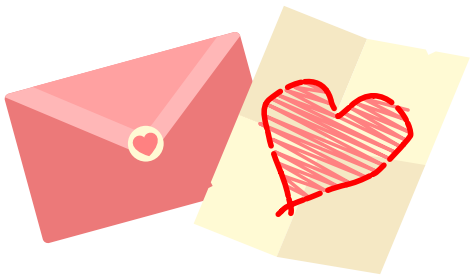
In mathematics and computer science, an *algorithm* is a finite sequence of well-defined instructions, typically used to solve a class of specific problems or to perform a computation. Algorithms are used as specifications for performing calculations and data processing.

A *decision tree* is one way of easily displaying the structure of an algorithm that is made up of conditional statements – such as IF/ELSE statements. This means that each layer of the decision tree consists of a question or statement – if the question/statement is true, the algorithm follows one branch of the tree, otherwise the algorithm follows the other branch of the tree. This continues through the different branches until the outcome of the algorithm is reached.

By making use of artificial intelligence, algorithms can perform automated deductions (referred to as automated reasoning) and use mathematical and logical tests to divert the code through various routes (referred to as automated decision-making).

This question comes from Uruguay

In love



Beaver Dai received a heart drawing as a secret gift.

It must have been sent by one of her four best friends - David, John, Robert or James.

Each of them makes a statement:

- David said - "John sent the drawing."
- John said - "The gift is from James."
- Robert said - "I didn't send it."
- James said - "John is lying."

It is a habit among Dai's four best friends that only one of them tells the truth.

Question

Who sent the drawing?

John | David | James | Robert

EXPLANATION

Answer

The correct answer is Robert.

Explanation:

If John sent the drawing, then David, James, and Robert were telling the truth.

If James sent the drawing, then John and Robert were telling the truth.

If David sent the drawing, then Robert and James were telling the truth.

If Robert sent the drawing, then James was telling the truth.

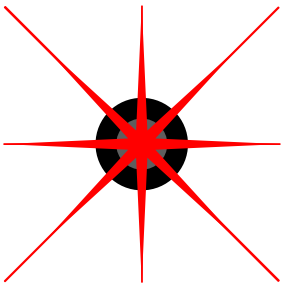
As we know that exactly one of the four statements is true, Robert must have sent the drawing.

BACKGROUND INFORMATION

The four friends in this Bebras task make simple statements. Each of these statements, or *propositions*, can only be either true or false. In 1847, George Boole formulated ways to calculate by using such simple propositions. Such a *calculus* (a set of rules telling how to calculate) tells us how to construct true statements from other true statements without considering their real meaning. Computers work with propositional logic as well. The smallest unit of information - the bit - is like a proposition, as it can only carry two different values, corresponding to true or false, or 1 or 0. Objects that follow these rules are called *Boolean*.

Bits, and propositional calculus working on bits, can be easily and inexpensively implemented in computer hardware. This is the foundation for the huge success of digital computers – and computer science.

Super security system



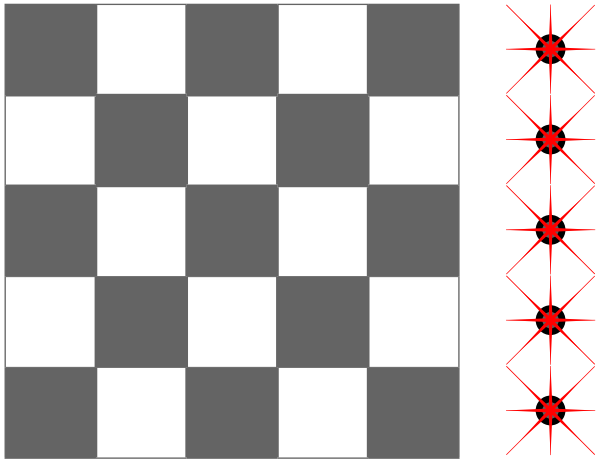
Beaver Business Banking needs a new security system. They want to secure the room in front of their safe from intruders.

They bought five detectors. Each detector sends out laser beams in eight directions until they hit an object or a wall.

If an intruder crosses the laser beam of any detector, the alarm is activated. But if one of the detectors stands in the way of a laser beam of another detector, the alarm goes off as well!

Question

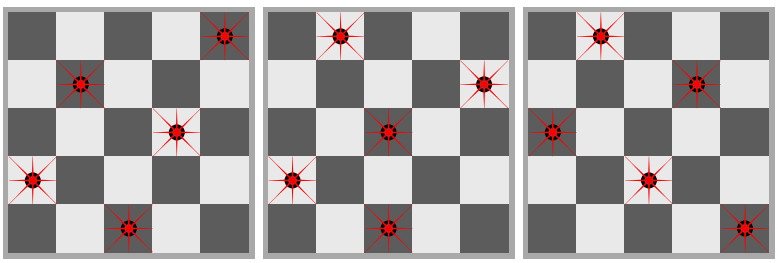
Position the five detectors on the square floor tiles, so that they secure the remaining tiles in the room, but do not activate each other.



EXPLANATION

Answer

There are multiple solutions to this problem. Here are three:



Continued on next page

Super security system – cont'd

Explanation:

To create a solution, we can first identify that one detector at most can be placed in each row and in each column. This is because they send out horizontal and vertical beams, and must not detect each other. But since the detectors also send out beams diagonally, care must be taken so that no two detectors will be placed on the same diagonal line.

More generally, in a correct solution, the following conditions must exist for any pair of two different detectors. x is the number of the column in which the first detector stands, and y is the number of the row of that detector. a is the number of the column in which the second detector stands, and b is the number of the row of that detector:

- *x* and *a* cannot be the same due to the vertical beams;
- *y* and *b* cannot be the same due to the horizontal beams:
- when subtracting the smaller of x and a from the larger one, and doing the same for y and b, the results of those two subtractions cannot be the same, due to the diagonal beams.

BACKGROUND INFORMATION

This problem is a variation of the eight queens puzzle, which is an example of a *constraint satisfaction problem*. You have multiple items, which can have different values, and for every item you must satisfy the values. You must do this in a way that does not violate a given set of rules.

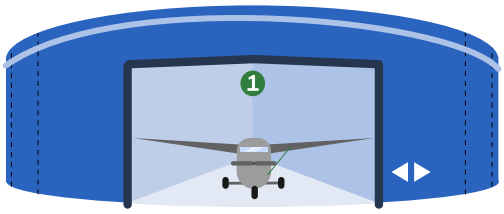
Such problems often occur in *automated planning* which is used to make robots work, in *computational linguistics* which are required for language recognition, or in *resource allocation* when businesses decide who uses certain assets. These problems can get complicated very quickly to the extent that a computer is required to solve them. Computers typically go ahead in the same way a human probably would; they try to build a solution little by little. As soon as a rule is violated, they go back one or more steps, then try the next option until they find a solution that fits all constraints. This process is called *backtracking*.

Hangar carousel

At Beavertown airfield, six planes are parked on a rotating turntable in a round hangar.

The turntable can be rotated using a control panel with two arrow buttons ◀ ▶.
One button press rotates the turntable exactly one parking position, either left or right.

The gate of the hangar is wide enough for one plane to be rolled out.
The turntable is very slow to rotate, so having fewer button presses will avoid delays.



In the mornings, when pilots come to pick up their planes, the parking position 1 is always at the gate.
The pilots do not all arrive at the same time.

For example, one morning the pilots arrive such that the planes must be accessed in the order:
4, 3, 5, 2, 1, 6.

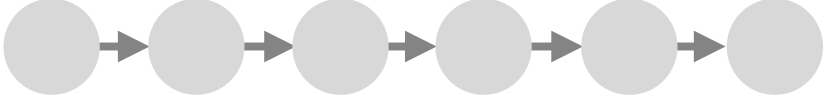
This can be achieved by pressing ▶▶▶◀▶▶◀◀◀◀◀ (11 presses).

The control tower is trying to prepare for a worst-case-scenario, where the pilots arrive in an order that requires the most number of button presses to solve.

Question

Assuming that the most efficient number of presses is always used to move to the next required plane, provide a worst-case order for pilots to arrive at the airfield to access their planes.

1 2 3 4 5 6



Continued on next page

Hangar carousel – continued

EXPLANATION

Answer

There are two worst-case orders of parking positions: **4 1 3 6 2 5** and **4 1 5 2 6 3.**

Explanation:

The worst case and hence the correct answer can be found by consistently selecting the next parking position that is farthest away from the gate. Because you can move the turntable in both directions, there is more than one correct solution; with six planes there are two correct solutions.

4 1 3 6 2 5 :

- First, accessing the position 4 requires three presses (either left or right)
- Next, accessing position 1 requires three presses (either left or right)
- Next, accessing position 3 requires two presses right
- Next, accessing position 6 requires three presses (either left or right)
- Next, accessing position 2 requires two presses right
- Finally, accessing position 5 requires three presses (either left or right)

4 1 5 2 6 3:

- First, accessing the position 4 requires three presses (either left or right)
- Next, accessing position 1 requires three presses (either left or right)
- Next, accessing position 5 requires two presses left
- Next, accessing position 2 requires three presses (either left or right)
- Next, accessing position 6 requires two presses left
- Finally, accessing position 3 requires three presses (either left or right)

BACKGROUND INFORMATION

In computer science, assessing the *efficiency* of a process plays a very important role. While best cases for a problem are good to know, the worst case is more significant when trying to determine efficiency. Efficiency means that resources such as time, energy and storage space are used economically.

An example of efficient warehousing is practiced today in the mail order business: In order to access goods efficiently, they are often no longer organised according to the alphabet or the article number. Instead, they are stored according to sophisticated algorithms that take other factors into account, for example, how often the goods are usually bought.

Sometimes it is not always possible to be efficient in every category. In this task the advantage of a round hangar is that it saves a lot of space when parking planes. However, pilots arriving in the worst case scenario leads to much time lost, so there is a trade-off between space and time considerations.

Overlapping villages

As years went by, the villages of

Cabbageville ,

Strawberrton , and

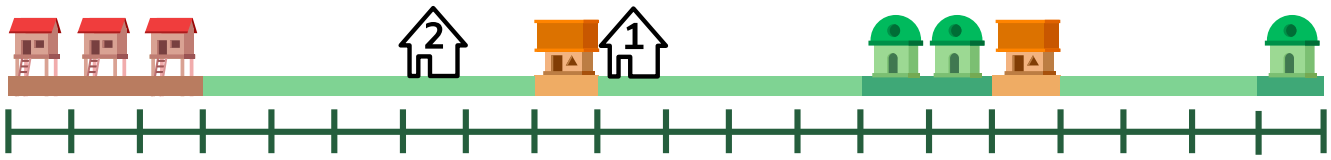
Carrotford , grew and started to overlap.

Whenever a new house is built, the villagers use the following rules to decide which village the house is assigned to:

The new house belongs to the village most houses belong to, amongst the X nearest houses.

- If there are multiple villages with the same number of nearest houses, ties are broken by assigning the new house to the same village as its nearest neighbour house.

Two new houses are to be built as per the image below. They will be assigned to villages using the value of X. House 1 will be built and assigned before House 2.



Question

What is the smallest value of X so that House 2 is assigned to Strawberrton ?

EXPLANATION

Answer

5.

Continued on next page

This question comes from
Lithuania

Years 3+4
Years 5+6
Years 7+8
Years 9+10 Medium
Years 11+12

Overlapping villages – cont'd

Explanation

If X=1, both House 1 and House 2 are assigned to Carrotford  .

If X=2, House 1 is still assigned to Carrotford  , since the nearest of its two neighbours is from Carrotford. House 2 is also assigned to Carrotford, since its nearest neighbours are both from Carrotford.

If X=3, House 1 is assigned to Cabbageville  , since two of its neighbours are from Cabbageville. House 2 is assigned to Carrotford , since its neighbours are all different, but the nearest one is from Carrotford.

If X=4, House 1 is assigned to Carrotford  , since it has two neighbours from Cabbageville  and two from Carrotford, and the nearest one is from Carrotford. Therefore House 2 also is assigned to Carrotford, since it has two Carrotford neighbours and two Strawberrton  neighbours, but the nearest one is from Carrotford.

If X=5, House 1 is assigned to Carrotford  , similar to when X=4. House 2 is assigned to Strawberrton  , since three out of its five neighbours are from Strawberrton.

When X=6 or X=7, House 2 is also assigned to Strawberrton , but these are not the smallest values of X when this will occur.

BACKGROUND INFORMATION

The rule to assign each house to a village is an example of a classification algorithm called *k-nearest neighbors* (kNN). kNN classifies each new data item by looking for the most similar items that have already been classified, using the value of k. In this task, the variable X plays the role of k.

In *machine learning*, kNN is often used because it is easy to implement and does not require fitting a complex model to the given data.

This question comes from Germany

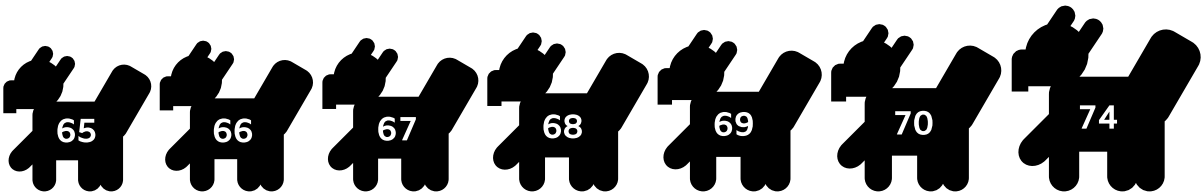
Beaver AI

A group of student beavers construct an “astonishingly intelligent” (AI) system. The system is to measure an animal’s size and, based only on that, to decide whether the animal is a beaver or not. The AI system learns to make its decision from examples. At first, the AI system learns from example animals with these sizes:

- 65, 66, 67, 68, 69 => beaver
- 11, 101, 110, 120, 130 => no beaver

Then, the beavers let the AI system decide. The outcome is as follows:

- 70, 74 => beaver
- 86, 38 => no beaver
- 40, 80 => beaver



The AI made a mistake, as the two animals of sizes 40 and 80 are in fact not beavers!

The students realised that this mistake happened because the AI originally observed that an animal with size 11 **is not** a beaver, and an animal with size 65 is a beaver. Looking at the difference in sizes, the AI had decided that only animals with sizes greater than 38 and less than 85 are beavers.

