

Bebras Australia Computational Thinking Challenge

2022 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 2.9 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 March 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

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.

digitalcareers.csiro.au

481

Australian schools
participated in
Round 1 2022



27,435

Australian students
participated in
Round 1 2022



2.9 million

Students participate
worldwide



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 2022. 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 a Solutions Guide?	3
What is Computational Thinking?	6
Computational Thinking skills alignment	7
Computational Thinking skills alignment	8
Australian Digital Technologies curriculum key concepts	9
Digital Technologies key concepts alignment	10
Digital Technologies key concepts alignment	11

Years 7+8 **12**

Downtown	13
Best Route	14
Wrestling Holds	16
Butterflies	18
Line of Fish	19
Colourful Tube	21
Hey Taxi!	22
Towns and Highways	25
Ada's Marble Machine	27
Party Foul	28
Hashing	30
Error Detection	33
Meeting Race	35
Desk Trouble	39
Art Theft	41

Years 9+10 **43**

Necklaces Instruction	44
Chez Connie	45
Detective Lawn Mower	48
Viewer Numbers	50
Encrypted Path	52
Fruit Stack	54
Bird Migration	57
Ada's Marble Machine	59
Audit Committee	61
Stacks of Tokens	62
Truchet Tiles	64
Spider Quilts	66
Jumping Jack	69
Still Life	71
Playing with Hats	73

Years 11+12

Vaccination Centres	76
Secret of the Diary	77
Robo-Rally	79
Snow White	80
Comfort Temperature	83
Secret Number	85
Symbol Reading Robot	87
Cupcakes	89
Bench Workshop	93
Burrow Business	95
Quiz Night	97
Counting by Nodding	100
Log Sort	102
Unification	104
Two Beavers are Working	106

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

2022 Round 1 Questions	Grade level	Decomposition	Pattern Recognition	Abstraction	Modelling & Simulation	Algorithms	Evaluation
Years 7+8							
Downtown	Easy						
Wrestling Holds	Easy						
Butterflies	Easy						
Line of Fish	Easy						
Best Route B	Easy						
Colourful Tube	Medium						
Hey Taxi	Medium						
Towns and Highways	Medium						
Ada's Marble Machine A	Medium						
Party Foul	Medium						
Hashing A	Hard						
Error Detection	Hard						
Meeting Race	Hard						
Desk Trouble	Hard						
Art Theft	Hard						
Years 9+10							
Necklaces Instruction	Easy						
Chez Connie A	Easy						
Detective Lawn Mower	Easy						
Viewer Numbers	Easy						
Encrypted Path	Easy						
Fruit Stack	Medium						
Bird Migration	Medium						
Ada's Marble Machine B	Medium						
Audit Committee	Medium						
Stacks of Tokens B	Medium						
Truchet Tiles	Hard						
Spider Quilts	Hard						
Jumping Jack	Hard						
Still Life	Hard						
Playing with Hats A	Hard						

Computational Thinking skills alignment

2022 Round 1 Questions	Grade level	Decomposition	Pattern Recognition	Abstraction	Modelling & Simulation	Algorithms	Evaluation
Years 11+12							
Vaccination Centres	Easy						
Secret of the Diary	Easy						
Robo-rally	Easy						
Snow White	Easy						
Comfort Temperature	Easy						
Secret Number	Medium						
Symbol Reading Robot	Medium						
Cupcakes A	Medium						
Bench Workshop	Medium						
Burrow Business	Medium						
Quiz Night	Hard						
Counting by Nodding	Hard						
Log Sort A	Hard						
Unification	Hard						
Two Beavers are Working	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

2022 Round 1 Questions	Grade level	Abstraction	Data Collection	Data Representation	Data Interpretation	Specification	Algorithms	Implementation	Digital Systems	Interactions	Impacts
Years 7+8											
Downtown	Easy										
Wrestling Holds	Easy										
Butterflies	Easy										
Line of Fish	Easy										
Best Route B	Easy										
Colourful Tube	Medium										
Hey Taxi	Medium										
Towns and Highways	Medium										
Ada's Marble Machine A	Medium										
Party Foul	Medium										
Hashing A	Hard										
Error Detection	Hard										
Meeting Race	Hard										
Desk Trouble	Hard										
Art Theft	Hard										
Years 9+10											
Necklaces Instruction	Easy										
Chez Connie A	Easy										
Detective Lawn Mower	Easy										
Viewer Numbers	Easy										
Encrypted Path	Easy										
Fruit Stack	Medium										
Bird Migration	Medium										
Ada's Marble Machine B	Medium										
Audit Committee	Medium										
Stacks of Tokens B	Medium										
Truchet Tiles	Hard										
Spider Quilts	Hard										
Jumping Jack	Hard										
Still Life	Hard										
Playing with Hats A	Hard										

Digital Technologies

key concepts alignment

2022 Round 1 Questions	Grade level	Abstraction	Data Collection	Data Representation	Data Interpretation	Specification	Algorithms	Implementation	Digital Systems	Interactions	Impacts
Years 11+12											
Vaccination Centres	Easy										
Secret of the Diary	Easy										
Robo-rally	Easy										
Snow White	Easy										
Comfort Temperature	Easy										
Secret Number	Medium										
Symbol Reading Robot	Medium										
Cupcakes A	Medium										
Bench Workshop	Medium										
Burrow Business	Medium										
Quiz Night	Hard										
Counting by Nodding	Hard										
Log Sort A	Hard										
Unification	Hard										
Two Beavers are Working	Hard										

Bebras Challenge 2022 Round 1

Years 7+8

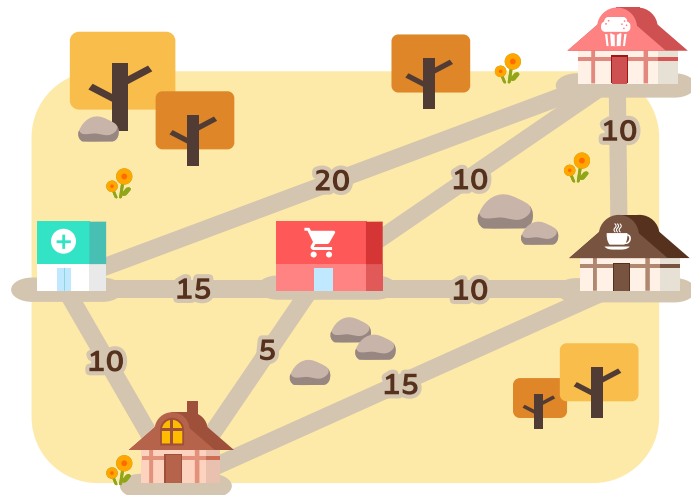


Downtown

Beaver Ben has to do some shopping downtown. The numbers on the roads in the image below show the walking time from one shop to another. Ben's route starts and ends at home (the house at the bottom of the map).

Question

What is the time of the shortest route to visit all four shops and get back home?



EXPLANATION

Answer

The correct answer is 55.

Explanation

There are two shortest ways for Ben. One shortest way is as follows: home to grocery to cafe to cake shop to barbershop to home.

We get another way by following all the arrows along the opposite direction: home to barbershop to cake shop to cafe to grocery to home.

To answer the question quickly, we can simplify the map. Ben can ignore the road between the barbershop and the grocery, because it is just as fast to go from barbershop to home to grocery, or the other way around. Ben can also ignore the road between home and the cafe, because it is just as fast to go from home to grocery to cafe, or the other way around.

After the map is simplified, it is easy to see what the shortest way must be.

BACKGROUND INFORMATION

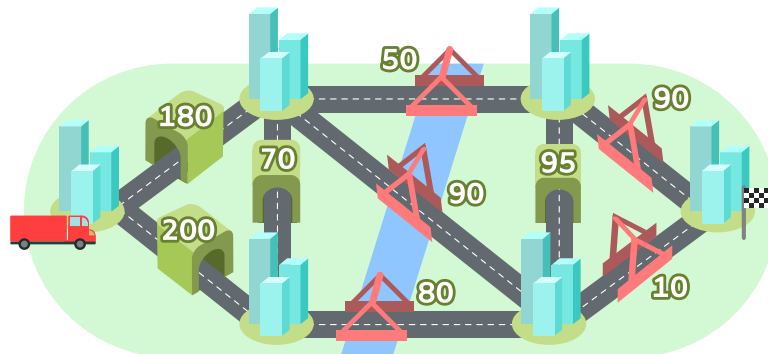
This task is similar to the *Traveling Salesman Problem* (TSP). The problem is to find the best route for a salesman to visit a list of cities on a map and return home. It is a very famous problem in informatics. This problem arises in many practical applications. For example, when an express delivery robot tries to find a route to distribute goods to different destinations, its path planning program may solve a TSP. Another example is when driving a car, the navigation software in your mobile phone uses TSP to find a way to avoid traffic jams.

TSP is a hard problem. You may find it not very difficult to help Beaver Ben in the task above. But with even just a few more cities or shops, say around a hundred, computers will take nearly forever to find the best route using currently-known algorithms. When there is a need to solve TSP for more cities or shops computer scientists design algorithms to find approximations (routes which are short enough rather than shortest) because it takes too much time and computational resources to find the overall shortest solution.



Best Route

Trucks travel between cities on the highways shown below.



Bridges and tunnels limit how high trucks can be. Specifically, the label on each highway is the maximum height of a truck that can use the highway.

Question

What is the maximum height of a truck that can be sent from Start to Finish?

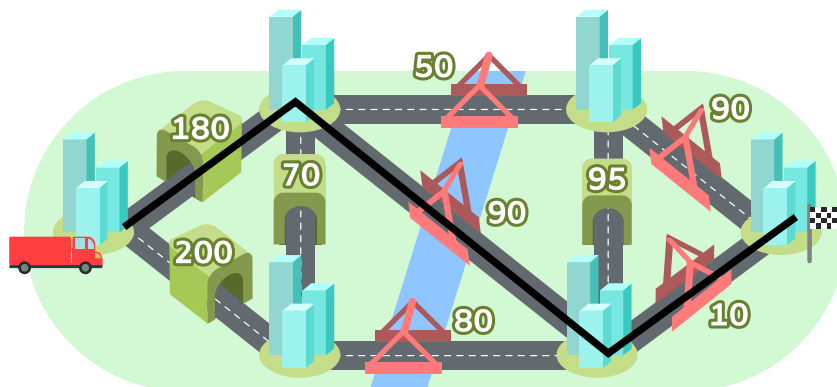
EXPLANATION

Answer

The correct answer is 90.

Explanation

To get from Start to Finish, a truck must pass from the three leftmost cities to the three rightmost cities. Only three highways can be used to do this and the limits on these three highways are 50, 90 and 80. This tells us that a truck with a height greater than 90 cannot be sent from Start to Finish.



Continued on next page



This question comes from
Cyprus

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



Best Route – continued

Is it possible for a truck with a height of 90 to be sent from Start to Finish. Yes! If a truck travels using the route shown below, then the height limits it will encounter will be 180, 90, 95 and 90. The smallest of these is 90. This means it is possible for a truck of height 90 to be sent from Start to Finish.

Another (longer) way to solve this problem is to carefully check all the possibilities either Starting from Start or working backwards from Finish.

BACKGROUND INFORMATION

This problem was inspired by limits on *data* sent on a *computer network* (e.g. wifi or cables in a building). *Routers* and *routing algorithms* typically direct data traffic. We typically say that *data packets hop* from segment to segment. One important consideration is the *bandwidth* of a hop which gives us a limit on how quickly data packets can be transferred along the segment over a given period of time. This is somewhat like the limit that bridges put on trucks in this problem.



Wrestling Holds

Tohn Beava is training to become a professional wrestler. He knows that during a match, he can be in the ring in any of the six different positions listed below:

- Laying
- Standing
- Running
- Against the ropes
- In the corner
- Top Rope

His wrestling trainer can teach him a set of moves, and each move has a list of positions that it can be performed from. John wants to make sure that he learns a move for every position, but wants to learn the fewest number of moves possible, to make sure that he has more time to practice each one. The moves that his trainer can teach him and the positions they can be used from are as follows:

- Crossbody – Running, Top Rope
- Suplex – Standing, Top Rope
- Clothesline – Standing, Running, Top Rope, Against the ropes
- Back Elbow – Standing, In the corner, Against the ropes
- Armbar – Standing, Laying
- Running Splash - Running



Question

What is the minimum number of moves John needs to learn to make sure that he can perform a move from any position?

2

3

4

5

EXPLANATION

Answer

The answer is 3.

Explanation

There are several ways to achieve this with only 3 moves.

One possible way to attempt this question is to find the move that covers the largest number of remaining positions possible. In this case, Clothesline (M3) will cover 4 positions – Standing, Running, Top Rope and Against the ropes (P2, P3, P4, P6). This leaves us with 2 positions left to be covered – Lying (P1) and In the Corner (P5). There are no moves that will cover more than one of the remaining positions, so we need to select Armbar (M5) and Back Elbow (M4). Thus, 3 moves (M3, M4, and M5) covers all positions (P1, P2, P3, P4, P5, and P6).



Wrestling Holds – continued

We found an answer with 3 moves. Because the minimum number of moves is sought, we must argue that it is not possible to have an answer with two moves. This is because if we take any pairs of moves, that pair will not cover all 6 position. For pair containing in total less than 6 positions (e.g. for pair of M1 and M2) this is obvious. And for other pairs, this is easy to check.

BACKGROUND INFORMATION

The problem is also known as a *Dominating Set* cover problem. Imagine six sets, each representing a different move. Every set contains the positions that move can be performed from. The task is to select the smallest number of sets (moves) such that the union of these sets (all positions from which the list of moves can be performed) contains all elements (positions). In other words, the task is to cover all elements with the smallest number of sets.

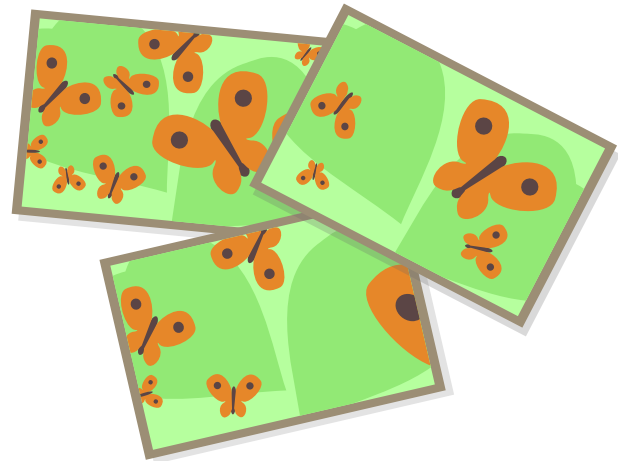
In this case, we are limited in our choices of set for two of the given positions, making it easier to select sets that will cover the specific list of elements.