Therefore, in order to improve the AI, the beavers give it a new example: an animal with size 42 is **not a beaver.**

Question

After the new training, how does the AI classify two animals of size 48 and 84?

beaver, beaver | beaver, not beaver | not beaver, beaver | not beaver, not beaver

Continued on next page

This question comes from Germany

Years 3+4
Years 5+6
Years 7+8
Years 9+10 Medium
Years 11+12

Beaver AI – cont'd

EXPLANATION

Answer

not beaver; beaver

Explanation

We can see from the initial diagnostics by the students, the AI had seen 11 was not a beaver, and 65 was a beaver. It had therefore computed a threshold for being a beaver or not and put it somewhere between 11 and 65.

The AI had (somewhat) sensibly set this threshold to be the average point between them, (11+65)/2 = 38 and a similar reasoning can be applied on the other extreme: (69+101)/2 = 85.

Using the same procedure, after having seen a new example (42 is not a beaver) the 85 threshold should not change, but the new left threshold should become (42+65)/2 = 53.5.

Recognising this, an animal of size 48 would be identified as "not beaver", while an animal of size 84 would be identified as a beaver.

BACKGROUND INFORMATION

The "Astonishingly Intelligent" system in this Bebras task applies a *Machine Learning algorithm*. In the area of Machine Learning (ML) computer scientists study computer algorithms that can improve their decision making automatically. ML algorithms build a model based on example data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so. Hence, their quality may strongly depend on the quality of the training data used, as can be seen from this task.

In this case, it is relatively easy to see how the ML algorithm is calculating the threshold for defining a beaver from a non-beaver. In complicated systems, such as in image-recognition algorithms, the way the ML algorithm is 'thinking' is far more complicated (and sometimes unknowingly complicated) to determine.

ML algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

The printer

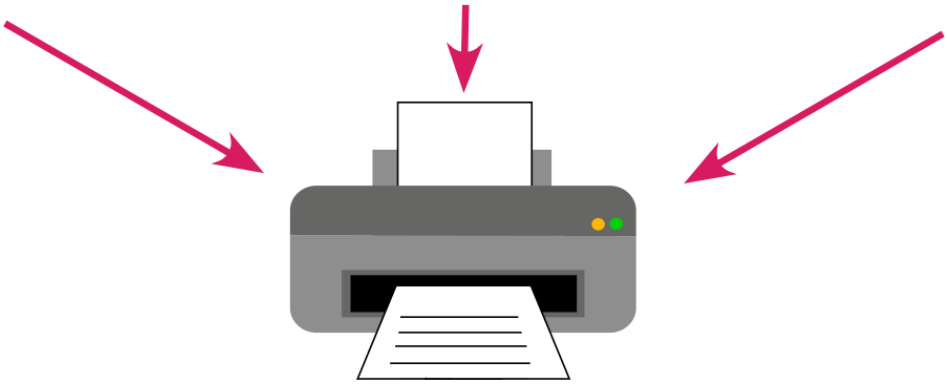
Javier, Kara and Lucy share the same printer on the same network. When one of them prints a document, the printer starts and prints the whole document. This happens even if, during this time, the order to print another document has been given.

Today is a busy day at the office. The following tables show the printing times of each document and when the print order was given, for each colleague. If the printing for a document starts at 10:00 and lasts 2 minutes, then the print will finish at 10:02. No processing (waiting) time is necessary between two consecutive print jobs.

Order no.	Time	Print duration
11	10:00	2 minutes
12	10:03	1 minute
13	10:10	5 minutes

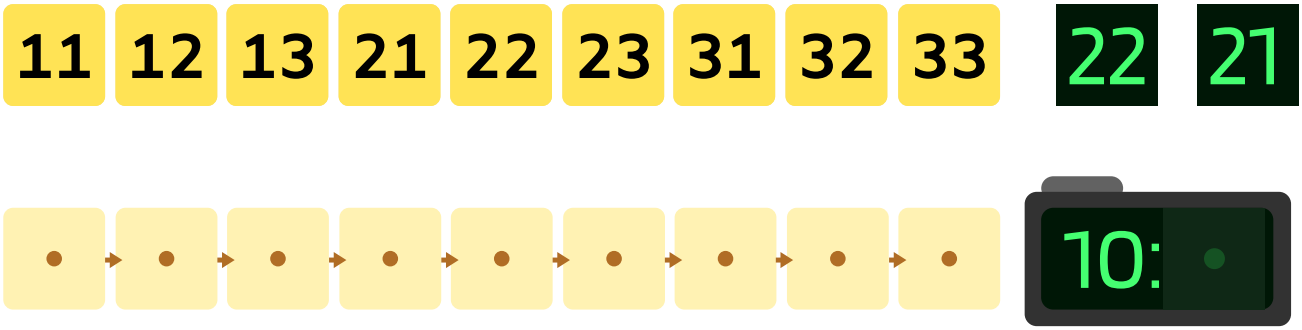
Order no.	Time	Print duration
21	10:01	1 minute
22	10:04	3 minutes
23	10:12	1 minute

Order no.	Time	Print duration
31	10:08	5 minutes
32	10:05	3 minutes
33	10:13	1 minute



Question

In what order will the documents be printed, and at what time will the printer finish printing?



Continued on next page

This question comes from
Romania

Years 3+4
Years 5+6
Years 7+8
Years 9+10 Medium
Years 11+12

The printer – continued

EXPLANATION

Answer

11, 21, 12, 22, 32, 31, 13, 23, 33 and the finish time will be 10:22.

Explanation

Order no.	Start Time	Duration	Finish Time
Order 11	10:00	2 min	10:02
Order 21	10:01	1 min	10:03
Order 12	10:03	1 min	10:04
Order 22	10:04	3 min	10:07
Order 32	10:05	3 min	10:10
Order 31	10:08	5 min	10:15
Order 13	10:10	5 min	10:20
Order 23	10:12	1 min	10:21
Order 33	10:13	1 min	10:22

The table below shows the order in which the printer will print the documents and time of completion for each document:

BACKGROUND INFORMATION

In this question, the printer uses a data structure called a *queue* to print documents. The queue is an abstract data structure in which the addition operation is performed at one end, and the deletion operation is performed at the other end. During queue operations we have access to only one element, the one at the beginning of the queue - the element that is about to be removed. The following operations can be performed with a queue:

- Creating an empty queue;
- Checking if there are still elements in the queue
- Adding a new item in the queue (push;)
- Remove an item from the queue (pop;)

Identifying the value of the element at the beginning of the queue (first)

These operations are similar to how a queue works at a supermarket. Buyers come and sit in line and the order in which they buy things is the one in which they arrived. Because the delete operations are done in the same order as the add-ons, the queue is a *FIFO (First In First Out)* data structure. In computer science, a queue is used in many situations, such as:

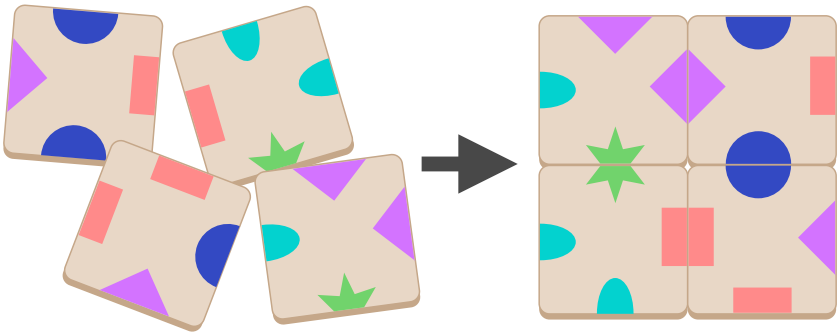
- Sending messages
- Information flow: if information is transferred faster than it is recorded it is put in a queue
- Computer networks

Four tiles

Lena Beaver likes to make games for her friends. Her favourite game is giving them four tiles which need to be arranged into a 2 x 2 square, according to the following rule:

- Tiles may only touch each other at sides that carry matching symbols.

Example:

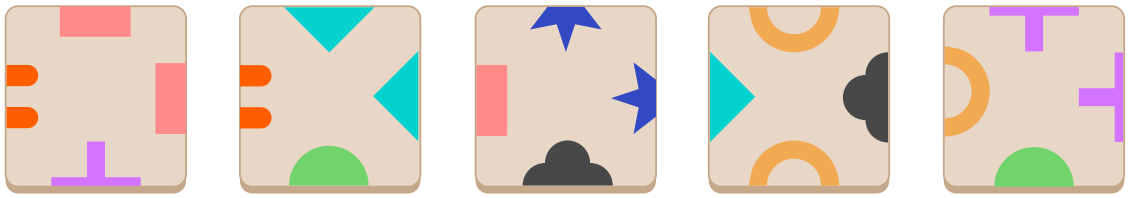


Lena accidently included an extra tile in one of her sets. These five tiles are shown below.

Despite this, Lena's friends managed to arrange four of the tiles into a 2 x 2 square, that follows the rule above. (In this case, there is only one possible choice of four tiles that allows this.)

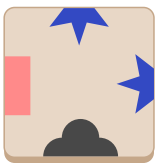
Question

Which tile did Lena's friends **NOT** use?



EXPLANATION

Answer

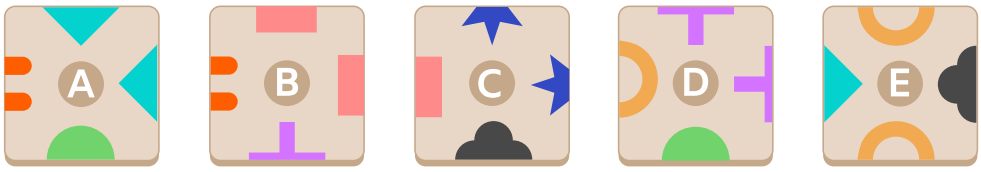


Continued on next page

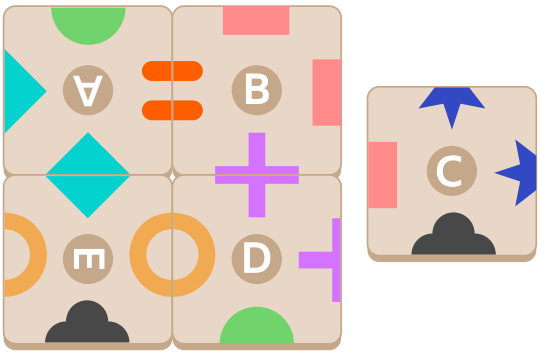
Four tiles – continued

Explanation

We assign letters to the tiles as follows:



It is possible to create a 2 x 2 square that follows the rule using tiles A, B, D and E, as shown below:

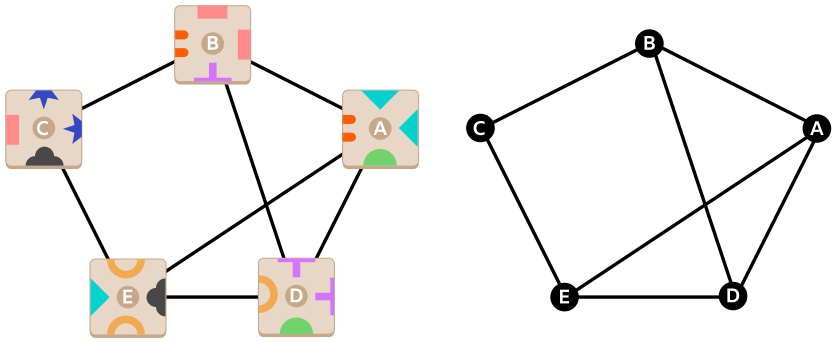


Trying the many possibilities until one happens to fit is one option, but there is a way to reduce the number of possibilities that have to be checked.

- We do this in two separate steps.
- We make a diagram of the five tiles.

Tiles are connected with a line if they share at least one symbol.

C and E are connected by a line, because both tiles have a little black cloud on one of their sides. A and C are not connected, because none of the symbols on A occurs on C. (The simplified diagram on the right is called a graph.)



Continued on next page

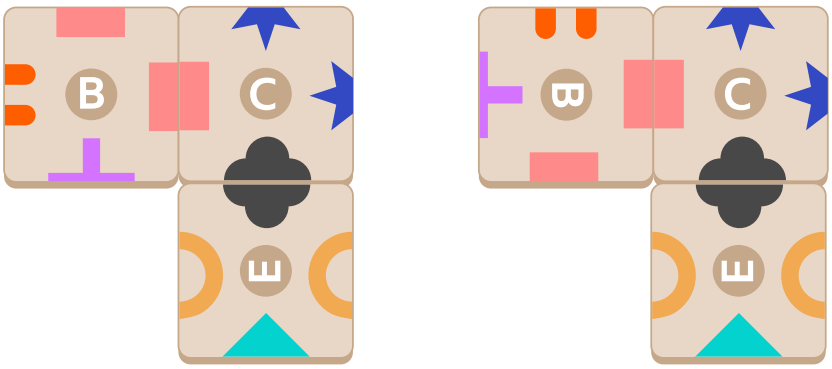
Four tiles – continued

We are looking for a path from tile to tile that follows the lines and ends up at the start after 4 steps from one tile to the next. In our diagram, there are three of these "4-cycles":

- The first goes from A to B to D to E and then back to A (That 4-cycle corresponds to our solution).
- The second is A-B-C-E-A.
- The third is B-C-E-D-B.

Of course, all cyclic permutations of the five letters (i.e. starting from any letter, but keeping the order) are equivalent, as well as their reverse. For example, A-B-C-E-A, B-C-E-A-B and E-C-B-A-E are equivalent.

Both the second and third cycles contain B-C-E, and there are only two possibilities to join these tiles in that order:



Neither tile A nor D can be used to complete these to a 2 x 2 square (and still conform to the rule). Therefore, the 2x2 square above is the only solution possible, so the tile not used must be tile C.

BACKGROUND INFORMATION

Graphs are often used in computer science as an abstraction of the data that needs to be processed. Many clever and fast methods have been devised to determine various properties of such graphs, for instance, to easily find all 4-cycles as seen in in this task.

In these graphs, information like the tiles in this task are represented by nodes, and they are connected by *edges*. This representation helps computer scientists to visualise paths through the nodes.

This technique is used to solve all kinds of computational thinking problems, from evaluating transport systems, to analysing social networks, and even internet connections.

This question comes from Canada

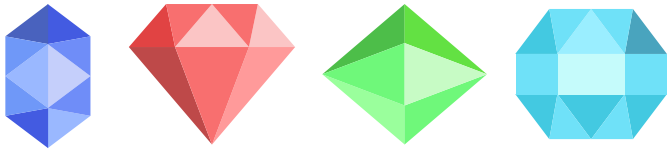
Favourite gem

Troy has a collection of gems. He ranks his gems from his overall favourite to his least favourite.

Sarah knows which gems are in Troy's collection, but she does not know how he has ranked them.

Sarah has a plan to find out which gem is Troy's overall favourite:

- Sarah chooses 4 of Troy's gems and asks Troy: "Out of this group of 4, which gem is your favourite?"
- Sarah chooses a new set of 4 gems and asks her question again.
- Then she chooses a third set of 4 gems and asks her question for the last time.



Note: When Sarah chooses her second and third set of 4 gems, she may sometimes include gems she has chosen before.

Question

If Sarah is to successfully find Troy's overall favourite gem using this plan, what is the largest possible number of gems in Troy's collection?

7 8 9 10 11 12

EXPLANATION

Answer

10

Continued on next page

This question comes from Canada

Years 3+4
Years 5+6
Years 7+8
Years 9+10 Hard
Years 11+12

Favourite gem – continued

Explanation

With 10 gems, Sarah can ask Troy about eight different gems with the first two requests. Troy's answer to each of these requests is a candidate to be his favourite of all his gems, but the other three cannot be. Therefore, with Sarah's third and final request, she can include these two candidates and the two gems that have not yet been part of a request. Troy's answer to this third request must be his favourite gem.

We have shown that there is a strategy of Sarah's that works if there are 10 gems. (Note that there are other correct strategies that work for 10 gems.)

If Troy has at least 11 gems, we consider Sarah's first two requests.

If at least one gem is part of both requests, then at least four gems will be unconsidered after the first two requests. In this case, Sarah must ask about these four unconsidered gems, because if she doesn't, an unconsidered gem could be Troy's favourite. On the other hand, she won't have any information about Troy's favourite among the last four unconsidered gems compared to the other seven gems. In this case, her strategy cannot determine which gem is Troy's most favourite.

If there is no gem that is part of both of Sarah's first two requests, then candidates for Troy's favourite include the two answers to these requests and the three remaining unconsidered gems. This is a total of five possibilities for Troy's most favourite gem, and Sarah does not have any information about how they rank, so her strategy cannot work.

This means that Sarah's plan will not work if there are more than 10 gems.

BACKGROUND INFORMATION

Sarah's strategy is an *algorithm* which is a sequence of steps that solves a problem. In this task, we evaluate the algorithm to find the maximum number of gems for which it works.

There are lots of algorithms that work well for some situations but not for others. For example, a sorting algorithm might be able to sort positive integers but not negative integers. Some *route finding algorithms* can find the shortest route between two towns, but do so too slowly when applied to a map with thousands of linked towns. In this case quicker algorithms are often chosen that find one of the shortest routes but not necessarily the shortest - a good solution obtained quickly is better than no solution!

In general, there are many ways of evaluating algorithms programmed to run on a computer. For example, we often consider the *running time o*f an algorithm (how long we expect it to take from start to finish) or the *amount of space* an algorithm uses which indicates how much computer memory it might use while it is executing.

This task is also about an algorithm that is constrained by some condition. In our case, Sarah is allowed to ask only three questions and every question can only cover 4 items. Despite this constraint, this algorithm works well for collection sizes smaller than 11, but it fails otherwise. There might be different reasons to impose constraints on algorithms. For example, one could demand that an operation must be completed in a fixed amount of time, which is necessary in real time operating systems. Another reason could be that operations have external costs, or they damage a component that is handled in a manufacturing process.

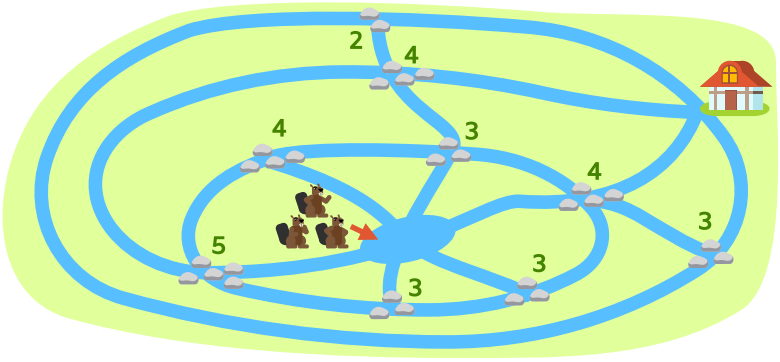
It may not be considered a problem that the algorithm fails a certain threshold, but it is the responsibility of the engineers to make sure that these conditions never occur. For example, the constrained strategy of this task must never be used on collections greater than 10.

This question comes from Switzerland

Years 3+4
Years 5+6
Years 7+8
Years 9+10 Hard
Years 11+12

Collecting stones

The Barker Beaver's family lodge consists of 21 canals. 31 stones need to be removed. The beavers start in the central pool and go for a swim to collect the stones, bringing them to the storage deposit on the right.



The stones are heavy. A beaver can only carry one or two of them at the same time, but no more. To get from one intersection to the following, it takes each beaver exactly one hour to swim. After starting simultaneously the beavers have to collect all stones within four hours. They also only want to recruit the minimum number of beavers they need to do the task.

Question

What is the minimum number of beavers the Barkers need to recruit to clear the canal within 4 hours?

12 beavers | 14 beavers | 16 beavers | 20 beavers | 22 beavers | 24 beavers

EXPLANATION

Answer

The correct answer is 14 beavers

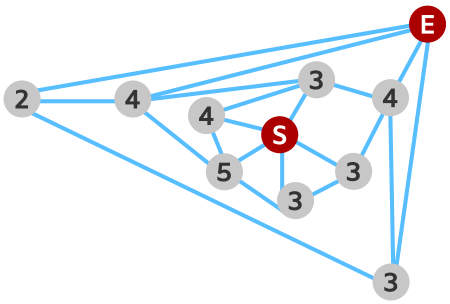
Explanation

There is only one path where the beavers can reach the deposit from the central pool within 2 hours. On this path are 4 stones. 2 beavers can carry these 4 stones to the deposit and may continue to carry 4 additional stones to the deposit.

Then there are 23 = 31 - 8 stones left, and at least 12 more beavers are necessary. The remaining stones lie on paths which consist of 3-4 intersections. Within 4 hours, no time is left after depositing stones to continue to collect more.

We can observe by inspecting the beaver lodge that the remaining 23 stones can be brought to the deposit by 12 beavers. 11 of them carry 2 stones and 1 of them carries just one stone.

Another way of thinking about this problem is that there are 31 stones, and only two beavers can bring 2 stones each and then return for more. The other beavers can bring at most 2. So at least 13.5 beavers are necessary. Therefore the best solution requires 14 beavers.



A more detailed explanation is shown left:

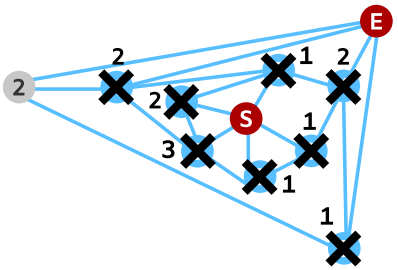
At the start, all the stones and the deposit of the beavers lie in canal intersections. In the following picture, we represent canals using straight lines. The numbers in the pictures represent the number of stones on the respective intersection.

Continued on next page

Collecting stones – continued

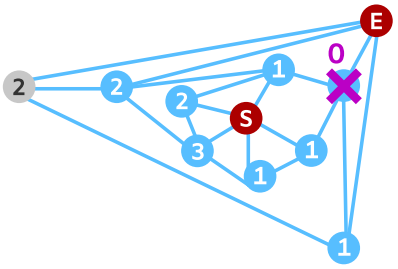
We measure the path length between the central pool and the deposit by counting the number of canals the beavers have to swim through:

- There are no canals that directly connect the starting point and the deposit (i.e., there are no paths of length 1 hour).
- There are paths of 2 hours, of 3 hours, and of 4 hours in length.
- The beavers are required to collect stones at most for four hours. Paths of more than four hours in length do not need to be further considered.

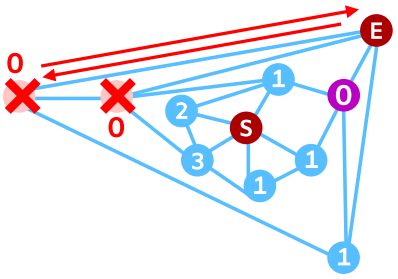


First, we consider all the intersections with at least three stones. In each such intersection, a beaver collects two stones and brings them to the deposit. To this end, we need 8 beavers. They collect 16 stones. 31-16=15 stones are still left. All the beavers except one were at least three hours on the way. They are not able to go back to the canals and to collect further stones. We can't "reuse" them.

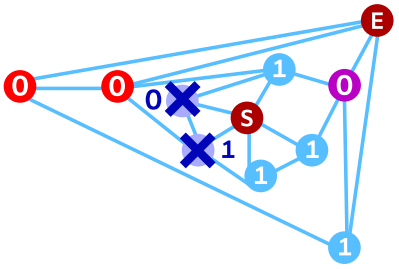
One of the beavers, however, collected two stones within two hours. They would have enough time to get back to an intersection, to collect two additional stones, and to bring them to the deposit. We can therefore "reuse" this beaver at a later point.



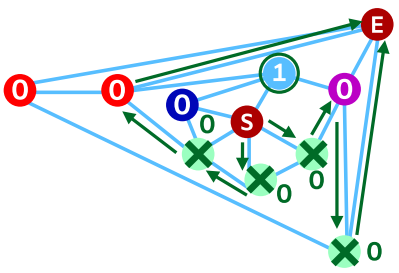
A new beaver collects the two stones within two hours. We can "reuse" this beaver as well. 15-2=13 stones are still left. So far, we have used 9 beavers. Two of them can be "reused".



Now, the two beavers can be "reused". Each of them collects two stones. 13-4=9 stones are still left. So far, we used 9 beavers. No one of them can be further "reused".



We apply the same strategy and further consider all the intersection with at least two stones. We need 2 beavers. They collect 4 stones. 9-4=5 stones are still left. So far, we have used 9+2=11 beavers.



The final step consists in finding a solution (possibly by trial and error) that uses the fewest number of beavers as possible. Two beavers collect two stones each. A third and last beaver collects the last stone. The best solution is A): 11+3= 14 beavers.

Continued on next page

This question comes from Switzerland

Years 3+4
Years 5+6
Years 7+8
Years 9+10 Hard
Years 11+12

Collecting stones – continued

BACKGROUND INFORMATION

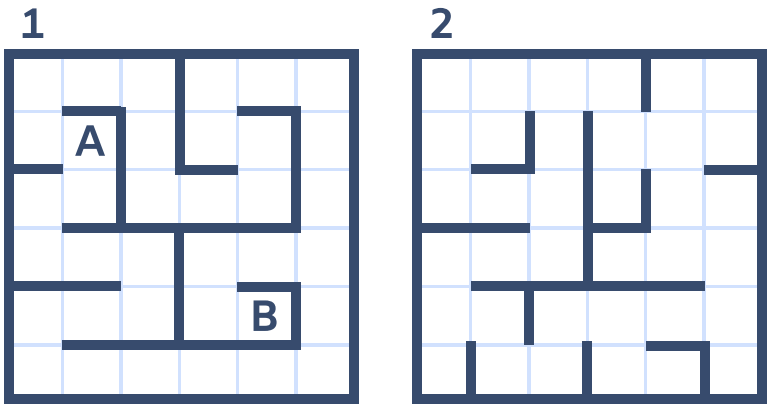
Computer scientists call the image representation of this problem a *graph*. A graph consists of *nodes*, and *edges* between the nodes. The distance between two nodes can be expressed in several ways, for instance by counting the number of edges we have to traverse to get from one node to another one. Graphs are one of the core problem-solving instruments in the life of a computer scientist. Navigation systems are a well-known example of applications that heavily make use of graphs.

In our task, the beavers are expected to collect all 31 stones. Possibly, there is more than one way to reach this goal. One could imagine to use 14, 18, 20 or even 31 different beavers. However, the question explicitly requires a solution that uses as few beavers as possible. From this perspective, we should carefully consider whether there is a way to "reuse" a beaver or not.

Problems that look at the best possible solution are very common. In the terminology of informatics, they are *optimisation problems*. A navigation system is expected to select the best (e.g., the optimal) way to get from A to B. Of course, the exact meaning of "the best" solution needs to be defined. In the case of a navigation system, "the best" solution could mean minimising the distance in km, or it could mean the time the car needs to reach the target. It could also mean either of these, but additionally avoiding certain highways or toll roads.

Maze

Beaver Sonia is in a maze. The maze is made up of two floors, each with its own grid of obstacles.



Sonia can move between two adjacent cells on the same floor if there is no wall between the cells; this takes **1 second.**

Sonia can also use her magic wand to move to the corresponding cell of the other floor; this takes **5 seconds.**

For example, if Sonia is in cell A, there are three possible moves:

1. Move left. This move takes 1 second.
2. Move down. This move takes 1 second.
3. Move to the corresponding cell of the other floor. This move takes 5 seconds.

Sonia starts at cell A and wants to reach cell B as soon as possible.

Question

What is the shortest time needed for Sonia to reach cell B, if starting from cell A?

16 17 18 20

EXPLANATION

Answer

18

Continued on next page

Maze – continued

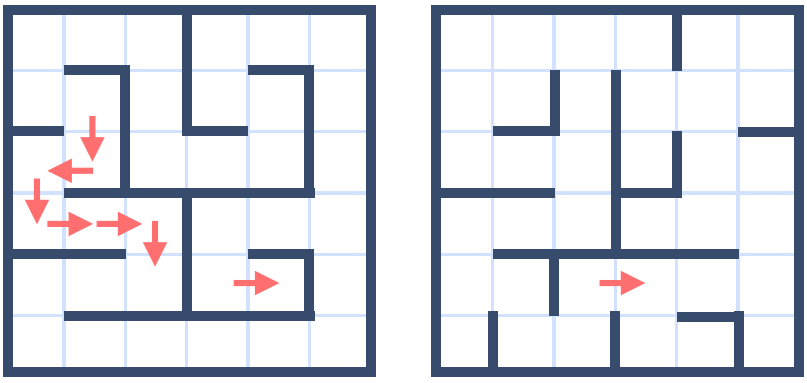
Explanation

The given problem is a shortest path problem. There are different approaches to obtain the solution, one of which is shown here. The image below shows the lengths of the optimal paths to all cells if starting from A.