The method used for finding the answer in the explanation is called a *greedy algorithm*, where the algorithm selects the best possible answer at each stage. At the selection of each move, this algorithm will select the move that will cover the highest number of positions. It should be noted that a greedy algorithm is not guaranteed to always produce the optimal solution, but in this case, it will do so successfully.



Butterflies

A beaver is photographing butterflies, but after each photo is taken, half of the butterflies fly away. The first photo has 64 butterflies on it and the last photo has just one butterfly on it.



Question

How many photographs did the beaver take?

5

7

3

6

EXPLANATION

Answer

The answer is Option D, 6 photos.

Explanation

We are told that the first photo has 64 butterflies in it. Since half the butterflies fly away after each photo is taken, we can record how many butterflies are in each photo.

We see that the photo with just two butterflies in it is photo number 6.

Photo Number	Number of Butterflies
1	64
2	32
3	16
4	8
5	4
6	2
7	1

BACKGROUND INFORMATION

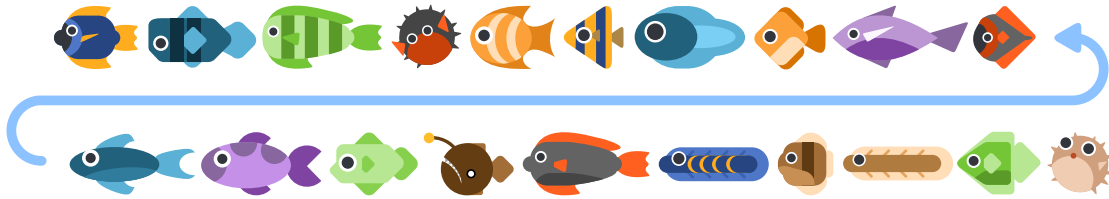
The number of photos the beaver takes is just one more than the number of times that half the butterflies fly away. So after the number of butterflies is cut in half only five times, we are left with two butterflies. Notice that six is quite a bit smaller than 64. This is not a coincidence.

The process of repeatedly cutting a quantity in half until you end up with only one item, can occur quite quickly. That is, you can move from a very large number to a very small number in this way in a relatively small number of steps. This idea is used by computer scientists to design algorithms which are very *efficient*. The most famous algorithm based on this idea is probably something called *binary search*. In a math class in a later grade, you might learn about *logarithms* which help measure this kind of process.



Line of Fish

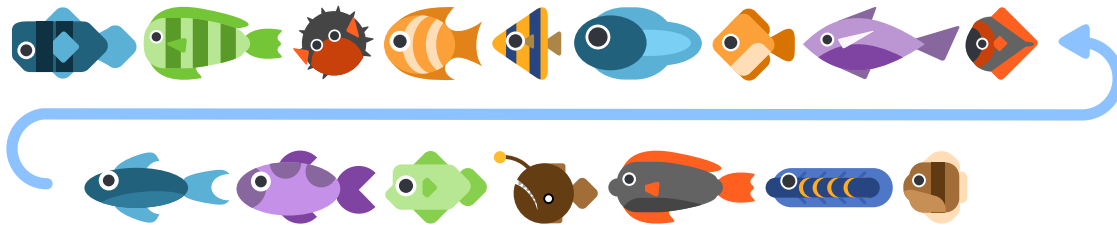
Fish swim in a line as shown.



Occasionally, someone says the positions of two fish. If these positions are A and B where $A < B$, then:

- all fish to the left of position A swim away, and
- all fish to the right of position B swim away.

For example, after someone said positions 2 and 17, there would then be 16 fish remaining in the line (now in positions 1, 2, ... 16) as follows:



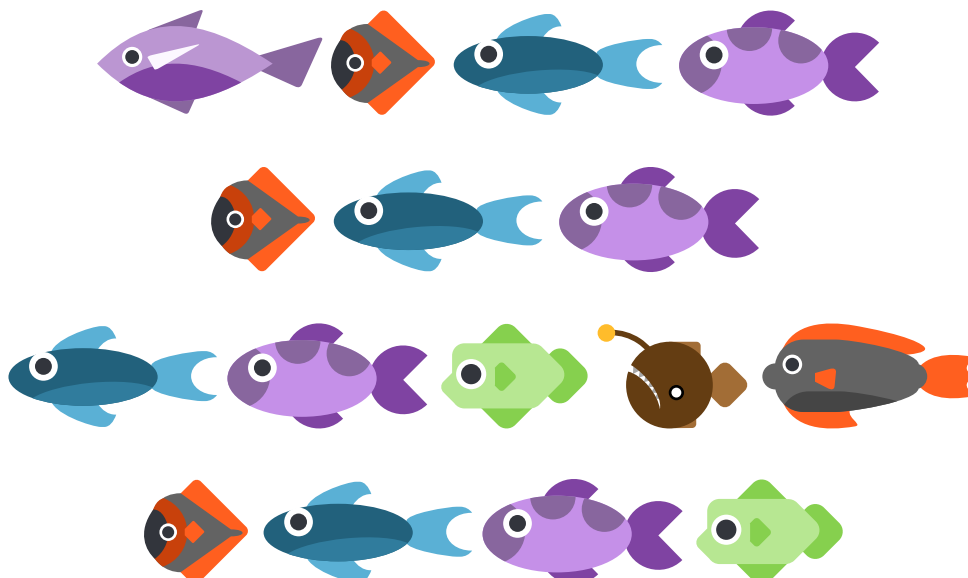
Positions are numbered starting from 1 on the left and positions are renumbered after any fish swim away.

Starting with the original line of 20 fish:

- someone says positions 4 and 18, then
- someone says positions 6 and 12, and then
- someone says positions 2 and 5.

Question

After this, which of the following is the new line of fish?



Continued on next page

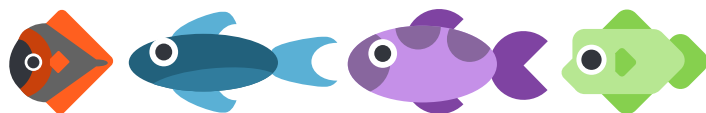


Line of Fish - continued

EXPLANATION




Answer







The correct answer is:




Explanation

One way to determine which fish remain is to write down the entire remaining line of fish after each time someone says two positions. However, we can be a bit more clever by only keeping track of the leftmost remaining fish and the number of remaining fish. This is because when fish swim away, only fish that were originally in adjacent positions remain.

After positions 4 and 18 are called, fish ,  and  will swim away as well as some fish to the right of position 18 so will be the left most remaining fish. Also, $18 - 4 + 1 = 15$ fish will remain in the line. (In general, after calling out positions A and B, $B - A + 1$ fish will remain in the line. Can you see why?)

Then, after positions 6 and 12 are called, fish , , ,  and  will swim away as well as some fish to the right of position 12 so  will be the left most remaining fish. Also, $12 - 6 + 1 = 7$ fish will remain in the line.

Finally, after positions 2 and 5 are called, fish  will swim away as well as some fish to the right of position 5 so  will be the left most remaining fish. Also, $5 - 2 + 1 = 4$ fish will remain in the line.

This means that the four fish beginning with  form the new final line of fish.

BACKGROUND INFORMATION

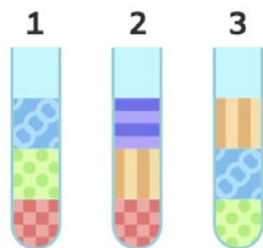
When a computer programmer is working with data, they need to determine how to represent this data. Related data is normally stored together in a collection. In this case, a second important decision is to determine how to arrange the collection in memory. Different data types can be used for this and one of the most common data types is a sequence. In this task, the fish are arranged in a sequence.

Data types are usually associated with common operations performed on the data. The key operation in this task is the selection of fish between two given positions. This is one of the most important sequence operations in general. When the sequence is a list of letters or other characters, it is typically called a string and this common operation is often named substring, or slice, or something similar.



Colourful Tube

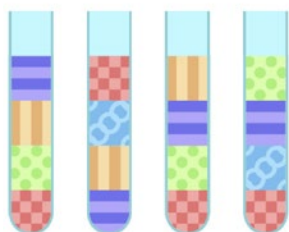
Omar wants to fill a tube with various types of liquids. Omar's science teacher explains that a liquid with a higher density will be *under* a liquid that has a lower density. The teacher gives Omar three examples of tubes which, between them, are filled with all the liquids available in the lab.



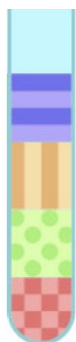
Omar uses the information in the three examples above to fill a new tube with a different combination of the available liquids.

Question

Only one of the tubes shown below is possible. Which tube could Omar have filled?



Answer



EXPLANATION

Explanation

By studying tubes 1, 2 and 3, the order of all liquids according to their density can be found, which is:



By comparing the available answers with the overall density order we can then easily spot tubes that cannot exist. There is only one tube that is consistent with the overall density order, which is the correct answer.

BACKGROUND INFORMATION

The question shows a repeating pattern that requires students to identify similarities in the component parts of a problem, namely comparing where liquids appear in each tube to create an ordering of all the available liquids. This skill is often called *pattern recognition* and involves looking for patterns in the problem, finding links to any of the problems or solutions that one may have encountered in this problem or in previous problems, and applying them to a new situation. Using this skill, it is possible to use what we have learned in the past to help us solve this new problem.

Two key parts of this skill are called *ordering* and *constraints*.

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

Constraints are a condition of a problem that the solution must satisfy. In this question, the density of the liquids becomes a constraint for ordering the liquids.



This question comes from
Austria

Years 3+4
Years 5+6

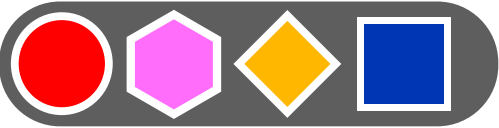
Years 7+8 Medium

Years 9+10
Years 11+12



Hey Taxi!

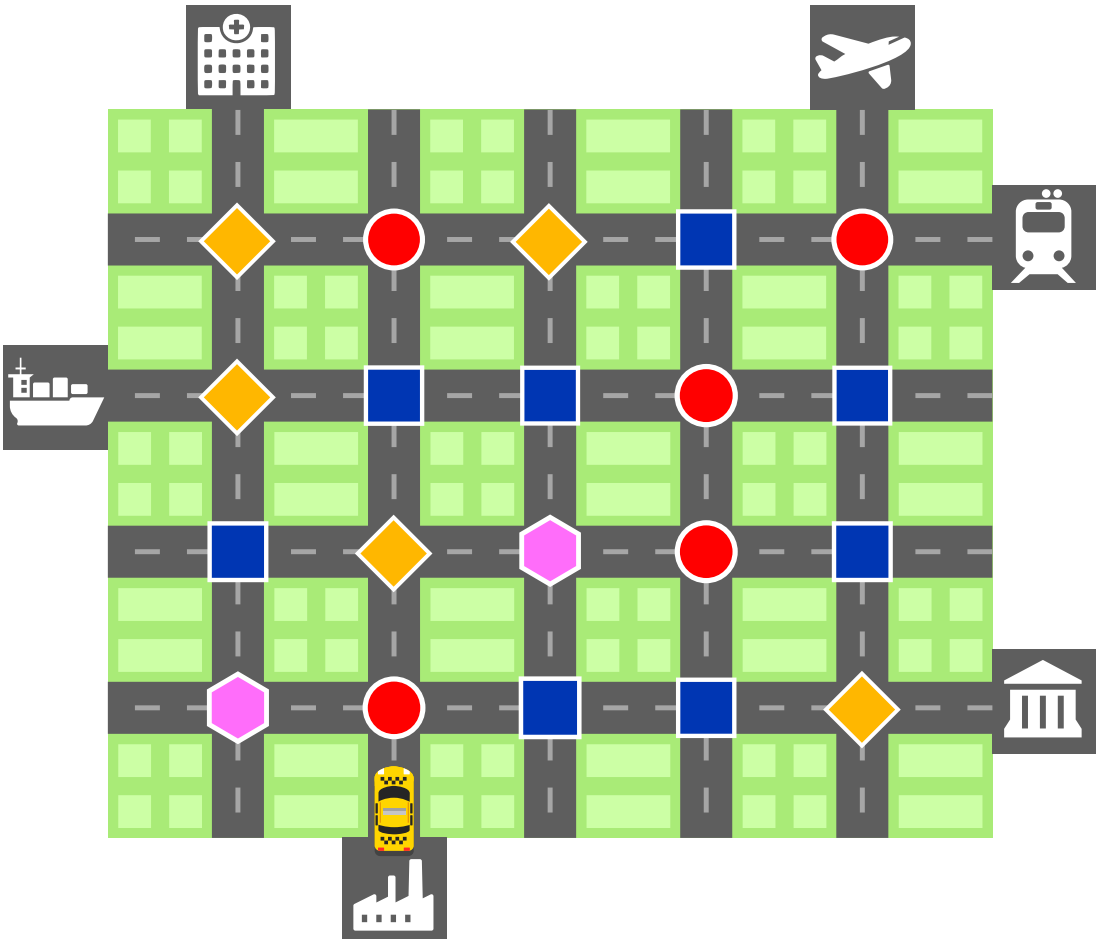
A self-driving taxi drives people to their desired destination.
The smart traffic signs know where the taxi should go and use these symbols to direct the taxi.



Each symbol has one of these meanings: forward, turn left, turn right and turn back.
Right, left, forward, and turn back are always relative to the orientation of the taxi.

				<div>Left Right</div> <div>Right Left</div>
Forward	Left	Right	Turn back	

The taxi moves one block at a time, and follows the meaning of each symbol.





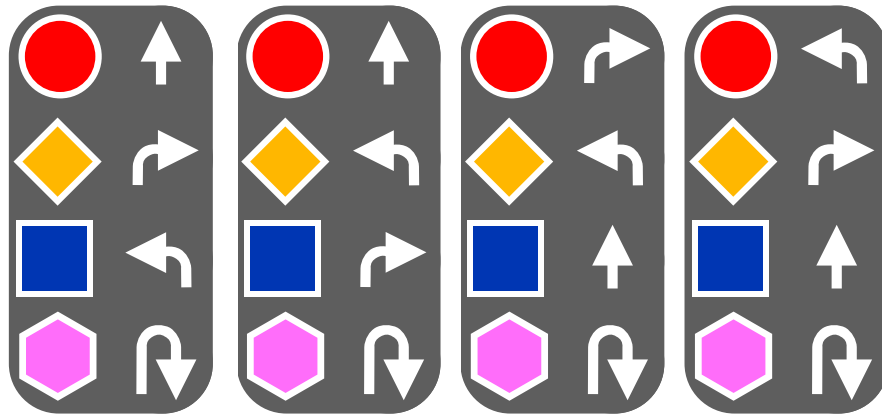
Continued on next page



Hey Taxi! – continued

Question

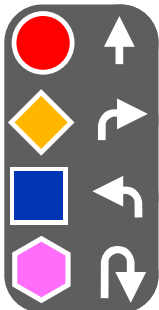
The traffic symbols in this picture direct the taxi from the industrial park  to the airport  .
What is the meaning of each traffic symbol?