2	3	4	11	12	13
1	0	5	10	9	14
2	1	6	7	8	15
3	4	5	18	17	16
8	7	6	17	18	15
9	10	11	12	13	14

7	6	7	8	11	12
6	5	8	9	10	11
7	6	7	10	11	12
8	9	8	13	12	13
9	10	11	12	13	14
10	11	12	13	14	15

One can see that the length of the shortest path to B is 18.
One of the possible optimal paths is the following:



Answer D (20) corresponds to the optimal path when moving only within one floor. Answer A (16) corresponds to the lower bound of the time to reach B from A if going via the other floor and assuming there are no walls (i. e. [Manhattan distance from A to B] + [time to move between floors]).

BACKGROUND INFORMATION

The shortest path problem is a well-studied graph theory problem. Its applications include finding an optimal path in a computer network, on a city map, etc.

One common algorithm for solving these sorts of problems is *Dijkstra's algorithm*, which looks at finding the shortest path between nodes by visiting each 'unvisited' node and updating the optimal path.

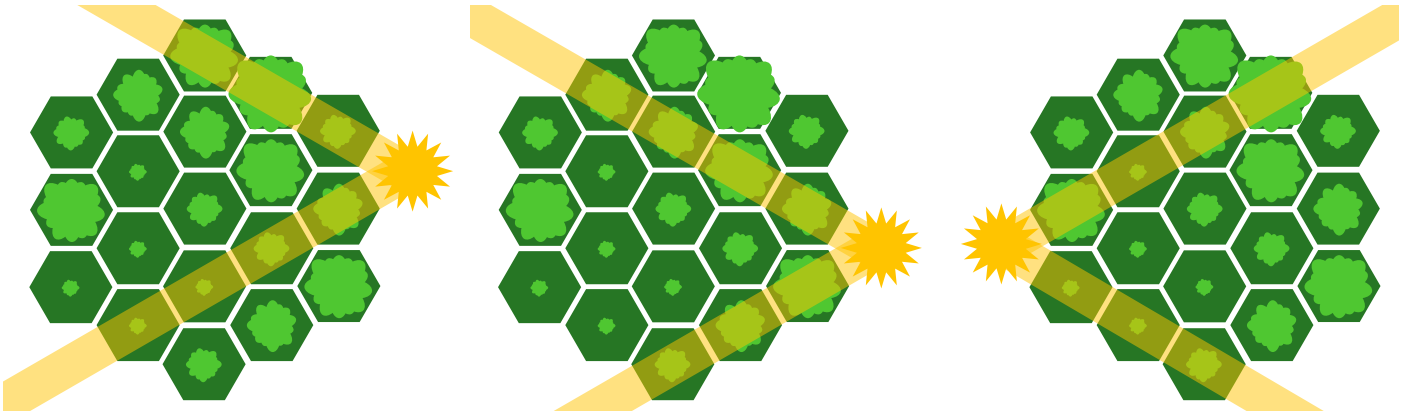
The instance of the shortest path problem on a grid graph has various applications, e.g. wire routing in *very-large-scale integration* (VLSI) when creating an integrated circuit (the so-called "maze runner"). In the given problem, moving between floors takes more time compared to moving within one floor (5 to 1). This corresponds to the fact that in multi-layer VLSI circuits, cross-level wiring is more expensive than intra-layer wiring.

Common VLSI circuits are placed on a 10-layer silicon crystal, while this task introduces only a 2-layer problem. Moreover, VLSI design requires you to route millions and billions of wires to connect transistors, inputs, outputs and memory cells together to get a VLSI circuit for digital devices.

Tree farming

Beaver Dana is surveying the local forests. She has created a map of the trees in the forest on a hexagonal grid so she can work out how much sunlight a given tree will get during the day. The sun moves in the following way:

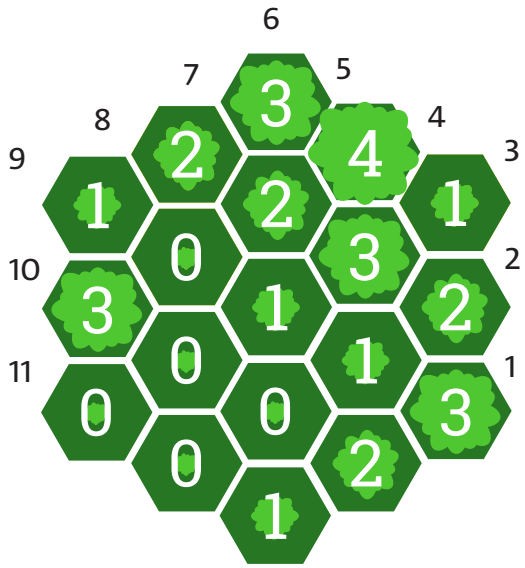
- The sun starts every morning in the position shown in the left image below.
- Every hour, it moves counter-clockwise around the hex grid by one side. One hour after sunrise it is located in the position shown in the middle image.
- It continues moving one side every hour until sunset, when it is located in the position shown in the right image.



Sunrise **1 hour after sunrise** **Sunset**

The sun shines in a straight line along any sides it is touching, as shown above.

The map Dana creates looks like this:



Continued on next page

This question comes from Australia

Tree farming – continued

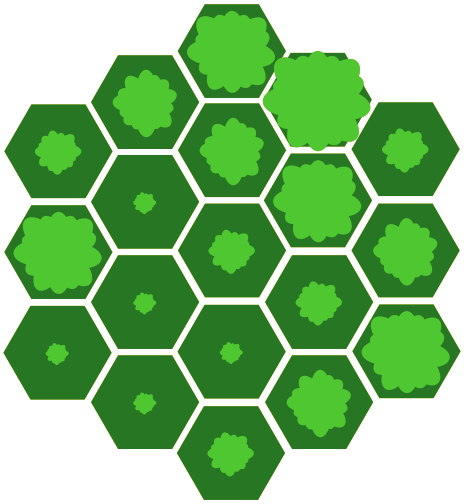
The following rules are true:

- The height of a tree is given by a number.
- A tree will block the sun from reaching a number of hexes behind it equal to its height. (A tree of height 2 will block the sun from trees in the 2 hexes behind it.)
- However, a tree will not block the sun from reaching a taller tree. (A tree of height 2 will not block the sun of a tree of height 3, even if it is within 2 hexes.)
- There are 11 hours of sunlight every day, including sunrise and sunset.
- The position of the sun at each hour is shown by the numbers on the hexagons' borders.

A tree needs to get at least 3 hours worth of sunlight every day so that it can grow.

Question

Which trees will get enough sunlight to grow in a given day?



EXPLANATION

Answer

This question comes from Australia

Years 3+4
Years 5+6
Years 7+8
Years 9+10 Hard
Years 11+12

Tree farming – continued

Explanation

For this question, any given tree on the map will need to get 3 hours of sunlight to grow. To determine whether a tree will get 3 hours of light, we need to establish which angles the tree will be able to get sunlight from.

One way to do this would be to count each tree every hour, but this can be time consuming.

Given that the sun only travels in an arc along the right, top, and left sides of the map, we only need to consider the sides of the hexagons that will receive sunlight from those directions.

By looking at each tree, if we can establish for each of those lines that at least 3 have unbroken lines to the sun, we can be certain that this tree will get the 3 hours needed to grow.

Further, for all of the hexagons below the centre hex, there are only three possible directions that they have a line to the sun at any point during the day. If any of these three lines are broken at all, we know that the tree will not grow.

BACKGROUND INFORMATION

This task shows a basic method of *modelling* a complex system, specifically the amount of light shining into a forest. Computer scientists often use *algorithms* to create models of real-world situations, so that we can better understand the world around us. These models often cannot be exactly perfect, but are intended to be as close an approximation as possible.

We can then use these models to test hypothetical scenarios, which is called simulation. Examples of this might include modelling weather, or creating a model of the terrain of a city, then using that to examine how large amounts of rainfall will impact the waterways in that city. This helps to answer questions like “will it flood?”, and “will it drain away through safe channels?”

When creating models, computer scientists often have to determine which aspects of the real-world situation they can simplify, and which need to be as detailed as possible. In this example, the sun operates in a very simple way, compared to the way the sun functions in real life. This has been simplified in order to make the model easier to create and use.

Bebras Challenge 2023 Round 1

Years 11+12

This question comes from Austria

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Easy

Lists

We can represent a list of numbers 3, 5, 2, 4, 1 visually as follows:

	1	2	3	4	5
X	3	5	2	4	1

Letters are used to represent the rows, and the red numbers are used to represent the columns. We can write (X 2) to describe the number in column 2 of row X. So, for example, (X 2) is 5. Similarly (X 5) is 1.

The positions of numbers in the list can be represented indirectly, for example:

(X (X 3)) is 5 because (X 3) is 2, so (X (X 3)) = (X 2) = 5.

Here are three lists, A, B and C.

A	3	2	4	1	5
B	5	4	1	3	2
C	2	5	4	3	1

Question

What is the number described by (A (B (C 3)))?

1 2 3 4 5

EXPLANATION

Answer

4.

Explanation

A	3	2	4	1	5
B	5	4	1	3	2
C	2	5	4	3	1

By solving the innermost brackets first, we find that:

(C 3) = 4,

so (B (C 3)) = (B 4) = 3,

so (A (B (C 3)) = (A (B 4)) = (A 3) = 4.

This can be shown by substituting these values working from the inside out:

(A (B (C 3))) → (A (B (4)))→ (A (3))→ (4)

BACKGROUND INFORMATION

Data structures are essential for programming. Those that can hold an entire list of data are especially useful. Data structures, like many other entities in computer science and programming, can be interconnected, so an element of one list can describe a position in another list. This indirect way of describing positions is a powerful concept. It can look confusing at first, but it is not difficult to work through the representations by computing the values step-by-step.

Listen and walk

Tina is vision impaired, and uses her talking glasses to navigate through the city.

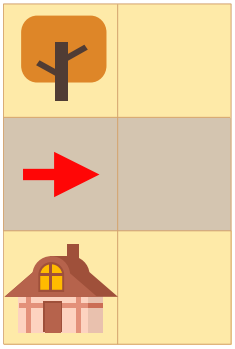
The talking glasses have cameras and an intelligent object recognition system.

The glasses can recognise: a house, a tree, a road and lawn.

When Tina enters a new road square, the talking glasses explains her surroundings in the following order:

1. What is on **her left.**
2. What is in **front of her.**
3. What is on **her right.**

For example, in the image below the glasses would say: "tree, road, house".



Tina starts at the arrow, facing in the direction of the arrow, and listens to her talking glasses. After the glasses finish talking, Tina moves one step forward, backward, left, or right.

This is what they tell her (beginning with the starting square):

- tree, road, house
- road, road, lawn
- tree, road, tree
- road, road, road
- tree, road, tree
- tree, house, road
- road, road, tree
- house, road, tree

At the end, Tina has arrived at one of the squares labelled with a letter.

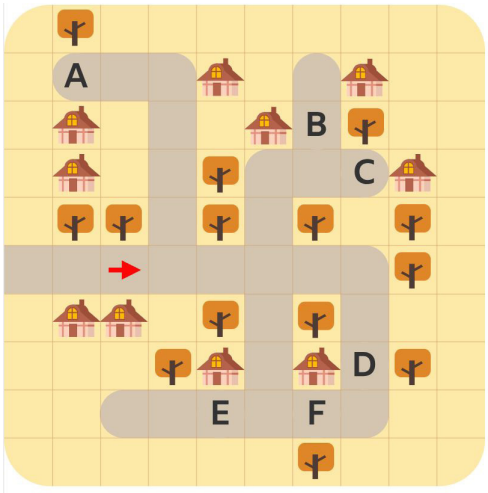
Question

At which letter on the map has Tina arrived?

EXPLANATION

Answer

B



Continued on next page

Listen and walk - continued

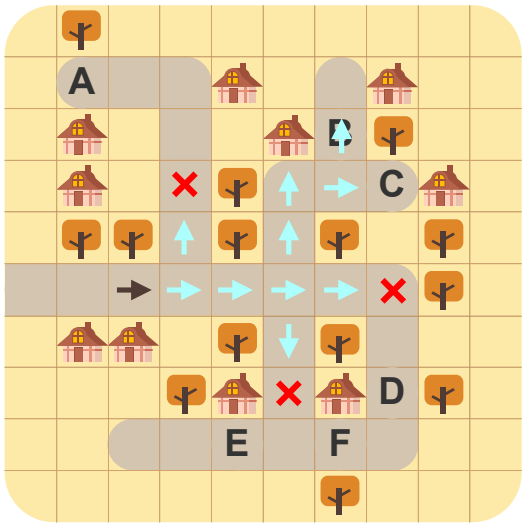
Explanation

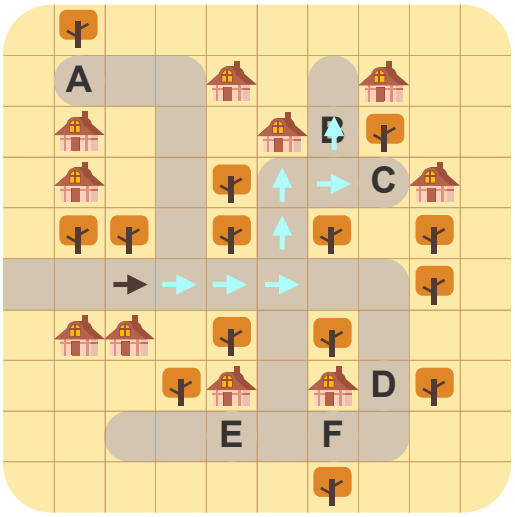
There are different strategies to find the solution.

You can just follow the path Tina takes by following the instructions given by the talking glasses. The objects do not tell us when Tina makes a turn on the road. So, if at any location the next set of objects match objects on more than one road, we try each road at a time.

For example, if we take the road to destination A, to continue along the road, we would realise that after "tree road tree" the next (4th) set of objects should have been "lawn road tree". However the objects are "road road road" which do not match the objects on road to destination A. So we trace back to the intersection where we chose the road to destination A and take another road.