EXPLANATION

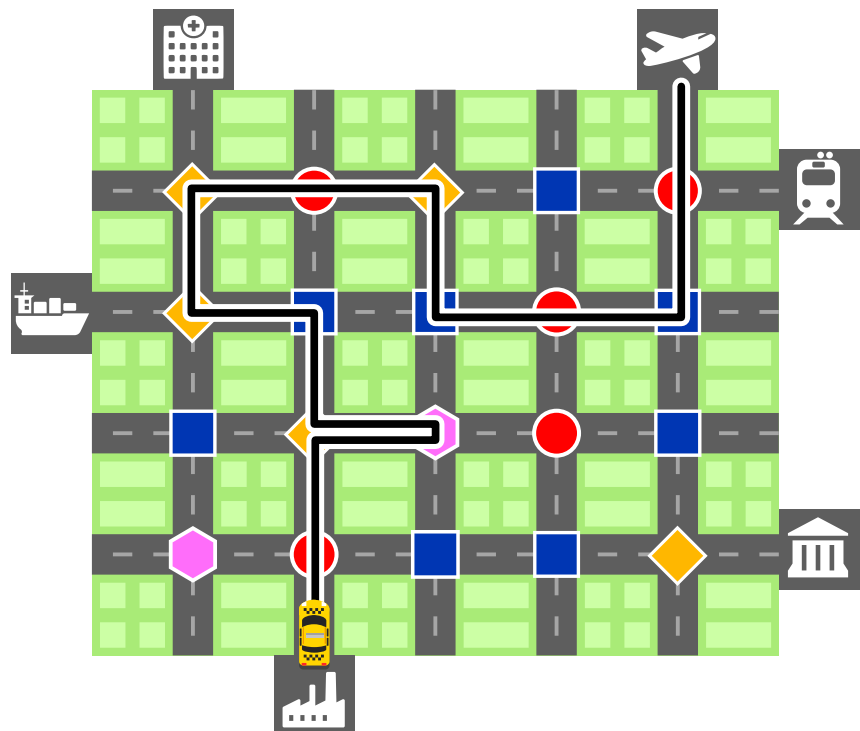
Answer

The correct answer is:



Explanation

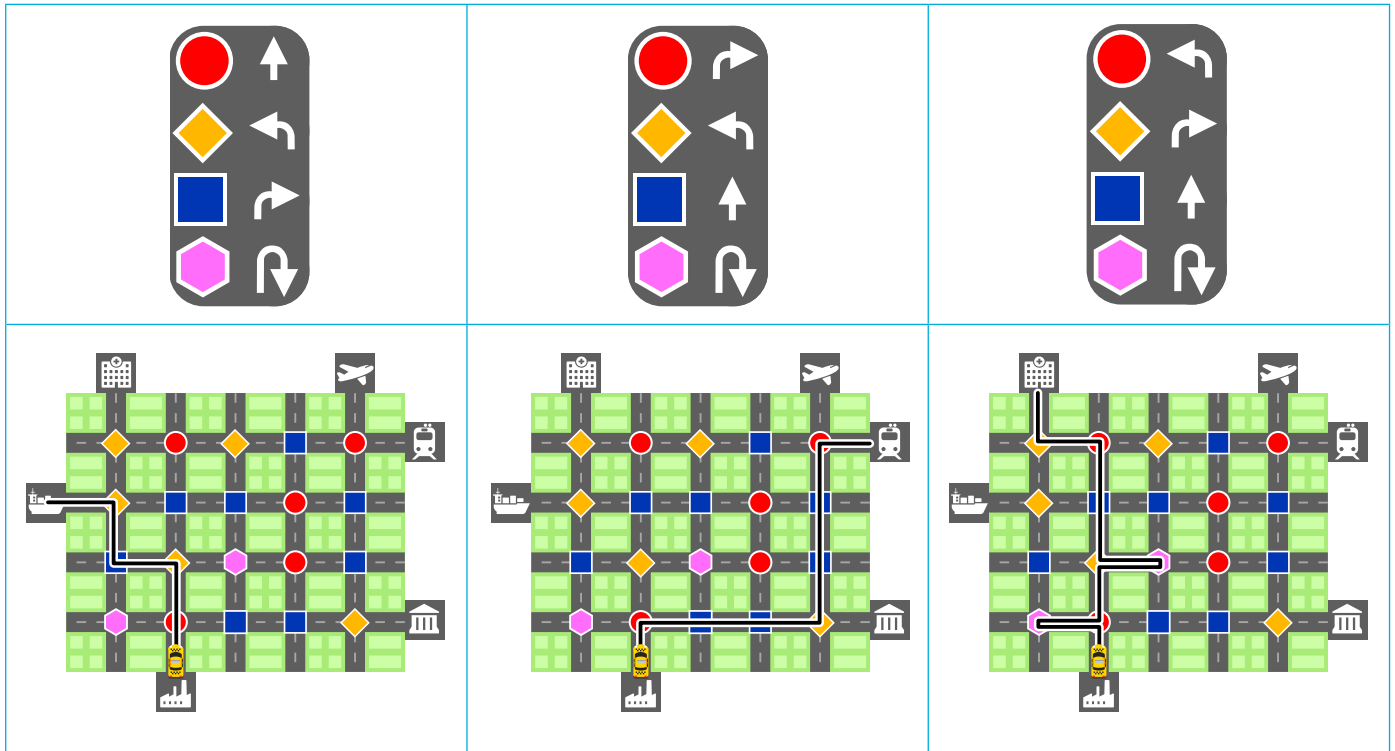
The path from the starting point to the destination is shown:





Hey Taxi! – continued

The other options end up in other destinations.



BACKGROUND INFORMATION

The computational thinking concept illustrated with this task is algorithms. A very simple computer program is written using four different types of instructions. Given the output of the program, you must figure out which symbol means which instruction.

Self-driving cars and other self-driving vehicles are examples of artificial intelligence that are slowly becoming an everyday part of life. The taxi in this task would need to be equipped with a broad range of sensors (like cameras, radar, ultrasonic) to understand its environment.

Computer vision software would use these sensors to keep the taxi in lane, follow signs, and avoid pedestrians.

While the taxi in this task is self-driving, it is not fully autonomous, because it follows one sign after another to get to its destination. An autonomous vehicle would use artificial intelligence to decide its own route based on sensing of the environment, GPS and map data, traffic reports, and even information from other autonomous vehicles!



Towns and Highways

Below is a map of five towns and four highways. The black dots with letters are the towns. The coloured lines are the highways.

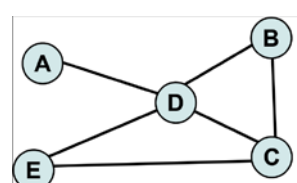
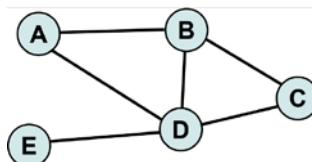
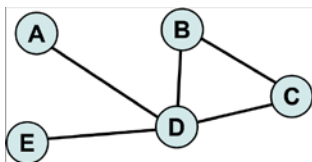
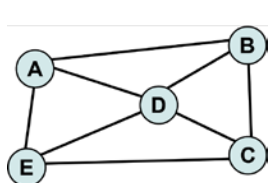


This map can be represented as a simplified diagram in which:

- towns are represented by circles
- two towns are connected with a line when they lie on the same highway.

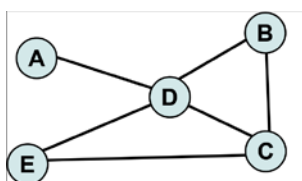
Question

Which diagram represents the map correctly?



EXPLANATION

Answer



Explanation:

By studying the map carefully, we can see that:

- Town A lies on the same highway as the town D.
- Town B lies on the same highway as the towns C and D.
- Town C lies on the same highway as the towns B, D and E.
- Town D lies on the same highway as all the other towns.
- And finally, the town E lies on the same highway as the towns C and D.



Towns and Highways – continued

These conditions are met by only the fourth diagram option, which is the correct answer. All of the other options are wrong:

- For the first option, there are connection between towns A and B, as well as between A and E, and those towns don't lie on the same highways.
- For the second option, the connection between towns E and C is missing.
- For the third option, not only is the connection between the towns E and C missing, but there is a connection between towns A and B when these towns don't lie on the same highway.

It is important to note some subtle but important features of the map that can lead to an incorrect choice of diagram:

- Even though town A can reach town B, they are actually not on the same highway.
- Even though towns E and C has another town in between them, they are still on the same highway.

BACKGROUND INFORMATION

In computer programs, reality must be represented by some data so that the computer can process them. The same reality can usually be represented in several different ways and computer programmers need to choose the one that is the most suitable for the task (algorithm) their program has to perform. Therefore it is essential for computer programmers to know many kinds of representations.

One commonly used representation is a *graph* – a diagram made of vertices (usually circles with names in or besides them) and edges (lines connecting vertices). A big part of informatics (where it overlaps with discrete mathematics) is called *Graph Theory* and it defines many useful algorithms for data represented by graphs.

But, as this task shows, the same reality can be represented by several kinds of graphs – the first picture is also a graph (when we abandon the parts of highways that do not connect any two towns), but it connects the towns in a slightly different way than the graph in the solution.



Ada's Marble Machine

Ada the engineer has been asked to create a sorting machine to sort marbles based on the following aspects:

- size (small, medium or large)
- colour (red, blue or yellow)
- material (stone, glass or metal)
- decoration (glitter, plain or mosaic).

Ada knows the following restrictions on the marble designs:

- each marble can only be of one size, one colour, one material, and one decoration
- marbles made of metal cannot be large-sized
- marbles made of stone cannot be red or yellow
- the glitter decoration cannot be applied to marbles made of metal or stone
- the mosaic decoration cannot be applied to marbles made of metal

Question

If a large, blue marble is plainly decorated, what is the marble made of?

Metal or stone

Stone or glass

Glass

Not enough information to tell

EXPLANATION

Answer

B) Stone or glass.

Explanation

As the marble is large, it cannot be made of metal (Restriction 2), so it must be made of stone or glass. The blue colouring and plain decoration do not tell us more about the material of the marble.

BACKGROUND INFORMATION

This question looks at **sorting** (i.e. categorising) objects based on given criteria. There are four criteria that we can use to sort the marbles by (size, colour, material, decoration), either alone or in combination.

We also need to work with **constraints**, which are restrictions or limitations on our solutions or methods. In this case, there are 5 restrictions which are applied to the marble designs. When these restrictions are combined, only a few design options remain.

A digital example of sorting with constraints is using filters to find subsets within a given data set. This can occur in spreadsheets, where a filter can be applied to omit/select certain rows. This can also occur in search results – such as from a search engine or online shopping interface – where a filter can refine the results to match desired criteria.



Party Foul



Corey Beaverton is having a party this weekend. He would like to meet new people. The party invitation includes an instruction to make four identical copies of the invitation and send them to another four people.

Corey sends out the first round of invitations to four of his friends. Each new group of invitations goes out in a round. Each beaver only sends out one set of invitations.

Corey forgets to put a limit on how many people can be invited in total.

Question

Assuming that no beaver is invited more than once, how many rounds of invitations will be sent out before more than 500 beavers in total have been invited to Corey’s party?

- 4
- 5
- 10
- 125

EXPLANATION

Answer

The correct answer is 5.

Explanation

In the table below, the number of new beavers invited to the party is shown for each round, along with the total number of beavers invited.

Invitation Round	New Beavers Invited	Total Beavers Invited
1	4	4
2	16	20
3	64	84
4	256	340
5	1024	1364
6	4096	5460

In the first round (R1), Corey **sends out** 4 invitations inviting beaver 1, 2, 3 and 4 (B1 to B4). See the illustration below. After the first round, 4 beavers (not including Corey) are invited to the party.

$1 \times 4 = 4$

[Corey x 4 invitations = newly invited beavers]

During the second round (R2) each of the four beavers sent out four more invitations, inviting 16 new beavers (B5 to B20), making a total of 20 beavers invited. See calculation below.

$4 \times 4 = 16$

[Invited beavers from R1 x 4 invitations = newly invited beavers]

$16 + 4 = 20$

[newly invited beavers + Invited beavers from R1 = total beavers invited in R2]



Party Foul – continued

In the third round (R3), the 16 beavers invited in the second round invited another 64 beavers, making the total invited beavers 84. The third round includes beavers 21 to 84.

$$16 \times 4 = 64$$

[Invited beavers from R2 x 4 invitations = newly invited beavers]

$$64 + 20 = 84$$

[newly invited beavers + Invited beavers from R2 = total beavers invited in R3]

In the fourth round (R4), those 64 beavers invite a further 256 beavers, making the total beavers invited, 340. The 265 beavers sent out invitations to beaver 85 to 340.

$$64 \times 4 = 256$$

[Invited beavers from R3 x 4 invitations = newly invited beavers]

$$256 + 84 = 340$$

[newly invited beavers + Invited beavers from R3 = total beavers invited in R4]

The fifth round (R5) sees the 256 beavers invited in R4 send out invitations to a further 1024 beavers. This round includes beaver number 341 to 1364. Therefore 500 beavers would have been reached after sending out the fifth round of invitations.

$$256 \times 4 = 1024$$

[Invited beavers from R4 x 4 invitations = newly invited beavers]

$$1024 + 340 = 1364$$

[newly invited beavers + Invited beavers from R4 = total beavers invited in R5]

BACKGROUND INFORMATION

It is often surprising how a lot of small numbers may combine together to one very large number, especially when multiplication is involved.

Hearing that something was multiplied fourfold two times in a row, may easily suggest the number 8 to you (for eight is two times four, is it not?) while in fact multiplying a quantity by 4 and then again by 4 is the same as multiplying the original by 16. (Which is 4^2 . We call 4 the base and 2 the exponent. The exponent tells us how many times a number is multiplied by itself.)

When something grows by *multiplying* it over and over by the same number (instead of *adding* the same number to it again and again) we say that it grows exponentially. Even starting small, exponential growth can quickly lead to huge numbers, as you have noticed in this task.

In informatics exponential growth is encountered often. Sometimes it is a bad thing: a computer virus, meant to harm your computer, may work by spreading copies of itself by e-mail to other computers – a bit like real viruses spread from person to person. Even if a virus on a single computer can copy itself only to a small number of other computers every day (say 4), then the number of infected computers after 4 days is already more than 500!

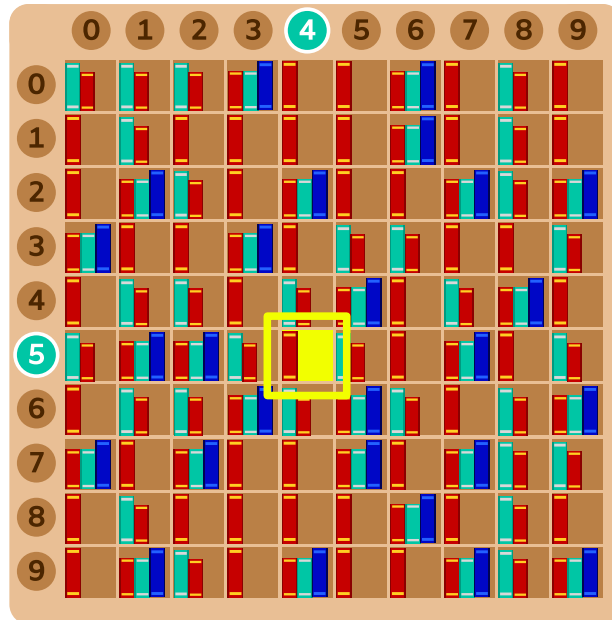
Computer programmers can also use exponential growth to their advantage. If for instance you have to search for something among a million items and at each step you can throw away half of the possibilities, then after just 20 steps you will have found what you are looking for. This is a kind of exponential growth ‘in reverse’: discarding half of the possibilities is the same as multiplying the number of possibilities by $\frac{1}{2}$, and $\frac{1}{2}$ times $\frac{1}{2}$ times $\frac{1}{2}$ times $\frac{1}{2}$... quickly becomes a really really small number.



Hashing

Harish accompanied his friend Sue to the Bebras Public Library. The library only had one huge bookcase made of many bays arranged in rows and columns. Sue wanted to borrow the book “Constructing Dams For Beginners”.

When they arrived, Sue went straight to the bookcase and pulled out the correct book.



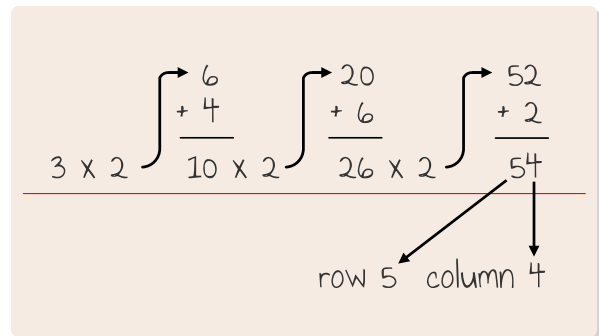
“How did you know where the book was?” Harish asked surprised. Sue smiled and showed him two pieces of paper:

a	b	c	d	e	f	g	h	i	j	k	l	m
1	2	3	4	5	6	7	8	9	10	11	12	13
n	o	p	q	r	s	t	u	v	w	x	y	z
14	15	16	17	18	19	20	21	22	23	24	25	26

Constructing Dams For Beginners

↓ ↓ ↓ ↓

3 4 6 2



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

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

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

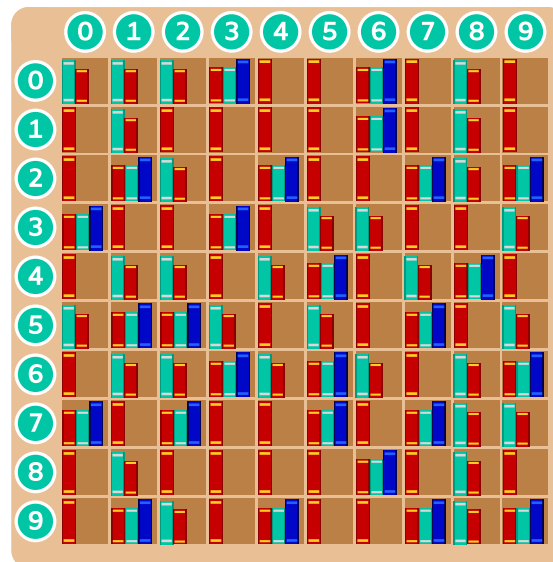


Hashing - continued

Question

In which bay can Harish find the book “How To Avoid Falling Trees”?

Select the correct numbers on the sides of the bookcase to the corresponding bay.



EXPLANATION

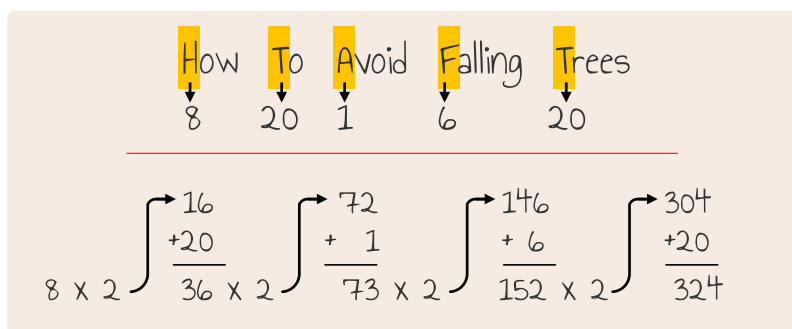
Answer

There are two correct answers for this task:

1. The book is in the bay that is in row 2 and column 4.
2. The book is in the bay that is in row 6 and column 6.

Explanation

The first acceptable answer is from the perspective of a developer who has had the method explained to them from a non-programmer. The speech marks show that this is a conversational description. The developer concludes that the speaker is giving an example and infers that for books with titles that have more than four words the calculation is:



Taking the last two digits of **324** gives the row and column.

The second acceptable answer is from the perspective of a computer working through the algorithm provided, without interpreting it in any way. The calculation continues as above until **146** is calculated but the description provided by Sue says to add the number of the last letter at this stage, which is 20, giving **166**.

Continued on next page



Hashing - continued

BACKGROUND INFORMATION

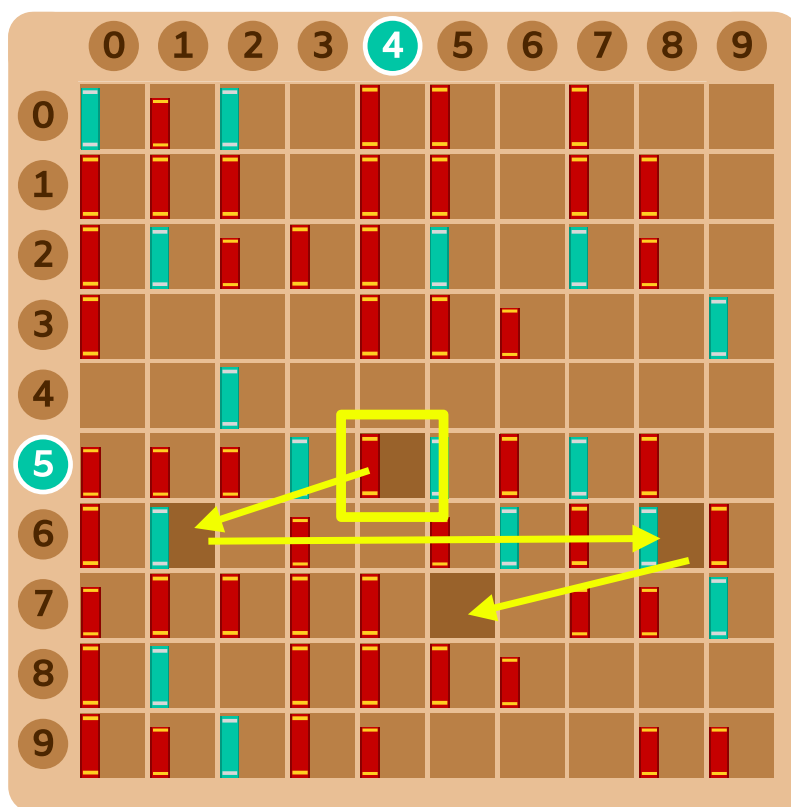
Behind the algorithm used by the Bebras Public Library is a concept called “hashing”.

If no system is used to sort the bookshelf, a (linear) search of every book would be necessary to find the book. On average you would have to check 50% of all books to find the desired book. Imagine searching the Library of Alexandria (~100,000 books), the Library of Congress (~38,000,000 books) or even just your local or your school’s library using this method - it could take days to find a single book!

This problem does not only exist for libraries. Big pharmacies also need to have a system to store and retrieve their medicine. In the last few years more and more pharmacies are adopting automated storing systems. For pharmacies a systematic order, sorting by type of medicine for example, doesn’t matter. Instead they look for an even distribution on the shelf.

Here the concept of “hashing” comes into play. A *hash value* is a value calculated via a hash function from properties of an item. In the case of this task the title of a book is transformed into two digits that make up the row and the column of a bay on the shelf. Of course, different books can end up with the same hash value like the books *Tree Bark Gourmet Guide* and *Tasty Trees to Gnaw On*. There are different ways of dealing with such a conflict.

One way is to simply put several items into the same place like in a bay on the shelf. If it’s not possible to store multiple items in the same place, then the next empty place is chosen, or an empty place n places away can be chosen instead to create a more even distribution. If that place is also not empty, the place a further n places away is checked. This is repeated until an empty place is found. To eventually fill up all places, n is chosen to be coprime to the total number of places - in the example below, 7 is chosen as there are 100 bays on the shelf and 100 and 7 do not share any common divisors except for 1.








Error Detection

In a Bebravian city, electricity is produced by wind turbines. The electricity then gets carried to the houses through the network shown below.

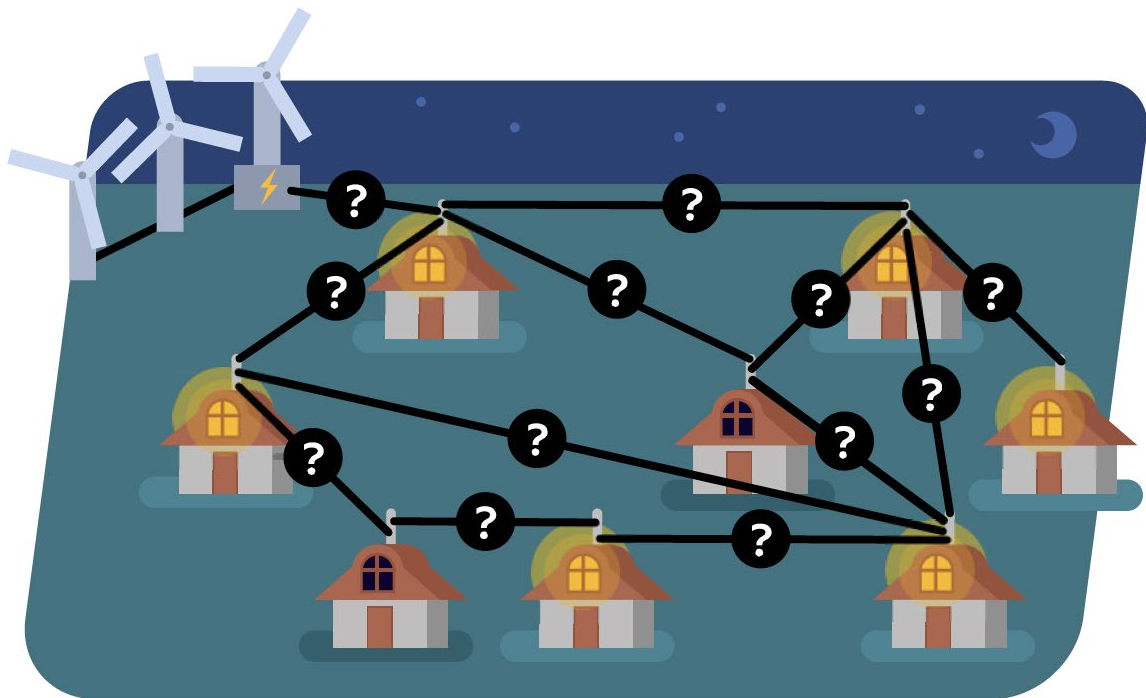
Electricity can be carried from house to house in any direction. However, some links are faulty - the two houses with the lights off don't have electricity anymore! All the others houses do.

The map allows each connection to be labelled by clicking on the question marks. They toggle through these three symbols:

-  The connection is known to be faulty.
-  The connection is known to be working.
-  It is not possible to tell if the connection is working or faulty.

Question

In what state are each of the connections?



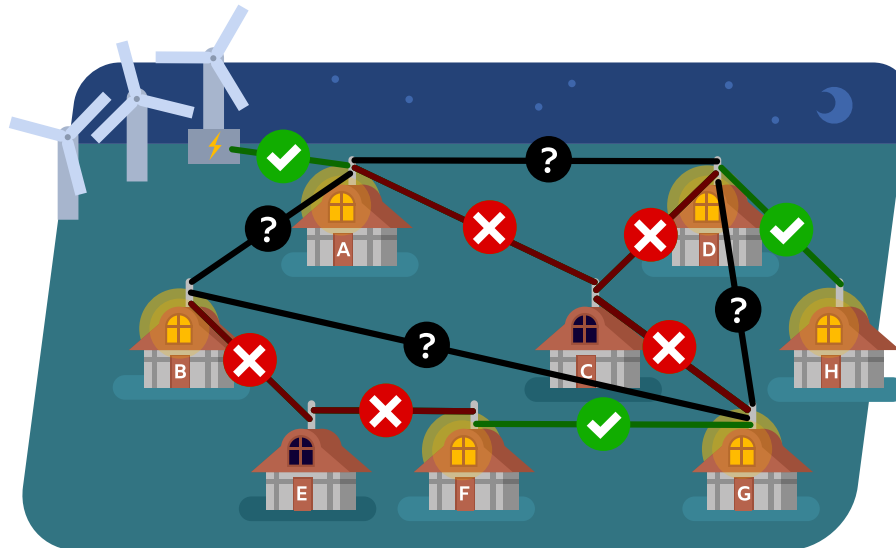


Error Detection – continued

EXPLANATION

Answer

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



Explanation

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

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

The remaining houses, B, G, and D, are multiply connected to house A. For instance, B can get its electricity directly from A, but it could also get it from G if the link to A was faulty. The same can be said about D. Finally, G can get its electricity either from B or from D. One of the links in the $A - B - G - D - A$ cycle could thus be faulty and these four houses would still get electricity. It isn't possible to determine the state of any of these connections from the current information.