We do this until we have either tried all the possible roads or until we reach one of the targets. The figure below shows possible paths Tina can take. X indicates that she could not have continued on the path.





Alternatively, you can check the six target squares. Only B and D match the description "house road tree". But both squares adjacent to D do not match "road road tree", which is the second last description. Therefore, B must be Tina's goal square.

BACKGROUND INFORMATION

This task highlights the important topic of *personal information*. It is shown here that just knowing a log of the messages the talking glasses gave to Tina is sufficient to recover her final square among several possibilities. In general one could think this log is not informative enough to say where Tina is in this moment, but the additional knowledge about the environment in which she acts is rich, and sufficient to track down her destination. It is often difficult to judge how sensitive personal information can be, and this is especially true for spacial or geographical data.

Object recognition is an important topic in computer science. An artificial intelligence system can be trained to recognise objects like houses, trees and road surfaces. For example, smartphone cameras are able to recognise faces and indicate them with a frame. An autonomous car must recognise persons, other cars, traffic signs and other objects in its environment, which are important for its decisions where to go. To help blind and visually impaired persons, modern web media include *Automatic Alt Text* (AAT). This technology uses object recognition to automatically create verbal descriptions (alt texts) of images. The "talking glasses" are still science fiction, but there exist real projects that try to implement this idea.

This question comes from Ireland

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Easy

Colourful candles

Sarah has candles in the shape of the numerals 0 to 9. There are two of each numeral.

The candles come in three colours: orange, red, and blue.

All 0-shaped candles are orange, all 1-shaped candles are red, and so on (see image). Each year for her birthday, Sarah places candles on her cake to spell out her new age.

Today is Sarah's 11th birthday and because both candles are the same colour her family gives her an extra birthday present. She must wait three years until she is 14 before both of her candles will have the same colour again. Then there is a three-year wait until she is 17, and a further five year wait until she is 22.

Question

If Sarah uses this system from today until she is 99 years old, what is the maximum number of years she has to wait between any two birthdays where two candles of the same colour are used to spell out her age?

5 6 7 8

EXPLANATION

Answer

5 years

Explanation

Let XY be the current age of a person where X and Y are two numerals from 0 to 9 representing the age, and X and Y are the same colour according to the table in the task body. Suppose Y is one of 0, 1, 2, 3, 4, 5, or 6. Then you must wait three years until the age X(Y+3) to see two candles of the same colour.

Next, let's look separately at the three cases where Y is 7, 8, or 9:

• If Y is 7, then X is also one of the reds (1, 4, 7) and you must wait 5 years until the age (X+1)2 for two candles of the same colour.

• If Y is 8 then X is one of the blues (2, 5, 8) and you must wait 2 years until the age (X+1)0 to see two candles of the same colour.

• If Y is 9, then X is one of the oranges (0, 3, 6, 9) and you must wait 2 years until the age (X+1)1 to see

BACKGROUND INFORMATION

Quite often in computer science, we are told the rules of how a *sequence* is formed and must simulate the sequence until certain conditions are met. In this task, the sequence of two-digit decimal numbers is mapped to a sequence of pairs of colours. The *n*th element of this abstract sequence is the colour of the two candles used to spell out (represent) the number *n*.

Movie night

Ms. Naya is organising a movie night for her seven students. She will be using the school's TV room.

To avoid seating conflicts, Ms. Naya sets specific seating rules:

- Birthday rule: each student adds together their birth day (DD) and birth month (MM). This is noted as (DD/MM) below their name.

Ms. Naya works out how many 7s can fit into the sum, and uses the whole number that is left over to assign their seat.

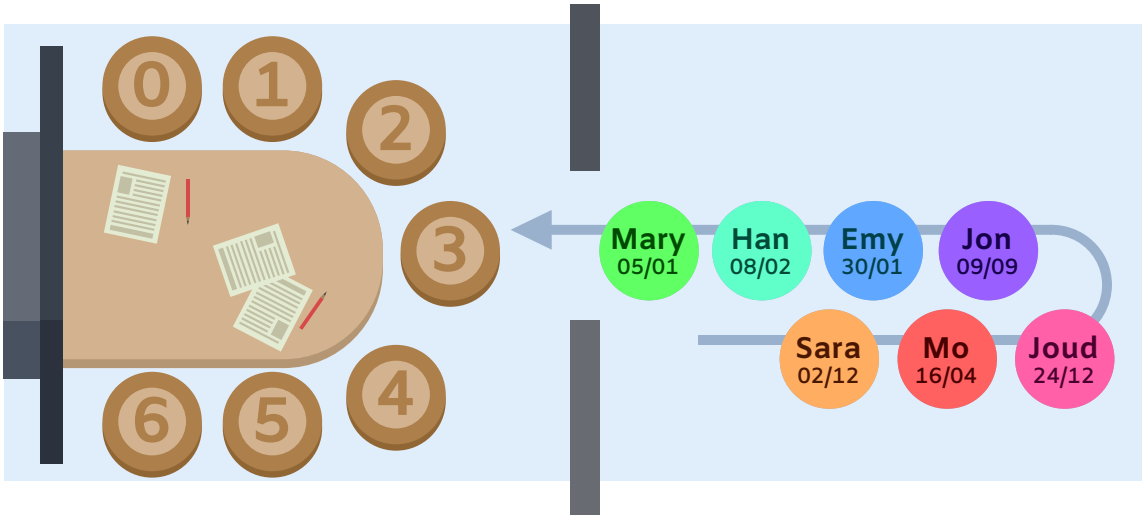
For example, Joud is assigned chair number 1 because when we add 24 + 12, we get 36, and the number 7 can fit into 36 five times with 1 leftover.

- Collision rule: if one student's chair is already used by another student, they should move to the next chair.

For example, if chair 3 is taken, the student moves to chair 4. If both chairs 4 and 5 are taken, then the student moves to chair 6. If chair 6 is taken, then the student moves to chair number 0.

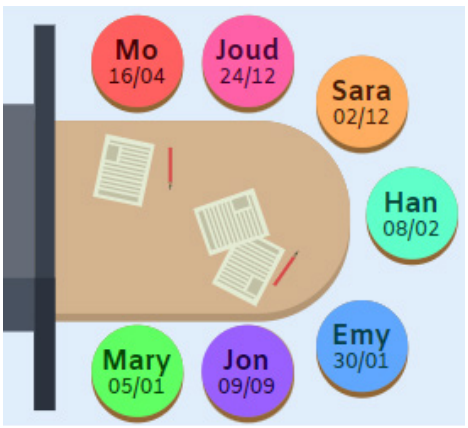
Question

The students enter the room in the order seen below. Match their names to their correct seats.



EXPLANATION

Answer

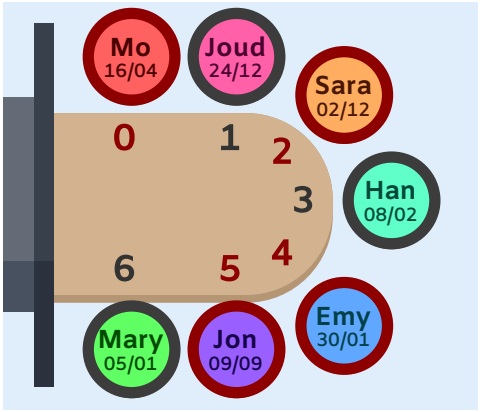


Continued on next page

Movie night - continued

Explanation

When applying the seating rules, the students will end up using chairs as represented in the following seating map:



The black borders mean the student is using the chair that matches their given number. The red borders means that the student moved at least one seat before they found their final seat.

We arrive at the seating map by using the following instructions:

First, calculate the sum of the birth day value and the birth month value. Then we calculate the result by dividing the sum by 7, and identifying the whole number which remains (modulo 7). This gives us the expected seat for each student.

Student	Joud	Mary	Han	Mo	Jon	Sara	Emy
Birth (DD/MM)	24/12	05/01	08/02	16/04	09/09	02/12	30/01
Sum (DD + MM)	24+12=36	05+01=6	08+02=10	16+04=20	09+09=18	02+12=14	30+01=31
Modulo 7	1	6	3	6	4	0	3

The table shows that Mary and Mo both are expected to sit in chair 6. However, only one of them may sit in chair 6, and the other must sit in a different chair. The same is true of Han and Emy, who both are given number 3. The final result is based on the order in which they enter the room.

BACKGROUND INFORMATION

This method of evaluating and prioritising is also used when computers save files to a disk. Computers use file names to find a place to put our files. As a way to prevent files with the same data from being saved in the same place, your computer may prompt you to save them in a different place, change their file name, or change their file name itself. We call this method *hashing* and prevent it by using a *collision avoidance* method.

Basically, it allows quicker access to information once it is placed in the data structure.

This question is an example of *modular arithmetic*. Modular arithmetic relates to numbers and cycles. One of the ways we use it daily, is when we think about 12 hour time on a clock. Programmers can also use modular arithmetic to track time, and loops of code.

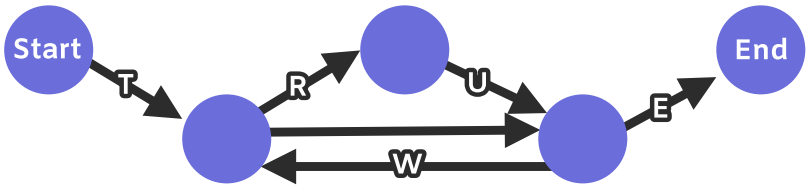
This question comes from the United States

Words

Bridgit is creating random "words" using a specific diagram that she has found.

She always starts in the circle labelled 'Start' and follows the arrows until she ends up in the circle with the label 'End'.

Every letter that she passes while doing this, she writes down.



Question

How many different words of 8 letters can Bridgit create?

9 10 12 14

EXPLANATION

Answer

The correct answer is 9.

Explanation

Note that Bridgit always has to use 'T' at the beginning, and 'E' at the end. So we're looking for all the six letter words with 'R', 'U' and 'W'.

Let's turn the problem into a smaller problem: how many different ways are there to create words of a specific length:

- length 1: 1 (W; follow the empty line, follow the W, follow the empty line)
- length 2: 2 (WW and RU; follow the R, follow the U)
- length 3: 3 (WWW; WRU; RUW)
- length 4: 4 (WWWW; WWRU; RUWW; WRUW)
- length 5: 6 (WWWWW; WWWRU; WWRUW; WRUWW; RUWWW; RUWRU)
- length 6: 9 (WWWWWW, WWWWRU, WWWRUW, WWRUWW, WRUWWW, RUWWWW, WRUWRU, RUWWRU, RUWRUW)

Important patterns about the words (excluding the 'T' at the beginning and 'E' at the end) include:

the letters 'R' and 'U' must always appear as a pair.

two 'RU' pairs cannot appear consecutively.

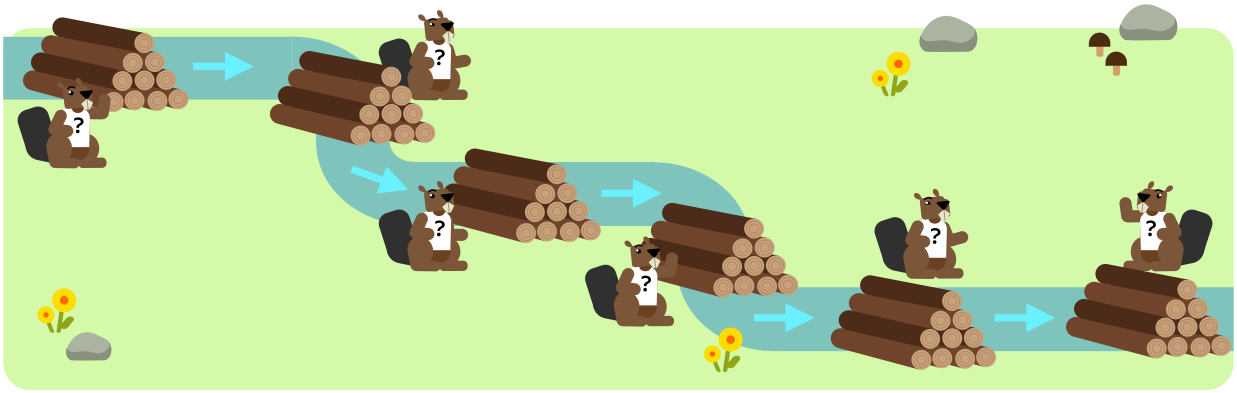
BACKGROUND INFORMATION

This question demonstrates the usefulness of an algorithmic "machine", or *routine*, that generates a sequence of letters with certain constraints. Searching through all possibilities is what a computer is really good at. In this question the task asks us to make a conscious effort of writing down all words, but we have to remember to not count words twice.

These sort of systems are sometimes used for generating passwords for students competing in Bebras, or to produce number plates for cars. In these cases unique sequences of characters are required but they must follow a specified format.

Beaver dam

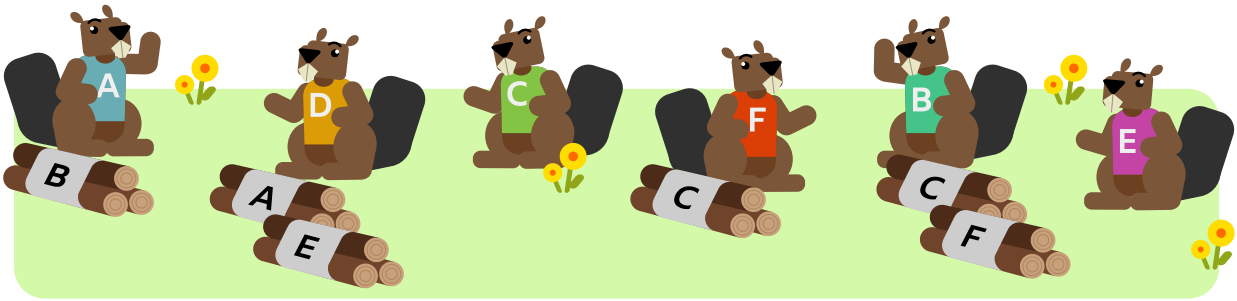
Six beavers (A, B, C, D, E, and F) have each built a dam along a river.



One day a storm washes some of the pieces of wood from the dams downstream (in the direction of the arrows).

Thankfully, all the pieces of wood are marked. For example, the pieces of wood from the dam built by Beaver A have an "A" on them.