BACKGROUND INFORMATION

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

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

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



Meeting Race

Two friends need to meet urgently. They are at opposite ends of a park - see the map below.
The friends need to end up on the same square to meet. They both travel according to the following rules:

- They cannot travel diagonally.
- They cannot travel over water.
- Walking from one square to a neighbouring square takes exactly one minute.
- If they reach a bike or car they can use it to travel faster:
 - 2 squares in one minute with a bike
 - 5 squares in one minute with a car.

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



Question

What is the minimum number of minutes they need to end up on the same square?



Meeting Race – continued

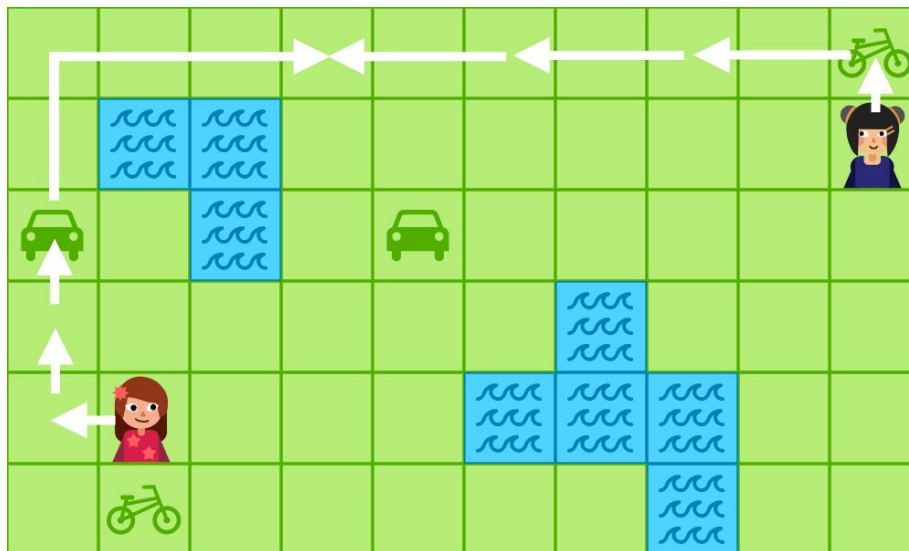
EXPLANATION

Answer

4.

Explanation

The meeting can take place in 4 minutes using the route shown below:



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

To see why 3 minutes are not sufficient (and therefore 4 minutes is the minimum time needed), one can reason as follows:

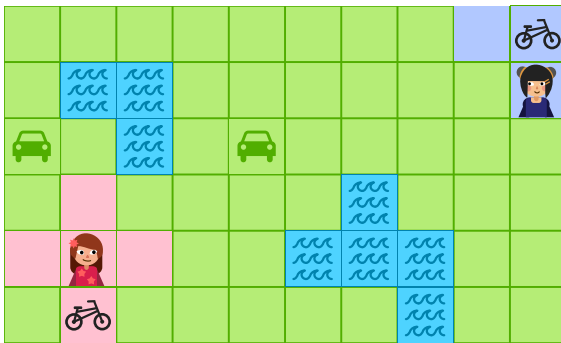
- Although in 3 minutes the friend in the bottom left can reach the car on the left, there is no time to drive it anywhere. And that position cannot be reached in 3 minutes by the other friend. The car in the middle cannot be reached within 3 minutes. So the cars can be ignored if trying to find a route in 3 minutes.
- The two friends are more than 5 minutes away from each other on foot, so they need a bike. In fact, both need a bike because they are separated by more than 9 positions. But finding that bike costs one minute and with only two minutes left they cannot reach each other, even by bike.



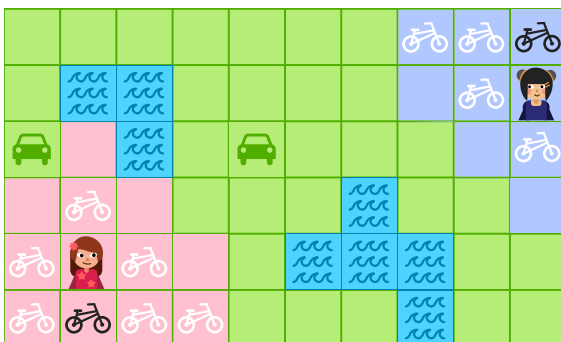
Meeting Race - continued

BACKGROUND INFORMATION

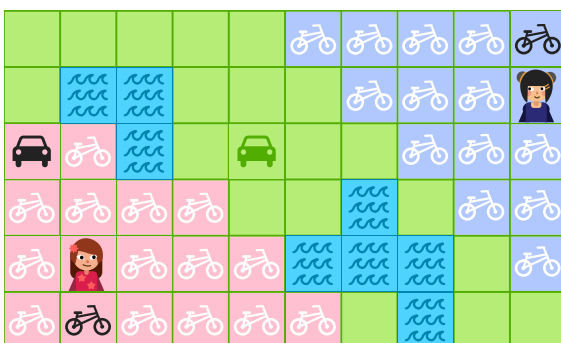
An common way for many people to solve this task is to simply stumble onto a short path, or to try out dozens of different possibilities and choose the one with the shortest time. A computer program that is designed for this kind of task would use a systematic approach, most probably using an algorithm called breadth-first search. For this task, this would be done as follows:



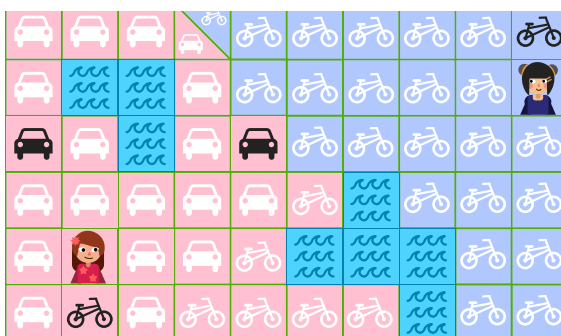
1. Mark all squares on the map that can be reached by either friend in one minute.



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



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



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



Meeting Race – continued

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

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

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

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

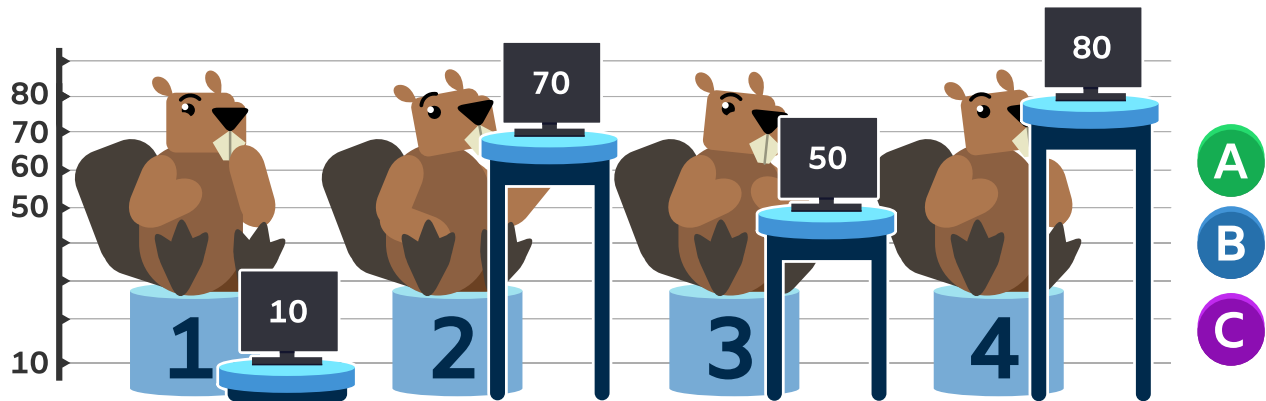
When a problem becomes too complex, going through all possible solutions to find the best one will take too long even for a fast computer. And in practice it is often sufficient to find a very good answer even if it is not the best possible. (If you can reach your destination in 78 minutes you are probably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

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



Desk Trouble

In the computing classroom, students adjust the heights of their desks by using an electrical system of three buttons. The desks can't be lowered below 0 units, but can be raised to any height. The recommended height of the students' desks is **60 units**.



After the control buttons got wet, this is how they function:

- The "A" button raises desktops 1, 2 and 3 by 10 units, each time it is pressed.
- The "B" button lowers desktops 2, 3 and 4 by 10 units, each time it is pressed.
- The "C" button raises desktops 1, 3 and 4 by 10 units, each time it is pressed.

Question:

Which of the following combination of button presses would place all the desks at the recommended height of 60 units?

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

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

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

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



Desk Trouble – continued

EXPLANATION

Answer:

Press “A” 3 times, “B” 4 times and “C” 2 times.

Explanation:

By methodically checking each answer below, it can be determined that only the option in bold gives the correct answer.

1. **Press “A” 3 times, “B” 4 times and “C” 2 times.**
2. Press “A” 4 times, “B” 5 times and “C” 1 time.
3. Press “A” 5 times, “B” 1 time and “C” 0 times.
4. Press “A” 2 times, “B” 4 times and “C” 6 times.

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

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

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

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

BACKGROUND INFORMATION

In programming, we look for computers to solve specific tasks using the instructions we give them. Often, we need computers that perform particular actions in some cases, and different actions in other cases. Conditional sentences allow the computer to choose which processes it needs to act on, based on different parameters given in the program.

In this task the answer can be considered as the program, written by the programmer, to achieve the desired goal. A more useful program would be one that is able to find the sequence of events needed to achieve a variety of goals from a variety of starting points. This type of program would likely have a greater level of complexity.

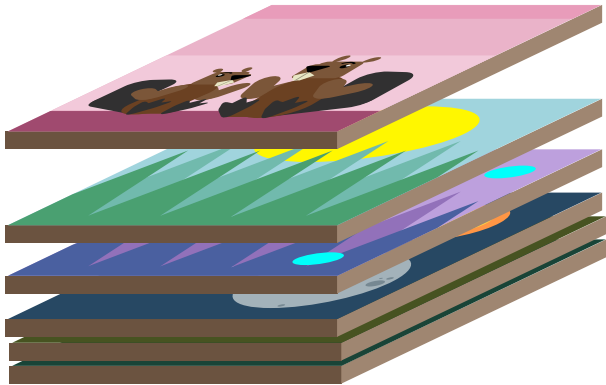


Art Theft

TransArt is a company that transports paintings. Paintings are brought to a store for inspection, and then couriers transport them to their final destination. When a painting arrives at the store, it is put at the top of the stack of paintings already there. When a courier takes a painting to deliver it, they take the painting that is at the top of the stack.

For security reasons, TransArt keeps good records of all paintings coming in and out. The records of a particular day are shown below:

Paintings brought in the store		Times paintings were taken from the store	
Time	Painting	Time	Courier
11:40	Beavers on the Grass	12:25	A
12:15	Happy Beaver	13:35	C
12:55	Sun and Moon	14:35	A
13:30	Enchanted Forest	14:40	B
14:18	Oak and Birch	15:20	C
15:10	Swampy Romance	15:35	D



However, that evening TransArt was told that “Sun and Moon” never reached the museum that was supposed to receive it. The courier who took it from the store must have stolen it!

Question

Who took “Sun and Moon”?

A

B

C

D

EXPLANATION

Answer

B.

Explanation:

There are two important types of events:

- 1. Somebody puts a painting on the stack.
- 2. Somebody takes a painting from the top of the stack.

From the tables in the task, we can create a new table that displays the events and the resulting state of the stack, sorted by time.



Art Theft – continued

Time	Event	Paintings on the stack
11:40	Arrival of Beavers on the Grass	Beavers on the Grass
12:15	Arrival of Happy Beaver	Happy Beaver, Beavers on the Grass
12:25	A takes Happy Beaver	Beavers on the Grass
12:55	Arrival of Sun and Moon	Sun and Moon, Beavers on the Grass
13:30	Arrival of Enchanted Forest	Enchanted Forest, Sun and Moon, Beavers on the Grass
13:35	C takes Enchanted Forest	Sun and Moon, Beavers on the Grass
14:18	Arrival of Oak and Birch	Oak and Birch, Sun and Moon, Beavers on the Grass
14:35	A takes Oak and Birch	Sun and Moon, Beavers on the Grass
14:40	B takes Sun and Moon	Beavers on the Grass

... and we can stop here as we have reached the critical information in the records that says courier B has taken “Sun and Moon”.

BACKGROUND INFORMATION

There are three computer science ideas that appear in this task.

One is the concept of a **stack**. A stack is not only a stack of paintings, but also a data structure, organised so that the last element that is put on stack is the first one to be popped from it (“Last In – First Out” or *LIFO* in short).

The second is **merging**. To reach the solution, we had to take two sorted lists (events sorted by timestamps) and merge them into one sorted list – the one shown here in the solution. This step is the base of one of the fastest algorithms for sorting data, the merge sort.

Finally, the entire process can be considered as the **execution** of a program. The theft of the painting is like an event that causes a running computer program to stop or “crash”. This is called an *exception*. To find the cause of an exception (which could be thought of as either the misbehaving courier or the misbehaving lines in the program), we need to follow the program execution until we reach the point where it crashes. This is called *tracing*. Then a programmer would try to find a way to handle the exception in order to prevent a crash of the program.

Bebras Challenge 2022 Round 1

Years 9+10

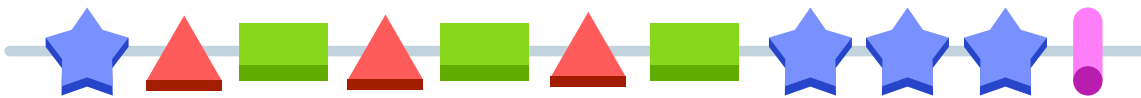


Necklaces Instruction

You like to design necklaces from beads with various shapes, and you would like to easily share your design with your friends using a compact representation. Each shape is described with a single bead letter (S for star, T for triangle, R for rectangle and L for line). Instead of just writing down the sequence of beads in the necklace, you use the following rules:

- if there are several identical beads following each other, just write the number of beads followed by the bead letter
- if there is a repeating sequence of beads, just write the number of repetitions followed by the repeated sequence in parentheses
- otherwise, just write the bead letter.

For example, for the following necklace:



one possible description is S3(TR)3SL, which has a length of 9 symbols.

Question:

How many symbols are there in the shortest representation for the following necklace?
(a symbol is either a digit, a letter or a parenthesis)



EXPLANATION

Answer

The correct answer is 13 (option B).

Explanation

There are many possible ways to represent this necklace depending on whether you use star-triangle or triangle-star as the repeated pattern. All of them give a different representation, with the first two results below being the shortest:

- S2RT3S3(ST)4L with 13 symbols,
- SRRT4STSTST4L with 13 symbols,
- and S2RT4S2(TS)T4L with 14 symbols.