After the storm, all the beavers got together to swap back the pieces of wood. Each beaver laid out the wood in front of them that arrived at their dam from upstream, as shown here:



Question

Judging by the pieces of wood each beaver picked up, what is the order of the dams from upstream to downstream?

A → B → C → D → E → F

C → B → F → A → D → E

C → F → B → D → A → E

E → C → F → B → A → D

Continued on next page

Beaver dam – cont'd

EXPLANATION

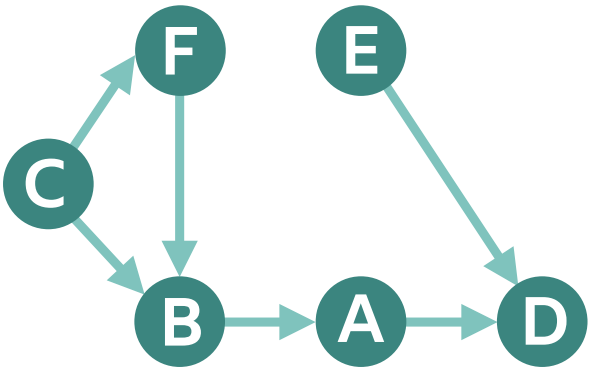
Answer

E → C → F → B → A → D.

Explanation

Since the pieces of wood drifted from upstream to downstream, if beaver A found beaver B's wood, we can assume beaver B is higher up the stream than beaver A.

To determine each beaver's position, it helps if we map out the relative position between two beavers with arrows to form a graph:



- As we can see, there is no arrow pointing to C or E, so only C and E can be the owner of the first dam. Therefore, option A is incorrect.
- Option B is wrong, because F should be higher up the stream than B, judging by the arrow pointing from F to B in the above graph.
- Option C is wrong, because D should be higher up the stream than E, judging by the arrow pointing from C to B to A and then to D in the above graph.
- Only the order of option D fits the graph above; therefore, option D is the correct answer.

BACKGROUND INFORMATION

A *directed acyclic graph* (DAG) consists of vertices and directed edges without any cycle. In this task, we can see each beaver as a vertex. The direction of each piece of wood drifted is a directed edge. Lining up beavers into the correct order from upstream to downstream is called *topological sort*; that is, to sort the vertices according to the direction of the edges.

There might be various order arrangements that fit the topological order. One of the ways to find a topological order is using *Kahn's algorithm*. First, select a vertex which has no incoming edges. Delete the vertex and all the edges outgoing from it. Repeat the previous process until the graph is empty.

In computer science, topological sort is used for processing scheduling, data serialisation, and resolving symbol dependencies in linkers.

This question comes from
Czechia

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Medium

Jumping game

Verunka loves jumping. She has found 17 paving stones in a line and created a game that uses them.

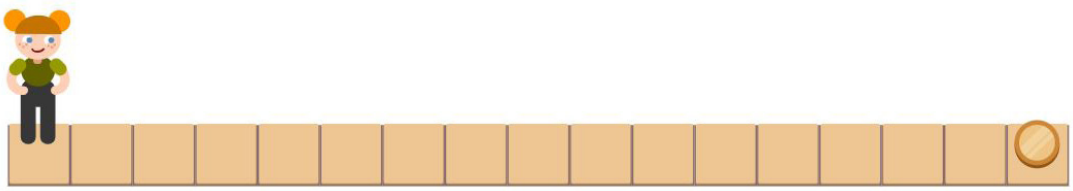
Verunka puts a coin at one end of a line of pavers and then stands on the opposite end, facing the coin.

She has two jumping rules:

- If you are standing on a paving stone marked “X”, jump 3 paving stones forward.
- If you are standing on a paving stone marked “O”, jump 1 paving stone backward.

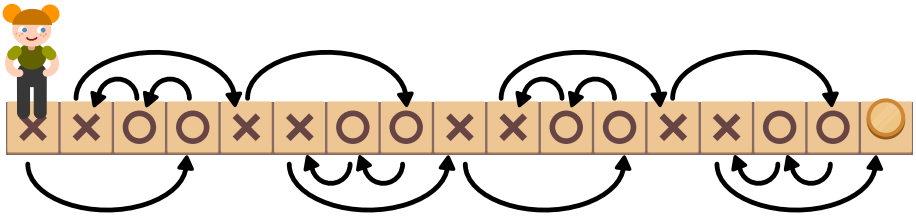
Question

Verunka wants to jump on **ALL** the paving stones, and still arrive at the coin. In what way should she mark the pavers to achieve this?.



EXPLANATION

Answer



Explanation

The arrows show how to jump. Because there is only one type of forward movement and one type of reverse movement, the task has only one solution. If Verunka jumped forward and left some empty tiles behind, she would not be able to return to them.

BACKGROUND INFORMATION

Finding a way of using consecutive actions to achieve a goal, according to given rules, is called *algorithmisation*. A robot or a computer application can be programmed so that it works according to given rules. If we then need a specific output from it, we must set a suitable input.

For example, if we have a long narrow corridor in the shape of an L and our cleaning robot can only go straight, turn to the right by 90°, and then go forward, we have to put it on the end of the shorter part of the “L” otherwise it couldn’t clean the whole corridor.

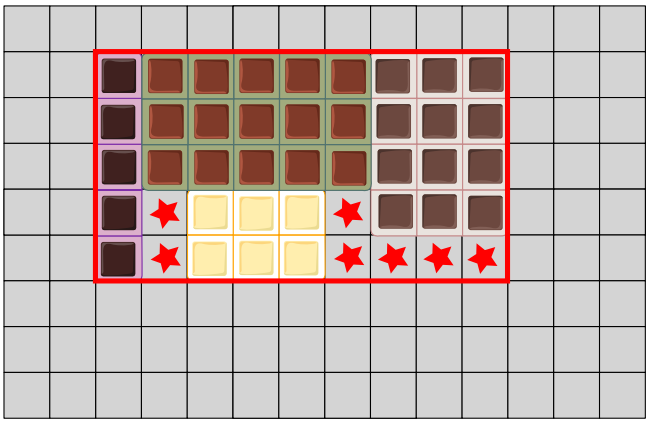
While this task only has one solution based on the specific rules outlined, many tasks often have multiple different solutions. Evaluating which solution to choose for a particular problem is an important skill in computer science.

Packing

A rectangular gift box is needed to hold 4 different sized chocolate bars. The biggest bar has 15 pieces.

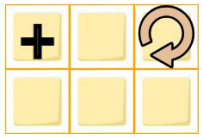
Chocolate bars are to be arranged in one layer, and the gift box should have as few gaps as possible.

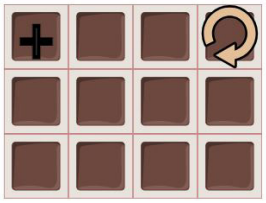
For example, if the chocolate bars are arranged as shown on the right, they would be contained in the rectangular gift box outlined in red, and there would be 7 gaps (marked by red stars).

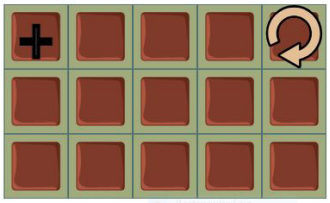


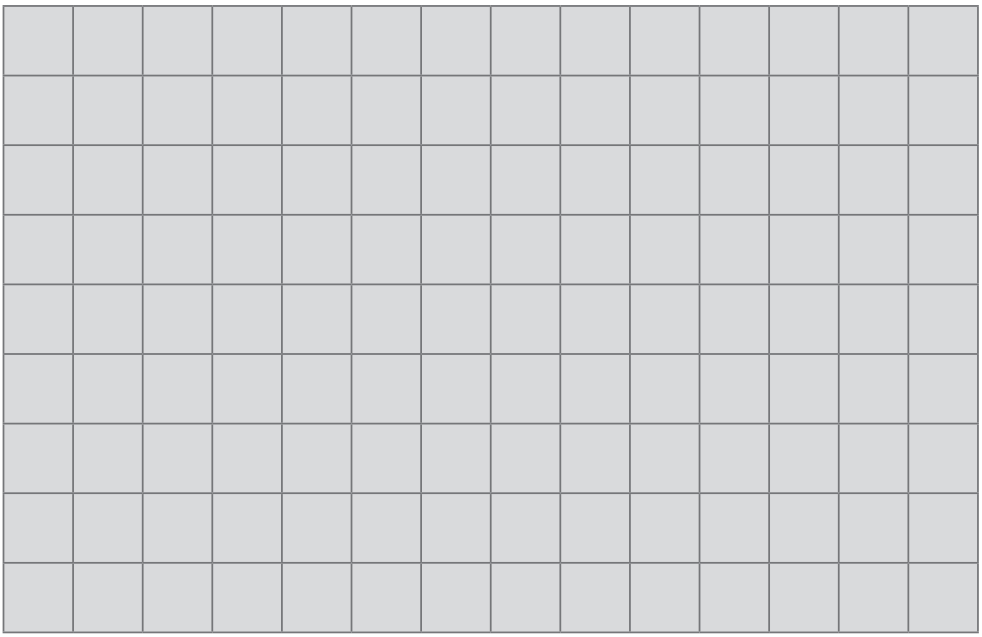
Question

Arrange the chocolate bars in to fit in a rectangular gift box, so there are as few gaps as possible.









Continued on next page

Packing – continued

EXPLANATION

Answer

The best possible arrangements leave only 2 gaps.

Explanation

One possible arrangement:

This leaves us with 2 gaps in the top left of the rectangle.

The number of chocolates pieces in all 4 bars put together is 12 + 15 + 6 + 5 = 38. A gift box that holds 38 pieces with 0 gaps must have dimensions 1 x 38 or 2 x 19. You will never be able to fit the 3 x 5 chocolate bars (or the 3 x 4 chocolate bar) in this gift box.

A gift box with 1 gap would hold 39 chocolate pieces. There are two possibilities, 1 x 39 (not possible) or 3 x 13. The two largest chocolate bars would take up 9 rows within such a gift box. The remaining 4 rows are not enough to place the smallest chocolate bar of size 1 x 5.

Hence, 2 gaps is the minimum we can have in any gift box, and we can achieve this as shown above.

BACKGROUND INFORMATION

This task is an example of *bin packing*. Bin packing is an important practical problem - for instance, loading as many boxes as possible onto a truck, or marking out parking places to allow as many cars as possible to park.

This is an example of an *optimization problem* where we want to find an algorithm that will quickly find the best solution. Unfortunately, it can be very difficult to find the best solution for bin packing when the number of boxes is large, even if we use very powerful computers. Instead, we can use algorithms that provide only approximate solutions, but are good enough in practical situations

This question comes from the Netherlands

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Medium

Floor pattern

A robot can move around a 10 x 10 grey grid and paint black or white squares.

The robot can only move up, down, left, or right, and one square at a time.

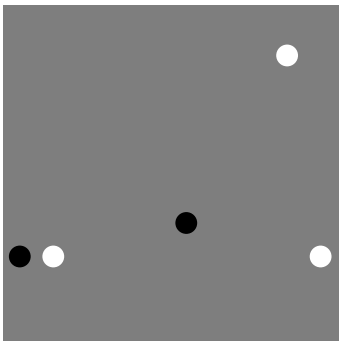
An experiment to produce some interesting patterns was performed. Each experiment followed some common rules:

- Some black dots and white dots are placed randomly on the grid.
- The robot is programmed to move from left to right across the top row, then right to left across the second row, then left to right across the third row, and so on. This way it snakes across all 100 cells of the grid.

Four different programs were run to see what patterns can be made by the robot.

Programs:

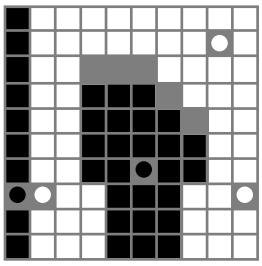
A. The cells are painted the colour of the **nearest dot** to the robot. If there is a tie, no colour is painted.
B. The cells are painted the colour of the **furthest dot** from the robot. If there is a tie, no colour is painted.
C. The cells are painted the colour of the **last dot it passes over**.
D. The robot looks at all the cells it could go to in two moves or less, from its current position. It paints this cell the **colour of the most common colour dot it "sees" in these cells.** If there is a tie, or there are no dots in these cells, no colour is painted.



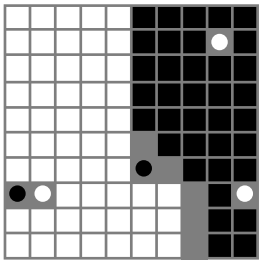
Question

Given the positions of the randomly placed dots, shown above, draw a line to match the patterns to the programs the robot followed.

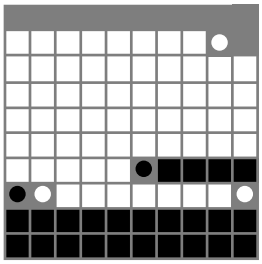
A



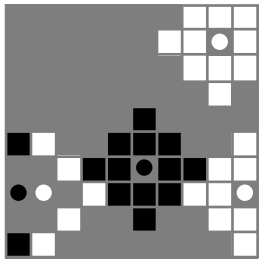
B



C



D

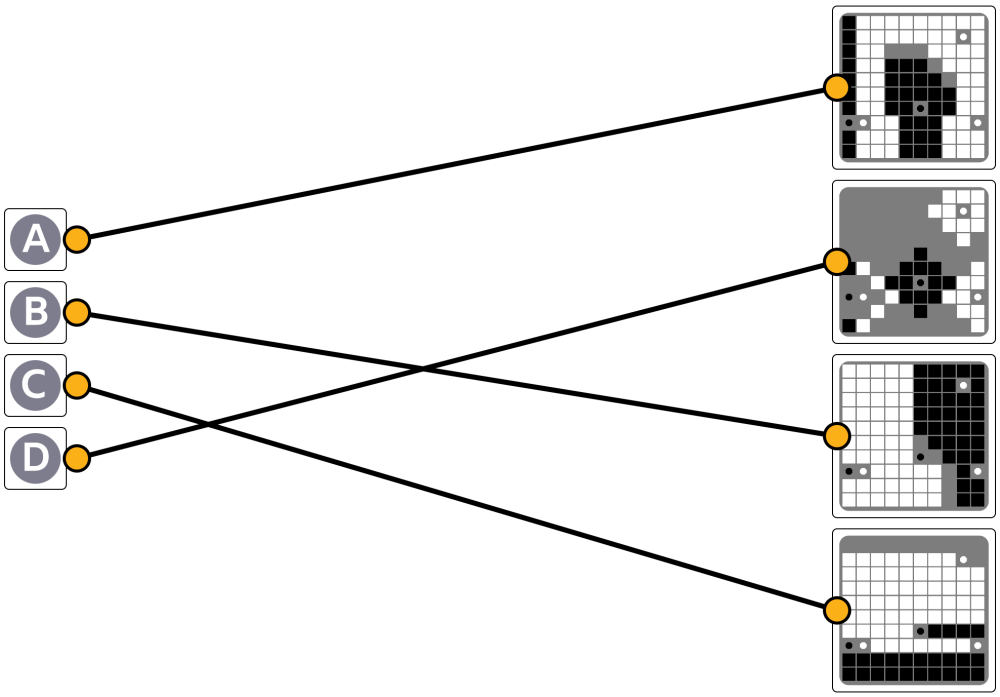


Continued on next page

Floor pattern – cont'd

EXPLANATION

Answer



Explanation

There are various ways of solving this task. The most laborious would be to to step through every cell and decide what colour they should be. Then compare the patterns with those given.