One can see that some representations are more efficient than others. For example, the second option writes out the repeating triangle-star pattern as TSTS for a total of 4 symbols, while the bottom option writes it out as 2(TS) for a total of 5 symbols. Therefore, comparing these options and optimising for the shortest symbol length is a valid approach to solving the question.

BACKGROUND INFORMATION

This task is related to the *data compression*. The idea of this field is to design techniques to make it possible to represent and store data using the smallest amount memory, while still being able to recover the original data at any time. All of these techniques use similar approaches, which is to identify redundancy in order to represent it in a more compact way. This is exactly what happens in this task when the repetitions of beads can be represented in a shorter way compared to explicitly writing out the repeated beads or sequence of beads - TSTSTS may be more efficiently written as 3(TS), for example.



Chez Connie

The takeaway restaurant “Chez Connie” is always busy at lunchtime because it sells three delicious menu items:



- Ice cream, which can be prepared in 3 minutes;
- Crêpe, which can be prepared in 8 minutes;
- Pizza, which can be prepared in 12 minutes;

There are three queuing windows A, B, and C. All three menu items can be prepared at any window. Connie wants to organize the orders so that the clients get served as quickly as possible. She notes the incoming orders on numbered pieces of paper so that she knows which order arrived first.

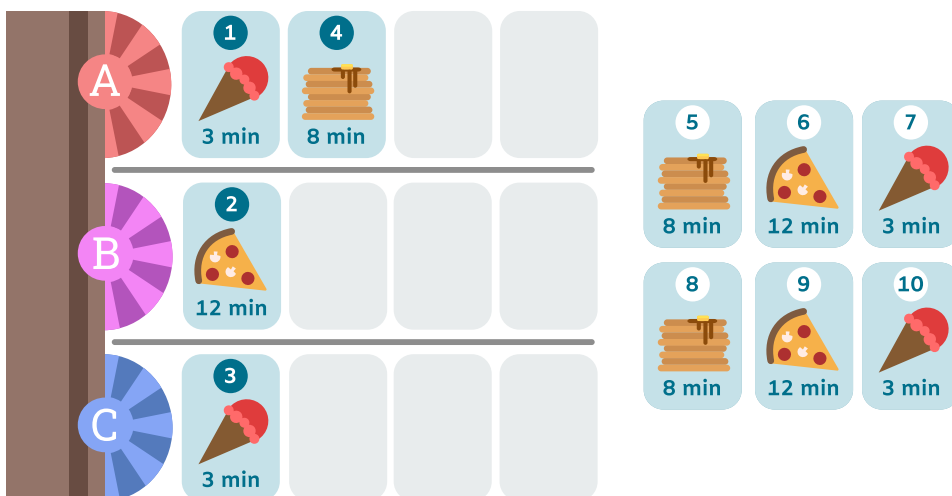
This is her first order today:



Then Connie distributes the orders to the windows A, B, and C. She always assigns the next order to the first available window. If two or more windows become available at the same time, the orders are assigned in the windows' alphabetical order - in other words, window A first, then B, then C.

Question

Connie has already distributed the first four orders. Can you distribute the next six orders?

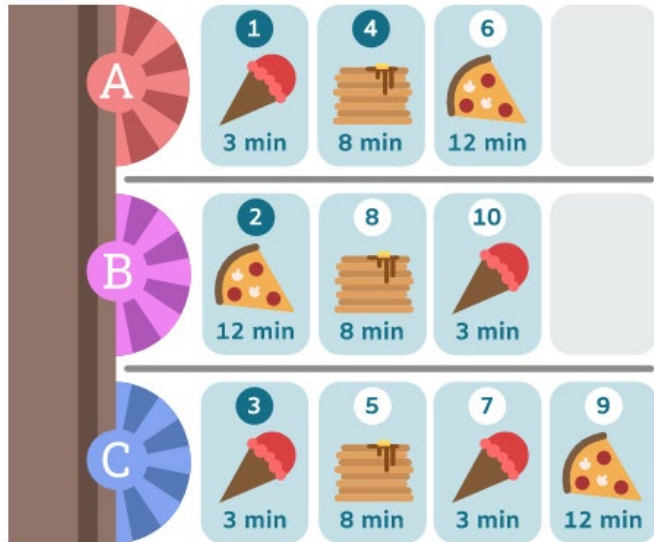




Chez Connie – continued

EXPLANATION

The correct solution is:



To distribute the orders to the different windows, Connie has to compare the time needed to prepare the items that are already in each window's queue.

When we start allocating orders, the windows already have the following preparation time needed:

- Window A: 11 minutes
- Window B: 12 minutes
- Window C: 3 minutes

The next item (5) will be allocated to Window C, as it has the shortest preparation time at this point. This makes the new preparation times:

- Window A: 11 minutes
- Window B: 12 minutes
- Window C: 11 minutes

Since we have the same time needed for preparation at both Window A and Window C, we assign the next two orders in alphabetical order. Since we know that assigning anything to A will make C the shortest time needed, we can go ahead and assign orders 6 and 7 at once, assigning #6 to Window A, and #7 to Window C. This makes the new preparation times:

- Window A: 23 minutes
- Window B: 12 minutes
- Window C: 14 minutes

At this point, Window B has the shortest preparation time needed, so we can assign the next order (8) to that window. This makes the new preparation times:

- Window A: 23 minutes
- Window B: 20 minutes
- Window C: 14 minutes

The next order (9) will be allocated to Window C, since it has the shortest preparation time needed by far. This makes the new preparation times:

- Window A: 23 minutes
- Window B: 20 minutes
- Window C: 26 minutes

This leaves us with the last order (10) - the window with the shortest preparation time at this point is Window B, so order 10 will be allocated to Window B.

Continued on next page

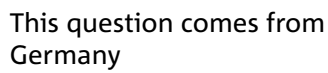


Chez Connie – continued

BACKGROUND INFORMATION

In modern computers, there are several *processors*, or several *processor cores*, which can carry out basic operations (like addition or multiplication) mostly independently of each other. In this task, the analogy with the three windows is direct: the three windows can each work on an order independently of the others.

Rather than preparing food like the windows in this question, processors execute *programs*: sequences of instructions of varying lengths designed to solve a given task. Often, hundreds of these programs (which are referred to as processes here) are waiting for a processor to become free and to start executing them. The attribution of processes to processors is called *scheduling* and is handled by the computer's operating system. In order to solve this task, we must play the role of the operating system ourselves - distributing orders to windows is similar, albeit in a simplified way, to assigning processes to processors at a given time.



Years 11+12



A stylized illustration of a bird, possibly a sparrow or finch, perched on a dark grey pedestal. The bird is facing left, with its head slightly turned. It has a dark grey body, a lighter grey wing, and a small dark beak. The pedestal is a simple, dark grey rectangular block with a slightly wider base and top.

The robotic lawn mower moves according to these rules:

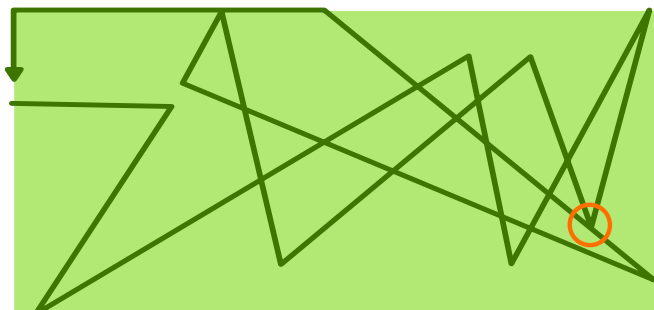
- ## Question

Continued on next page

Detective Lawn Mower – cont'd

EXPLANATION

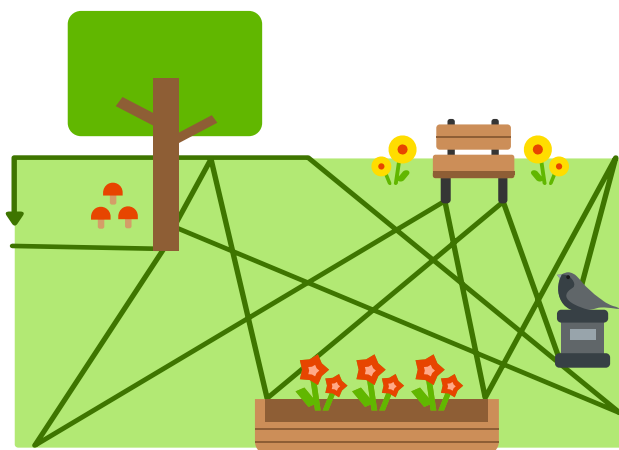
Answer



Explanation

The missing statue was located in the bottom right-hand corner of the park as shown below. This is where the robot mower's behaviour changed over the course of the night, as it collided with an object during its first pass but did not collide with an object on its second pass.

We can create a detailed description of the map to see that this is the case:



Shortly after the robot started mowing it changed its direction in the middle of the lawn. This shows that it had hit an obstacle, the tree. The robot continued in a new direction until it reached the boundary of the park where it changed direction and continued until it hit the park bench. The robot went on to collide with the flower bed, park boundary, the statue, park bench (again), flower bed (again), park boundary, the tree (again), and then eventually reached the boundary. The robot changed direction and again moved through the area where there had previously been an obstacle (the statue) but *did not change direction*, instead continuing straight to the top boundary, and then back to its charging station. The

map shows us that the robot passed through the bottom right-hand area of the park without changing direction whereas earlier in the night it had hit something there and changed direction.

BACKGROUND INFORMATION

The challenge of this task is to find a relationship between the obstacles within the park and the robot mower's behaviour. The behaviour of the mower is controlled by the rules and impacted by its environment. Noticing where two (allegedly) identical situations yielded different behaviours by the mower is key to finding the solution. This kind of reasoning is part of computational thinking and incorporates *pattern recognition*, *data representation*, and *data interpretation*.

The robotic lawn mower in the task is controlled by a very simple program. The software of commercial products might be smarter — some robots can move around objects on the lawn instead of just randomly changing direction. Some very smart robots create a digital map of the lawn and move systematically instead of randomly. Robotic lawn mowers are very practical but they also have some drawbacks. For example, they can injure or kill smaller animals. They should therefore only be allowed to operate during the day under human supervision.



Viewer Numbers

Betty and Bobby Beaver publish videos on a popular online video service. The service informs them each month how many viewers their videos have attracted. Betty and Bobby feel uncomfortable about outsiders seeing this information, so they receive the number of viewers as a secret message.

The message consists of cat (🐱) and dog (🐶) symbols and can be converted into a number with the help of a code table that tells which symbol combination corresponds to which digit.

Below is the code table used by Betty and Bobby:

0: 🐱🐶🐱🐶	1: 🐱🐶🐱	2: 🐶🐱🐶🐶	3: 🐶🐱🐱🐱	4: 🐶🐱🐱
5: 🐶🐶🐱🐶	6: 🐶🐶🐶	7: 🐶🐶🐶🐱	8: 🐱🐶🐶🐱	9: 🐱🐱🐶🐶

Betty and Bobby receive the message below, telling them the number of viewers they had last month:



Question

How many viewers did Betty and Bobby's videos get last month?

Answer:

EXPLANATION

Answer

The answer is 417511, as shown by the below method of partitioning the complete message into individual digit codes:

4= 🐶🐱🐱	1= 🐱🐶🐱	7= 🐶🐶🐶🐱	5= 🐶🐶🐱🐶	1= 🐱🐶🐱	1= 🐱🐶🐱
-----------	-----------	------------	------------	-----------	-----------

Explanation

The answer is most easily found by considering the message *right to left*. Moving this way, there is only one candidate for the correct code at each step of the process. This is because the digit codes in the task are “suffix-free” - This means that none of the digit codes can be found at the end of other digit codes. We can also work left to right to find the answer, but the digit codes are not “prefix-free” - Some of the digit codes used are found at the start of other digit codes.

For example, the code for 1 (🐱🐶🐱) can be found at the start of the code for 0 (🐶🐱🐱🐶).

Continued on next page



Viewer Numbers – continued

This means that it may take several attempts to find an answer, as using some of the code patterns will mean reaching a point where the next symbols do not make up a digit code.

For example, one could first try to decode the first digit as a 3, but the next symbols do not allow a match with any of the digit codes.

BACKGROUND INFORMATION

The digit codes in the task are essentially *binary codes*, where each symbol is either a cat or a dog. This is analogous to binary codes used in computing where each symbol is either 0 or 1. All data processed by digital computers is represented as binary codes. For example each character, including the digits 0-9, is represented by a binary sequence so that it can be stored on a computer.

The digit codes in this task were also an example of a *variable-length* character encoding. This refers to a kind of encoding where the codes for different characters may have different lengths. For example, the code for digit 0 consists of 4 symbols whereas the code for digit 1 consists of 3 symbols. Variable-length codes are used widely in real applications. For example, compression algorithms (zip etc.) use variable-length codes, and the very common UTF-8 character encoding is in itself a variable-length (or variable-width) code.

As the digit codes in the task were “suffix-free”, decoding the message was more simple working from right to left. In practice variable-length codes are usually inspected from left to right (or beginning towards end) and are hence designed to be *prefix-free*. Such codes are commonly called *prefix codes*.



Encrypted Path

Beaver Bela has described the path that she and her friends take from school (S) to her house (H). The path is shown in a square grid and described by using codes. The path they take goes past her friends' houses, so that they can go home as well. Her friends' houses are marked on the grid with numbers from 1 to 8. The houses are shown in the code with an asterisk "**".

The letters used in the code are as follows:

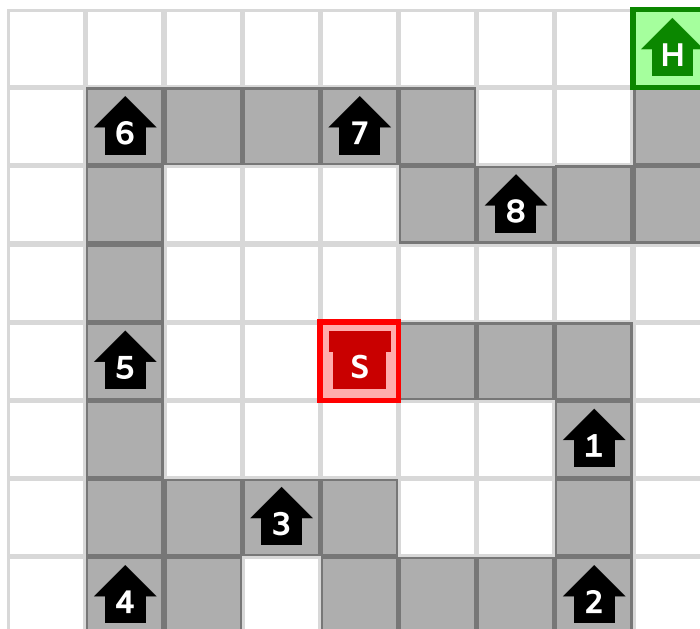
U - is up

D - is down

L - is left

R - is right

The number of squares to move is written first, followed by the letter that determines the direction. For example, "4D" means "four squares down"



Question

Which code represents the drawn path?

3R1D*2D*3L1U1L*1L1D1L*3U*3U*3R*1R1D1R*2R2U

3R1D2D*3L1U1L*1L1D1L*3U*3U*3R1R*1D1R*2R2U

2R1D*2D*3L1U1L*1L3D1L*3U*3U*3R*1R1D1R*2R5U

3R*1D2D*3L1U1L*2L1D1L*3U*3U*3R*1R1D1R*2R2U



Encrypted Path – continued

EXPLANATION

Answer

The correct answer is `3R1D*2D*3L1U1L*1L1D1L*3U*3U*3R*1R1D1R*2R2U` (option A).

Explanation:

We can see that each answer starts with a different sequence before the first asterisk, which denotes the house. We can use this section alone to rule out the incorrect answers without further investigation.

`3R1D2D*3L1U1L*1L1D1L*3U*3U*3R1R*1D1R*2R2U` has 3 squares right, 1 square down and 2 squares down before the first asterisk. This is incorrect because this path will skip house 1 completely and stop for the first time at house 2.

`2R1D*2D*3L1U1L*1L3D1L*3U*3U*3R*1R1D1R*2R5U` has 2 squares right and 1 square down before the first asterisk. This path will stop for the first time on the square next to house 1. This is incorrect because asterisks are intended to indicate houses.

`3R*1D2D*3L1U1L*2L1D1L*3U*3U*3R*1R1D1R*2R2U` has 3 square right before the first asterisk. This path will stop for the first time on the square directly above house 1. This is incorrect because asterisks are intended to indicate houses.

BACKGROUND INFORMATION



Debugging programs is one of the most important jobs for programmers. One method is to insert breakpoints into the program and check that there have been no errors up to this point.

In the proposed problem, you need to compare four long codes. However, there are asterisks inside the code that allow you to break the code apart, describing the movement in distinct parts - movement from one house to another. If you compare the codes in such parts, you will quickly find erroneous codes that do not match up with the given route.























Fruit Stack

A family of four prepares breakfast for the next day. They pile up four boxes, each filled with a different fruit:

apple  , pear  , orange  , or strawberry  .

As they are sleepy in the morning, they all just grab the box off the top of the pile. They do not know in which exact order they will get to the pile of boxes, but the mother always gets there before the daughter, and the father is always last.

Each of the four like and dislike different fruits. Fruits they like are marked below with a tick and fruits they dislike are marked with a cross:

				
Father				
Mother				
Daughter				
Son				

Question

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





Fruit Stack – continued

EXPLANATION

Answer

There is only one correct solution:



Explanation

We first look at what the father wants. He only likes oranges and will be the last one to reach the boxes. Therefore, we need to put oranges into the box at the bottom so he is guaranteed to get a fruit he likes.

Because we know that the mother will be taking her box before the daughter gets up, the mother is either the first or the second one to take a box.

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

To summarize, the following three arrival orders are possible:

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

We see that the second one to get up can be either the son, daughter, or mother. This means that the fruit in the second box from top must be something that they all like.

Looking at the table of options, the only option that all three enjoy is the apple. (Second row in the table below.)







Fruit Stack – continued

So we are left with two choices for the topmost box - pear and strawberry. This box can be taken by either the mother or the son. The mother does not like pear. Therefore we have to put strawberry into the first box, which the son also likes. (First row in the table below.)

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

In summary, we have the following options for the order of the family members arriving, which gives us the order of fruits shown below.

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

BACKGROUND INFORMATION

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

The task asks to find a way of sorting of the fruits which will work under multiple possible conditions. But there are some constraints, and not all possible orders of family members can occur. Solving such constraint problems can be very difficult. Often the best idea to do so is writing and using a computer program to solve the problem.

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

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

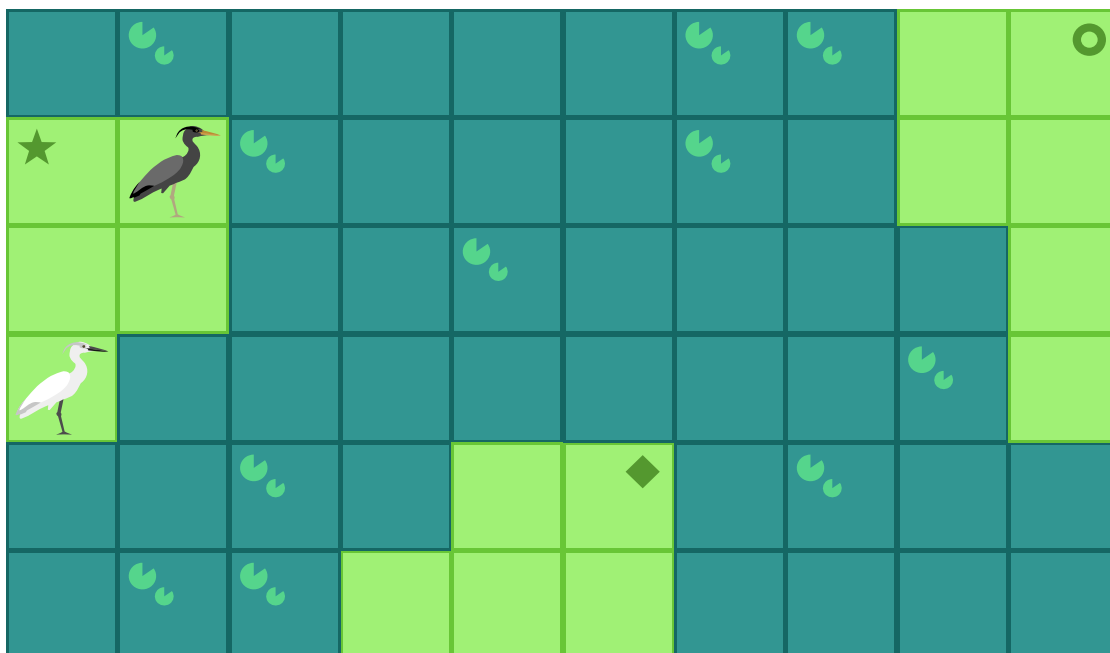


Bird Migration

Three islands (Star ★, Donut ○ and Diamond ◆) are separated by marsh. There are two birds, a black heron and a white ibis, that are currently located at different points on Star Island. Now that winter has come, the birds want to migrate to Donut Island to find a warmer area to build their nests.

The birds move according to the following rules:

- The black heron can fly over the marsh at a rate of 2 blocks per hour. Once it has flown for 4 blocks, it must immediately land on an adjacent square of land. It cannot move again until it has rested on this square for 1 hour.
- The white ibis can fly over the marsh at a rate of 4 blocks per hour. Once it has flown for 4 blocks, it must immediately land on an adjacent square of land. It cannot move again until it has rested on this square for 2 hours.
- Both birds can also walk on land at a rate of 1 block per hour.
- Each bird can only move left/right/up/down on the map shown below – they cannot move diagonally.



Question

Which bird can migrate from Star Island to Donut Island the fastest, and what is the difference between the fastest migration times between the birds?

The black heron is faster by 1 hour.

The white ibis is faster by 1 hour.

The black heron is faster by 2 hours.

The white ibis is faster by 2 hours.



Bird Migration – continued

EXPLANATION

Answer

The white ibis is faster by 2 hours.

Explanation

We must first find the fastest path for each bird, and then compare the times to find the faster of the two paths.

The white ibis first flies 4 blocks towards the right from Star Island to Diamond Island, which takes 1 hour. After resting for 2 hours, it walks 1 block on land towards the right, which takes 1 hour. Finally, it flies 4 blocks from Diamond Island to Donut Island, which takes 1 hour. The overall time for the white ibis is $1 + 2 + 1 + 1 = 5$ hours.

The black heron first walks 1 block downward on land, which takes 1 hour. From here, it flies 4 blocks to Diamond Island, which takes 2 hours. After resting for 1 hour, the black heron walks 1 block towards the right, which takes 1 hour. Finally, it flies 4 blocks to Donut Island, which takes 2 hours. The overall time for the black heron is $1 + 2 + 1 + 1 + 2 = 7$ hours.

Thus, the white ibis arrives at Donut Island 2 hours faster than the black heron.

Note that for both birds the flight segments over the sea are the longest that they can fly without a rest. This means that we can't replace any of the (slower) walking segments with (faster) flying, as then the flying range will not be sufficient to reach the next island. This in turn means that the total travel times computed above are optimal.

BACKGROUND INFORMATION

This question looks at optimisation, which is the process of finding the most efficient solution to a given problem. In this question, we need to find the most straightforward path from one island to the next for each bird.

We also need to work with constraints, which are restrictions or limitations on our solutions or methods. In this question, each bird has to take a period of rest after travelling a certain distance, which limits how far each bird can travel continuously.

A physical example is the task of building the largest box possible using a given amount of material, such as sheet metal. We want to maximise the volume of the box (optimisation), but we are limited by the amount of material we have to work with (constraint).

A technological example is the task of a computer deciding how to allocate its memory to various tasks at the same time. The computer has a set amount of memory (constraint), however it may want to allocate memory across tasks in a way that improves performance for the user (optimisation).

Combining optimisation and constraints allows us to make the best use of limited resources, whether that be conserving energy in the case of the migrating birds; sheet metal in the case of the largest box problem; or memory in the case of the computer.



Ada's Marble Machine

Ada the engineer has been asked to create a sorting machine to sort marbles based on the following aspects:

- size (small or large)
- colour (red or yellow)
- material (stone or metal)
- decoration (glitter or mosaic).

Ada knows that marble designs have the following restrictions:

1. each marble can only be of one size, one colour, one material, and one decoration
2. marbles made of metal cannot be large-sized
3. marbles made of stone cannot be red
4. the glitter decoration cannot be applied to large marbles
5. the mosaic decoration cannot be applied to red marbles

Question

What is the largest possible number of uniquely designed marbles?

4

6

8

16

EXPLANATION

Answer

The answer is 6!

Explanation

One way to work this answer out is to examine all combinations of the different marble traits, and eliminate the ones that are not possible. With the 4 traits we are given, and applying the first restriction, there are a total of 16 possible types of marble.

When we apply the second restriction, we eliminate 4 marbles:

- Large Red Metal Glitter
- Large Red Metal Mosaic
- Large Yellow Metal Glitter
- Large Yellow Metal Mosaic

This leaves us with 12 uniquely designed marbles remaining.

Applying the third restriction, we eliminate the another 4 marbles:

- Small Red Stone Glitter
- Small Red Stone Mosaic
- Large Red Stone Glitter
- Large Red Stone Mosaic

This leaves us with 8 uniquely designed marbles remaining.



Ada's Marble Machine – cont'd

Applying the fourth restriction, we eliminate the following marbles:

- Large Red Stone Glitter
- Large Red Metal Glitter
- Large Yellow Stone Glitter
- Large Yellow Metal Glitter

This time, however, we have already eliminated three of these options with previous restrictions, and the only one remaining to be eliminated by this restriction is Large Yellow Stone Glitter. This leaves us with 7 options remaining.

Applying the fifth restriction, we eliminate the following options:

- Small Red Stone Mosaic
- Small Red Metal Mosaic
- Large Red Stone Mosaic
- Large Red Metal Mosaic

Once again, we have already eliminated 3 of these options with previous restrictions, leaving only 1 remaining to be eliminated. Small Red Metal Mosaic is eliminated by this restriction, leaving us with 6 potential marble designs.

BACKGROUND INFORMATION

This question looks at sorting, which is the process of categorising objects based on given criteria. There are four criteria that we can use to sort the marbles by (size, colour, material, decoration), either alone or in combination.

We also need to work with constraints, which are restrictions or limitations on our solutions or methods. In this case, there are 5 restrictions which are applied to the marble designs. When these restrictions are combined, not all design options are possible.

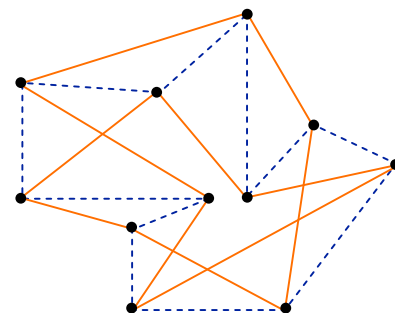
A digital example of sorting with constraints is using filters to find subsets within a given data set. This can occur in spreadsheets, where a filter can be applied to omit/select certain rows. This can also occur in search results – such as from a search engine or online shopping interface – where a filter can refine the results to match desired criteria.



Audit Committee

City council members usually have different relationships with each other - they can be colleagues from work, relatives, members of the same political party, business partners. The city council of the Bebras city has 11 members whose relationships are described in a graph. Points mean members and lines mean relationships (full lines means colleagues or relatives, dashed lines political or business partners).

A city council has its own audit committee consisting of some members of the council, which controls the management. Its members must not have relationships with anyone else on the audit committee.



Question

How many members can the audit committee in Bebras city have at most?

2

3

4

5

EXPLANATION

Answer

Correct answer is B) 3.

Explanation

For greater convenience, we name the points with letters. We can count that four lines go out from each point.

For example, the point C only has relationships with A, B, D, E.

We can start to choose the audit committee anywhere because the diagram does not have any special points.

If we choose a member A, the next nearest member without relationship with A counterclockwise is D. The next one without relationship with D is G and then the next such one is J. But J has a relationship with A so J can't be in the audit committee. Only 3 members A, D, G are independent in this case.

We get the same number of audit committee members when starting in any point and go any direction, and it is therefore not possible to have more than 3 members in this committee.

BACKGROUND INFORMATION

The relationships among city council members are modeled with a *graph*. This consists of *nodes* (representing some persons/objects, and usually depicted as points) connected in pairs by edges (usually depicted as lines connecting points). Both the diagram in the task and the diagram in the explanation depict the same graph.

A graph is an abstract structure and is useful when one wants to focus on the connections: the graph emphasizes the important features (who is in relation with whom) but omits non-important details (what are the persons/objects involved, what relation connects them).