A speedier method would be to identify the easiest programs to follow. For example, it is quick to identify the pattern produced by program C, as the grid should change colours in the middle of a row each time it passes a dot. Only the bottom pattern matches this.

The three remaining patterns all contain a different coloured top left cell. The three remaining programs only need to be applied to this top cell to determine which pattern belongs to which program - a much speedier approach than solving the entire grid.

BACKGROUND INFORMATION

The programs outlined in this question are examples of *partitions of a plane*. These patterns, as well as their algorithmic construction, play a role in different areas of computer science such as computer graphics and simulation.

It is interesting to note that biologists, mathematicians, and computer scientists are finding that many patterns found in the natural world are also created by the repetitive application of simple rules.

Voronoi-diagrams (similar to the result of case A) are also interesting and occur frequently in real world situations, for example, in cellphone coverage. Other patterns found in nature that follow simple rules include the Fibonacci sequence, which can be found in the arrangement of leaves in plants like succulents or cabbages.

Making use of abstraction thinking skills to solve this task quickly, instead of simulating each program entirely, is also a crucial skill in computer science.

This question comes from Austria

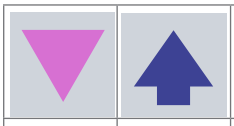
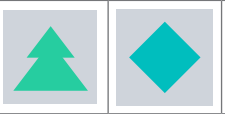
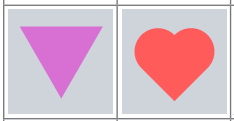
Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Medium

Treasure box

Maria found a box containing hidden treasure, but the box was locked.

To unlock the box, she needs to use the correct combination of three shapes.

Next to the box, Maria also found a note with hints to the right of some combinations, shown below.

	One shape is correct and in the correct place.
	Nothing is correct.
	Two shapes are correct, but in the wrong place.
	One shape is correct, but in the wrong place.
	One shape is correct, but in the wrong place.

Question

Which of the following combinations will unlock the treasure box?

EXPLANATION

Answer

Explanation

We will start by eliminating the shapes that do not fit into the combination that unlocks the box. In the second row of hints, we see that nothing belongs in our combination, which means that the Christmas tree, diamond and arrow do not lead to unlocking the box.

In the last row of hints, we see that one shape is correct, but placed incorrectly.

We have previously concluded that the Christmas tree and the arrow are not used, so the shape we need is a star, but it is placed incorrectly.

Continued on next page

Treasure box – cont'd

The possible positions of the star are as follows:

We can continue to find the other two shapes.

From the first row, we see that one shape is correct and in the right place. We know it cannot be the arrow, because it does not belong to the final combination, so it follows that one of these combinations is correct:

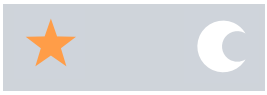
From the third row, we see that two shapes are correct, but placed incorrectly.

We definitely need the star, but its position is not in the middle, so now we have definitely found the position of the star and that position is given in the following picture:

We found one shape and it's correct position. Likewise, we continue this process to find the remaining two shapes.

From the first row, we had:

Since we found the star to be in the first position, the triangle can't be in the first position.
So, it follows that the moon is in its correct position. This leaves us with:



From the fourth hint, we see that one shape is correct, but incorrectly placed. The triangle is out, and the only position left is the middle position. The heart can't be correct, because it is in the middle position. Thus, the only shape that can be in the middle position is the circle.

There are other possible ways to determine the answer other than the explanation above. All these different ways will still lead to the correct answer.

BACKGROUND INFORMATION

In computer science, *pattern matching* is the act of checking whether a given sequence of tokens follows a set of defined characteristics.

In this task, the user is required to intepret the hints to create a pattern that follows all the hints. Each time the correct pattern is found, the treasure box will be unlocked.

Pattern matching is used in many different common applications. For example, it can be used to check if words are spelled correctly.

Beaver games

The Beaver Games are an annual competition consisting of three rounds. The first and second rounds are played in teams. A team's score is the sum of the scores of the beavers in the team.

- In round 1 the teams of four beavers are randomly selected. The team with the highest sum of points wins and advances to the next round.
- In round 2, new teams of two are selected, again by lottery, from the four beavers who advanced from round 1. The higher scoring duo then advances.
- In round 3, the remaining two beavers compete head-to-head. If you win three rounds in a row, you are the overall winner.

To make sure everyone has fun, even the beavers who do not advance to the next round still do all the activities and their scores are recorded so that next year they can aim for a PB (personal best score).

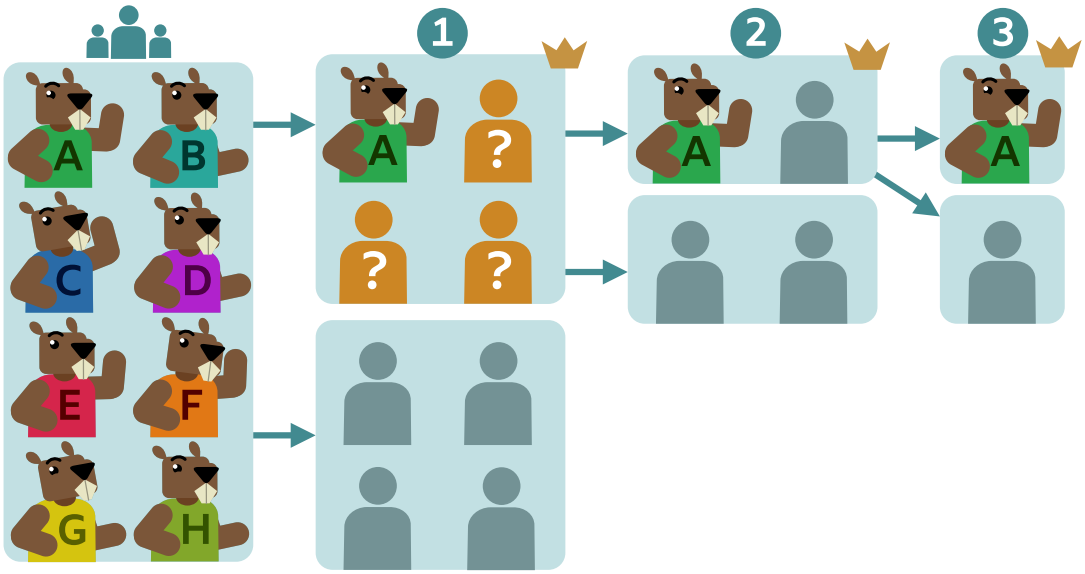
Beaver Ada 

was the lucky winner of this year's Beaver Games. For each challenge, the individual points are given in the table below.

Beaver:	A	B	C	D	E	F	G	H
Round 1	15	16	19	18	17	20	19	19
Round 2	20	27	30	24	28	24	30	30
Round 3	10	14	11	15	16	13	9	12

Question

Which three beavers were in Ada's first team in Round 1?



Continued on next page

This question comes from
South Korea

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Hard

Beaver games – continued

EXPLANATION

Answer

Ada's team in round 1.

Explanation

The third challenge is an individual match. Since Beaver G is the only player who has fewer points than Ada, they must have been on the same team in the second challenge.

Ada and Beaver G's sum of points in the second challenge is 50. This must be higher than the sum of points of the other two-person-team. The two players D and F are the only other ones whose sum of points is less than 50. Therefore, they have to have been in the same team as Ada and G in the first challenge.

Since we now know these members, we actually don't have to check the members of the other team. But, we can verify that in the first challenge the team of (Ada, D, F and G) has 72 points while (B, C, E and H) had 71 points. So Ada's team wins. In the second challenge, (Ada and G) had 50 points and (D and F) had 48 points. In the third challenge, Ada wins with 10 points against G with 9 points. So Ada is the overall winner.

BACKGROUND INFORMATION

The method used to solve this task is similar to a *backward search*. This technique is used by computer scientists when looking for a solution that has to fulfil some constraints. In some cases, a *forward search* and a backward search are combined in order to find a solution.

If one attempts to solve this task in a forward manner, one would actually have to consider all possible combinations, and evaluate for each of these combinations the results of the challenges. This takes a lot of time.

In such cases, computer scientists look for more efficient methods of solving the task. In this case, instead of solving the problem in a forward manner, deducting from the back gives one the correct solution very quickly.

This question comes from Italy

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Hard

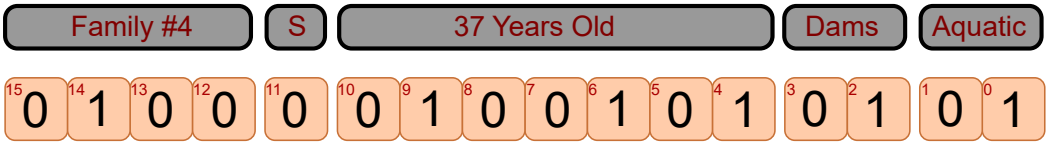
Beaver database

A dozen families live in the beaver village. Beaver Marta created a database of villagers, recording data about each beaver in the form of a binary 16-bit sequence. Each bit is labelled from b15 (left) to b0 (right), and represents data as follows:

- **b15-b12:** four bits for the **family number** (natural number);
- **b11:** one bit for their **teeth size** (0 = small, 1 = large);
- **b10-b4:** seven bits for their **age in years** (represents a natural number, eg: 0010100 = 20 years);
- **b3-b2:** two bits for **"skilled worker in"** (00 = construction of lodges, 01 = construction of dams, 10 = food collecting, 11 = education of young beavers);
- **b1-b0:** two bits for **favorite food** (00 = tree bark, 01 = aquatic plants, 10 = grasses, 11 = leaves).

For example, the sequence 0100 0 0100101 01 01 denotes a beaver that:

- belongs to family 4,
- has small teeth,
- is 37 beaver years old,
- is a skilled worker in the construction of dams,
- and likes aquatic plants.



Beaver Marta queries the database for beavers by using Boolean expressions that use positive logic (0 = false, 1 = true).

For example, the statement "b15 **and not** b11" would search for all entries where b15 is true (=1) and b11 is false (=0).

Question

Which set of beavers denotes the following expression?

b11 **and not** (b10) **and** b9 **and** b7 **and** not (b3 **and** b2)

Large teeth, of at least age 16, skilled worker in food cache.

Small teeth, of at least age 64, skilled worker in construction of lodges or dams.

Small teeth, between ages 40 to 63, skilled worker in construction of something or food cache.

Small teeth, of at most age 39, skilled worker in construction of dams.

Continued on next page

Beaver database – continued

EXPLANATION

Answer

Small teeth, between ages 40 to 63, skilled worker in construction of something or food cache.

Explanation

This question can be solved via decomposition of the query statement. Since: b11 = 1 means "small teeth", b10 = 0 means that the beaver can be no older than 63, b9 = 1 means that the beaver must be at least 32 years old. Requiring that either b3 = 0 or b2 = 0 means that we are excluding beavers skilled in educating young beavers, which requires both bits to be 1.

Note that according to De Morgan's laws **not** (b3 **and** b2) is equivalent to **not** (b3) or **not** (b2).

Alternatively, one could solve this question by finding the contradictions in the other three answers:

- Answer 1 is incorrect: For the beaver to have large teeth would require b11 to be false, which contradicts the statement.
- Answer 2 is incorrect: b10 being false means that the age of the beaver cannot be above 63, which contradicts the statement.
- Answer 4. is incorrect: b9 and b7 being true mean that the age of the beaver must be at least 40, which contradicts the statement.

BACKGROUND INFORMATION

In the context described above, a Boolean expression (or, better, a formula of propositional logic) can be used to query the simple database, that is, it may be viewed as a "search key". In general, such a formula identifies a subset of the data (and these situations can be graphically represented by means of Venn diagrams). Examples of the kind proposed here are found in Zuse's 1943-44 writings on propositional calculus and its applications to relay circuits. Konrad Zuse was a German civil engineer, but his greatest achievement was the first program-controlled electromechanical computer, that became operational in 1941.

Most modern programming languages have the Boolean unary operator not and binary operators and, or (possibly short-circuit). Operators not, and suffice – as well as not, or – by the so-called De Morgan's laws.

Boolean circuits, consisting of the corresponding logic gates, lie at the basis of many digital components used in computer engineering (multiplexers, adders, arithmetic logic units, and so on). In 1913 (but already discovered by Charles S. Peirce in 1880) Henry M. Sheffer showed that all logical operators can be derived from a single one, nand: A nand B is equivalent to not (A and B), thus A nand A is equivalent to not (A). Like nand gates, nor gates (A nor B is equivalent to not (A or B), thus A nor A is equivalent to not (A)) are "universal gates"; they can be combined to form any other logic gate, and in fact many electronic systems were built exclusively from nand gates or nor gates.

Ordering

Yuno has 11 cards with numbers from 1 to 9 and letters A or B on them. He uses the rule 9 < A < B.

Yuno wants to put all the cards into one row, by following rules like: **A → B**. This rule means that **card B must be somewhere to the right of card A.**

Yuno also wants the placement of the cards to create the smallest sequence possible. When comparing two sequences, if the first differing digit is smaller, then the entire sequence is considered smaller.

For example, consider the following sequences:

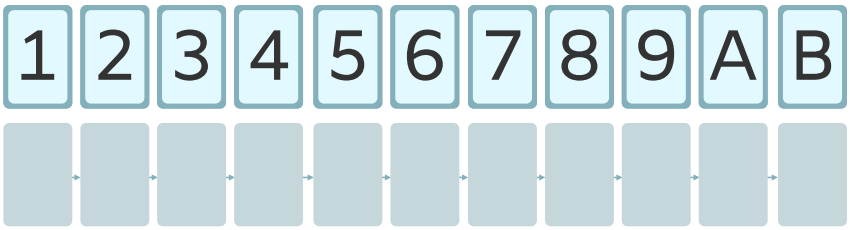
- sequence X: 5, 6, 5, 7, 9, 9, B.
- sequence Y: 5, 6, 5, 8, 0, 0, 0.

Sequence X and sequence Y match until the fourth number. This number is smaller in sequence X than in sequence Y (7 < 8). Therefore, regardless of the numbers that come afterwards, **sequence X is smaller than sequence Y.**

Question

Arrange the following cards into the smallest sequence possible, while following the rules set below.

1) 1 I 2	5) 3 I 5	9) 7 I 9
2) 1 I 6	6) 3I 4	10) 1 I B
3) 2 I 3	7) 5I 7	11) B I A
4) 6 I 5	8) 7 I 8	12) A I 7



EXPLANATION

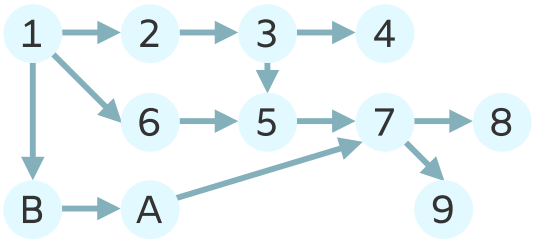
Answer

Continued on next page

Ordering – continued

Explanation

Let's look at this problem using graph theory. Each card will be considered a vertex of an oriented graph and each condition will be considered a directed edge from the left to the right card:



Next, we will list the entrance edges for each vertex:

1) -	5) 3, 6	9) 7
2) 1	6) 1	A) B
3) 2	7) 5, A	B) 1
4) 3	8) 7	

The lexicographically smallest sequence will be built as follows:

For each step, we will look at the vertices that don't have input edges and from those, we will choose the vertex with the lowest number.

The vertices and the edges we choose will be removed from the list of input edges.1) First, we will select the vertex that has no other input vertices. Current search sequence: **1.** Updated lists of input edges: **[1,...]**

1) -	5) 3, 6	9) 7
2) 1	6) 1	A) B
3) 2	7) 5, A	B) 1
4) 3	8) 7	

2) Now vertices **2, 6, and B** have no input edges. We choose vertex **2** (smallest number). Current search sequence: **[1, 2, ...]** Updated lists of input edges:

1) -	5) 3, 6	9) 7
2) -	6) -	A) B
3) 2	7) 5, A	B) -
4) 3	8) 7	

3) For the next three steps, consistently add vertices **3, 4, 6** in the current desired sequence: Current search sequence: **[1, 2, 3, 4, 6,...]** Updated lists of input edges:

1) -	5) 3, 6	9) 7
2) -	6) -	A) B
3) 2	7) 5, A	B) -
4) 3	8) 7	

Continued on next page

Ordering – continued

4) For the next three steps, consistently add vertices **5, B, A** in the current desired sequence.
Current search sequence: **[1, 2, 3, 4, 6, 5, B, A, ...]**

1) -	5) -	9) 7
2) -	6) -	A) -
3) -	7) -	B) -
4) -	8) 7	

5) The last three vertices **7, 8, 9.** We put them in ascending order. **[1, 2, 3, 4, 6, 5, B, A, 7, 8, 9]**

BACKGROUND INFORMATION

Algorithms for arranging elements in an ordered sequence are called *sorting algorithms*. They are one of the classics within algorithm theory.

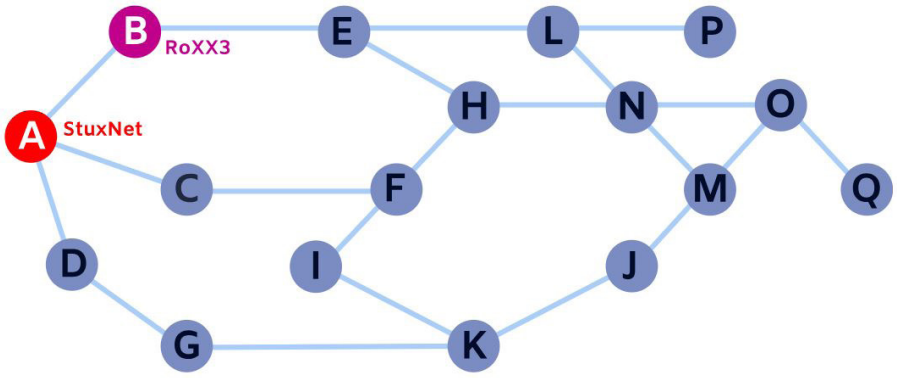
The simplest method to find the solution, is by trying all possible options (the so-called *brute force* method). This method works for sequences with a small number of elements.

However, with a large number of cards and set conditions, this method is very cumbersome and requires multiple checks of the same conditions. Meanwhile, the algorithm still needs to keep track of what the lexicographically smallest solution is.

Therefore, this problem is much more efficiently solved by the method of topological sorting, an example of which is demonstrated in the answer explanation.

Virus

In the following diagram, the circles are computers and the lines are connections between them.



Computer A is infected with the StuxNet virus and Computer B is infected with the RoXX3 virus. At the beginning of each new day, each virus spreads to any computers that are directly connected to an infected computer. If both viruses end up on a single computer, that computer is automatically destroyed and no further spreading will occur from that computer in the following days. After a few days, every computer is either infected or destroyed.

Question

How many computers are operational but infected with the RoXX3 virus after a week?

5 | 6 | 7 | 8 | 9 | 10

EXPLANATION

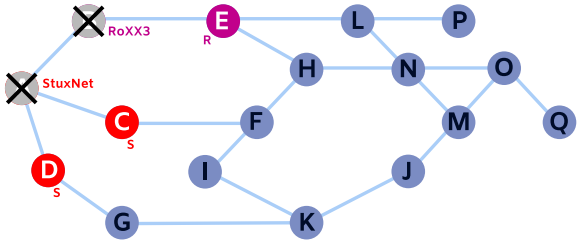
Answer

6 computers are still operating but infected with RoXX3.

Explanation

By simulating the rules for each day, we can find the correct answer:

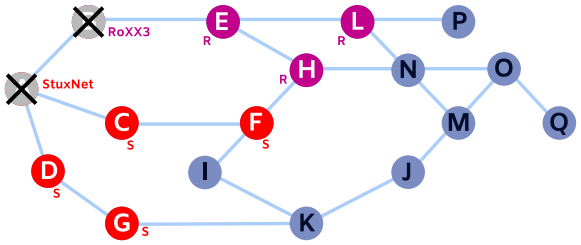
Day 1: B,C,D are infected with StuxNet (by A), and A & E are infected with RoXX3 (by B), destroying
A & B.



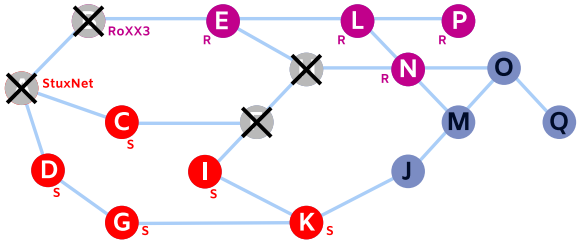
Continued on next page

Virus – continued

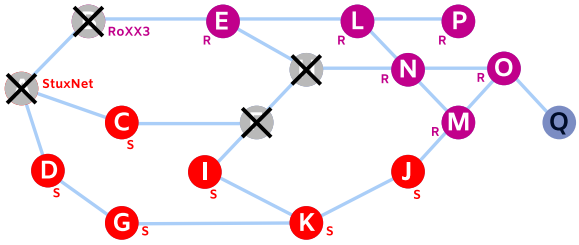
Day 2: F & G are infected with StuxNet (by C and D respectively), and H & L are infected with RoXX3 (by E), no destructions.



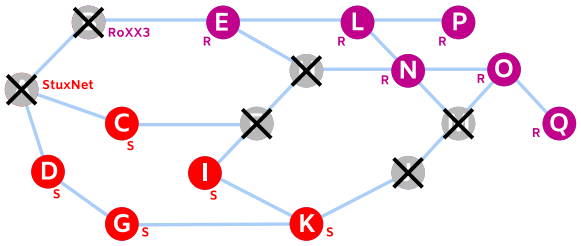
Day 3: H,I,K are infected with Stuxnet (by F, F, and G respectively), and N & P are infected by RoXX3 (by H, and H & L respectively), destroying H & F.



Day 4: J is infected with StuxNet (by K), and O & M are infected with RoXX3 (by N), no destructions.



Day 5: M is infected with StuxNet (by J), and Q & J are infected with RoXX3 (by O and M respectively, destroying J & M.



As neither StuxNet or RoXX3 can make new infections, the computers infected by RoXX3 are: BAEHLNFPOMJQ (roughly in order of infection) so 12 got the virus. Of these ABFHMJ are destroyed (6) leaving 6 infected with RoXX3 but operational (E,L,P,N,O,Q).

BACKGROUND INFORMATION

This task demonstrates how graphs can be used to represent dependencies between items. A graph is a data structure that is used a lot in Computer Science to demonstrate relationships.

The computers in this problem are *nodes*, which represent data points. The communication lines are *edges*, which represent the connection between data points. Graphs also make it easier to visualise a task, compared to just reading the descriptions of the relationships in text. In this problem nodes are removed, thus eliminating edges or possible connections.

Seashells and pebbles

Anushka and Bruno are playing a game on the beach.

Anushka has collected many light coloured shells  ,

and Bruno has collected many dark pebbles  .

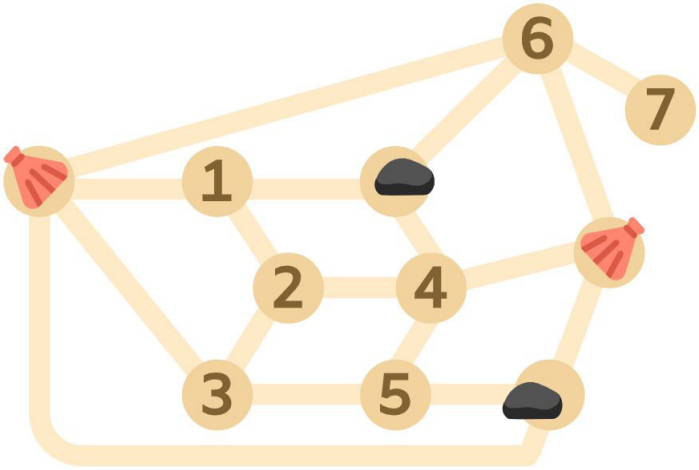
They make holes in the sand, and connect them with narrow paths called trenches. They then take turns to put one of their pieces in an empty hole.

The loser is the first player to place two of their pieces into two holes connected by one trench.

Anushka started the game. The game has reached the position shown in the figure below. It is now Anushka's turn.

Question

In which numbered empty hole must Anushka place a shell to ensure victory for herself?



EXPLANATION

Answer

Hole number 7.

Continued on next page

This question comes from Italy

Years 3+4
Years 5+6
Years 7+8
Years 9+10
Years 11+12 Hard

Seashells and pebbles – cont'd

Explanation

Holes 1, 3, 4 and 6 are "closed" to Anushka, because this would means she is placing directly next to a shell and would lose. Similarly, holes 1, 4, 5 and 6 are "closed" to Bruno. The holes that are "open" to each player are therefore the following:

- Anushka: 2, 5, 7
- Bruno: 2, 3, 7

After Anushka has put a shell in 7, Bruno can play either 2 or 3. Regardless of which of these Bruno chooses, he also "closes" off the other one as they are next to each other. In both cases, Anushka just puts a shell on 5 to force Bruno to lose as he no longer has any "open" holes.

If, from the position shown in the figure, Anushka had put a shell in 2, the game would have continued in this way: Bruno 7, Anushka 5, Bruno 3, and Anushka loses.

Finally, if Anushka had put a shell in 5, the game would have continued in this way: Bruno 7, Anushka 2, Bruno 3 and Anushka loses again.

It is interesting to note that, in the position shown in the figure, if it were Bruno's turn, Bruno would lose anyway. In fact, if Bruno occupies either 2 or 3, Anushka occupies 7; instead, if Bruno occupies 7, Anushka occupies 2, then Bruno 3 and Anushka 5.

BACKGROUND INFORMATION

This question demonstrates an example of COL, a two-person perfect-information positional game introduced by Colin Vout. There are many popular games in which the two players, after placing their pieces on the game board, take turns moving them to reach a certain goal (e.g., a "mill" in Nine men's morris, or three-in-a-row in Picaria). The board has a fixed pattern, or it is built ad *hoc* by the players or someone else (as in COL). In any case, the game analysis and "solution" can be performed by setting up a specific computer program.

In all these games, the board can be modelled by a *simple and undirected* (and also finite, connected, planar) graph, where pieces occupy *vertices* and connections do not link a vertex with itself, nor does it have the same repeated connection. In general, in mathematics and computer science, a simple and undirected graph is a structure suitable for representing a symmetric relation between the elements of a set.

Game trees, which are trees that start with the current board state and branch out to all possible board states in the future, are useful in developing algorithms for winning strategies for these types of games. In this question, it is possible to use a brute force approach of checking all possible outcomes in this game tree to find the winning strategy. However, for much larger game trees (in chess, for example), it can be computationally too difficult to search every state of play. *Negamax*, a variant of *minimax*, is just one common example of an algorithm used to find winning strategies from a game tree. Highly efficient algorithms of this type have been used in many applications, famously in the Deep Blue chess program of the 1990s.

We would like to thank the International Bebras Committee and community for their ongoing assistance, resources and collaborative efforts. Special thanks to Eljakim Schrijvers, Edwin Tomlins, Alieke Stijf and Dave Oostendorp for their support and technical expertise.

If you would like to contribute a question to the International Bebras community, please contact us via the details below.

Contact us

CSIRO Digital Careers
digitalcareers@csiro.au
csiro.au/education/Programs/Digital-Careers