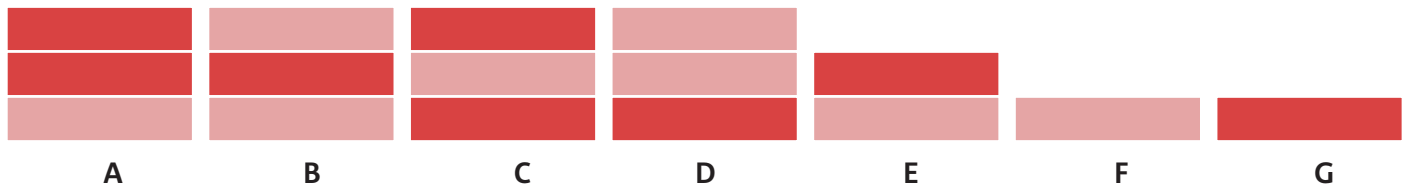
Computers can work very effectively with graphs, so computer scientists have to understand graphs, their types and properties very well.

Using the graph terminology, the task requires to find *independent sets* of the graph, i.e., subsets of nodes in the graph, no two of which are connected by an edge. In particular here we are looking for the *independence number* of the graph, which is the size of the largest maximum independent sets.



Stacks of Tokens

Stephan placed seven stacks of tokens (light and dark red) on the table, as shown in the figure below.



He wants to put these stacks on top of each other without splitting them to form two perfectly equal stacks; the resulting two stacks must have the same height (eight tokens) and the same sequence of colors (from the bottom up).

Question

Possible stacks are represented below with a bracketed notation. (x, y, z, ...) indicates that the stack is made by placing y onto x, z onto y, and so on. Which of the following four pairs of stacks is NOT a valid solution?

(A, F, B, G) and (E, D, C)

(B, E, A) and (F, D, G, C)

(B, A, E) and (F, D, G, C)

(B, E, A) and (F, D, C, G)

EXPLANATION

Answer

The right answer is: 3. (B, E, A) and (F, D, G, C).

Explanation

The other answers are all valid solutions to the proposed problem.

Since there are 16 tokens, of which 8 are white and 8 red, each of the two resulting stacks must be made up of 4 white and 4 red tokens. Of the four stacks A, B, C, and D (3 tokens high), two must be in one stack and the other two in the other stack. Examining the 3 token high stacks, we can see that we cannot put A and C in the same stack, and similarly we cannot put B and D in the same stack. Either of those combinations would make it impossible to balance the colors (two white tokens would need to be added to the pair A-C, and two red tokens to the pair B-D). It can also be shown that putting A and D in the same stack (and B and C in the other stack) does not lead to any result.

The quickest way of finding the solution in this case is to examine the top and bottom of each stack. Since none of the 3 token stacks are identical, any solution which uses one of the 3 token stack on the top (or bottom) of both parts of the final answer is incorrect. Since the incorrect result has A and C at the top of the two stacks, we can quickly compare those two stacks and see that the top of each stack will be different.



Stacks of Tokens – continued

BACKGROUND INFORMATION

Imagine you have a bag with a lot of coins and you want to divide them into two parts of equal value. If all the coins had the same value and were even in number, then the problem would be easily solved: it would be enough to divide them into two heaps that have the same number of coins; but if the coins have many different values, then the task becomes a little more difficult.

In “more mathematical” terms, the partition problem is to decide whether a multiset (i.e., a set with a multiplicity for each element) of positive integers can be partitioned into two sub-multisets such that the sum of the numbers in the first sub-multiset equals the sum of the numbers in the second sub-multiset. This problem (in general) is NP-complete (i.e., in practice, there is no known procedure for solving it that is efficient in any case); but it can be solved using a pseudo-polynomial time algorithm, based on dynamic programming.

Our task (in general) is certainly not simpler than solving a partition problem: in fact, it is not enough that the height of the two resulting stacks is the same, but the color sequence of the tokens must also be the same. A brute force method can however be implemented through an exhaustive search algorithm.

In the case of our task, let's list all the subsets of the seven stacks for each of which the sum amounts to eight tokens: {A, B, E}, {A, B, F, G}, {A, C, E}, {A, C, F, G}, {A, D, E}, {A, D, F, G}; here we can stop, since continuing we would find the complements of these already listed sets. Moreover, we can eliminate the sets {A, C, E} and {A, C, F, G}, since the stacks that belong to them have a total of five red tokens. Therefore, only four sets remain; for each of them and the respective complement, it is now a question of trying all the possible stackings, in order to verify if any of these produces two identical stacks... In our case, the set {A, B, E} leads to two solutions, while the set {A, B, F, G} leads to only one solution, and the last two sets to none. It is then understood that the time required by this procedure grows very quickly as the initial number of stacks (only seven, in our case) increases.

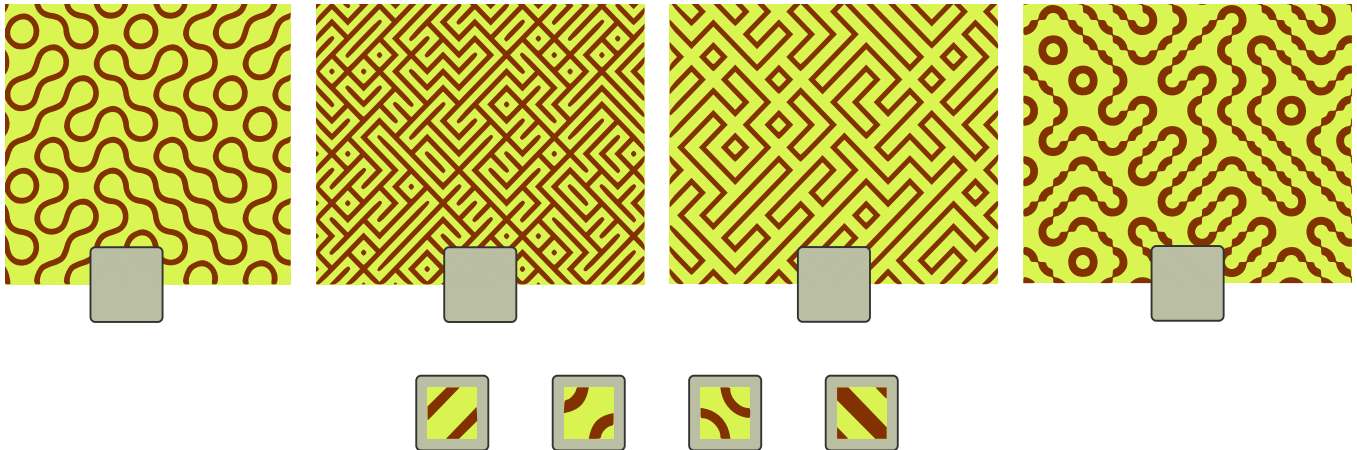


Truchet Tiles

The following patterns have been created from lining up square tiles. Each pattern has been created using a single type of tile - these four tiles are also shown below.

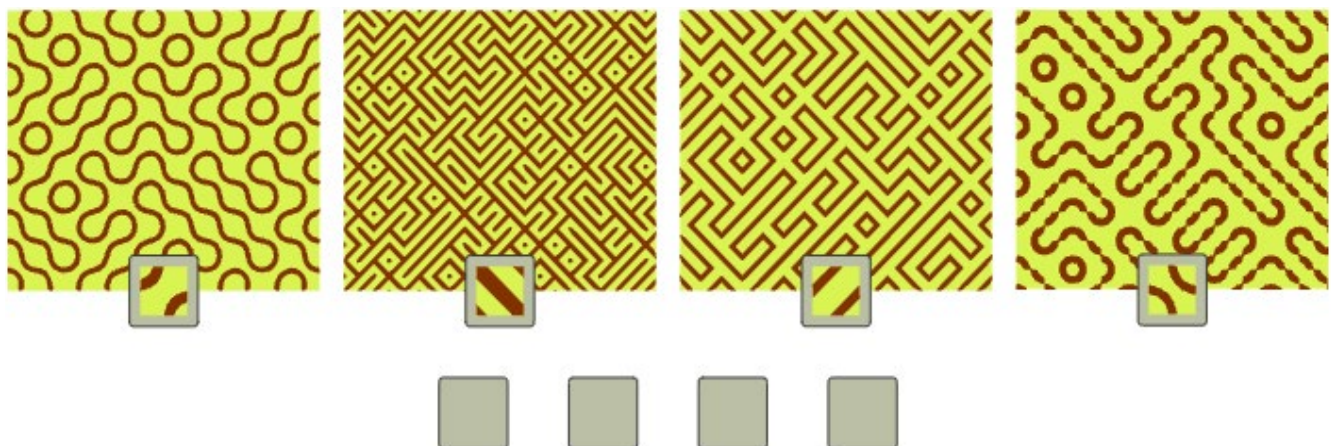
Question

Match each square tile to its corresponding pattern.



EXPLANATION

Answer



Explanation

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

In the pattern on the left, we see one continuous wavy line. So the lines in the tiles that make this pattern would need to be curved and intersect in the middle of the edges of the tile. Of the two tiles with curved lines, only one fits this criteria.



Truchet Tiles – continued

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

One of the remaining tiles has a dark colour right in some of its corners. By looking at the corners in the remaining two patterns, only one of them has the same dark colour there - they appear like dots or small triangles. This tile and pattern can therefore be assigned to each other.

Now only one tile and one pattern remain, so they have to go together.

BACKGROUND INFORMATION

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

The fact that complex patterns can be created with very simple building blocks has fascinated people for a long time. Truchet tiles are studied in mathematics and computer science, and they are used in computer games to generate mazes or decorations.



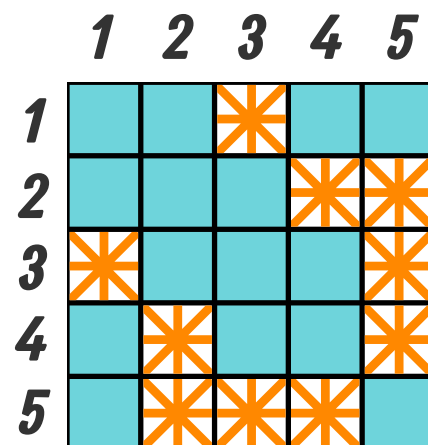
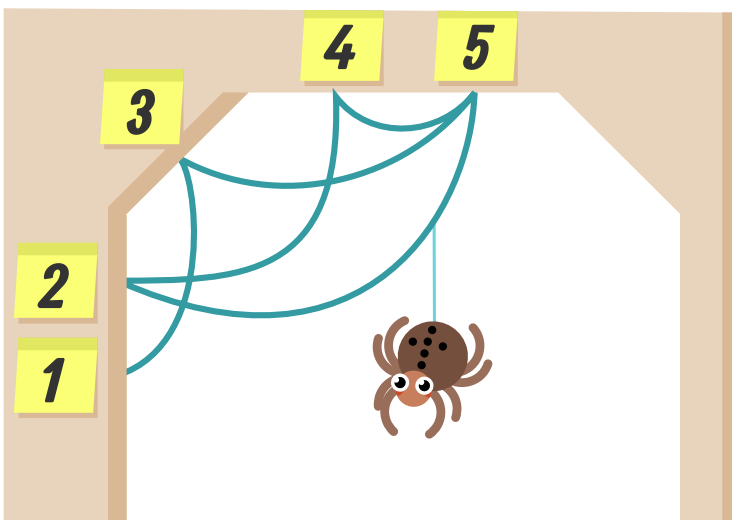
Spider Quilts

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

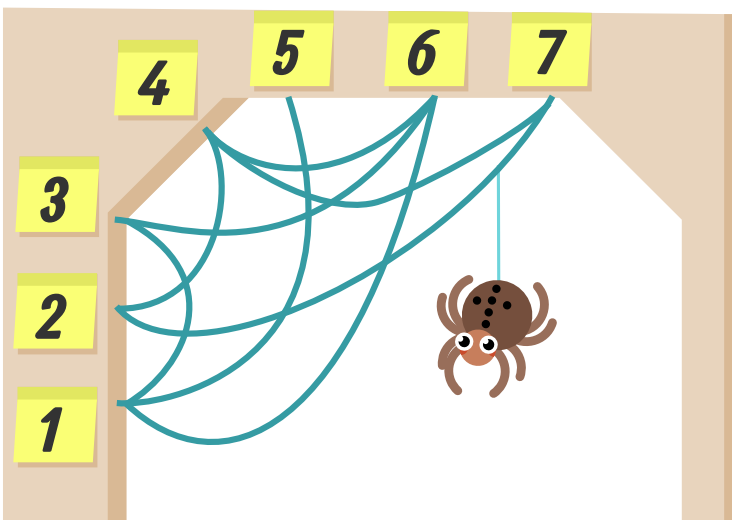
She numbers the web's anchor points from 1 to N and then arranges squares of fabric into an N-by-N grid as follows:

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

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



Wanda has now seen the following web and wants to use it to design a new quilt:

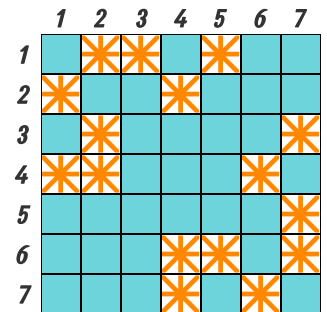
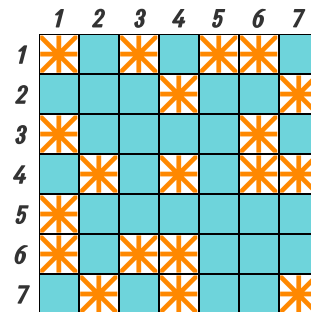
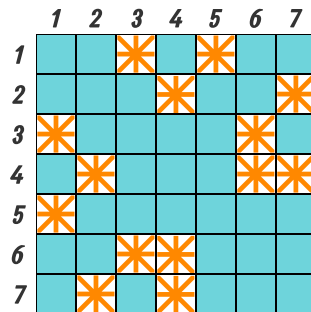
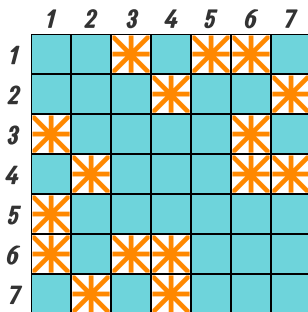




Spider Quilts – continued

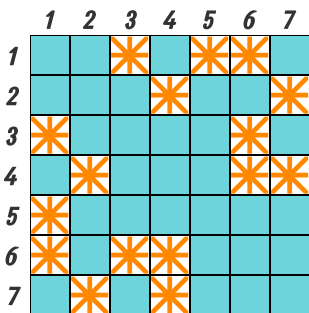
Question

What will her quilt look like?



EXPLANATION

Answer



Explanation

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

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

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

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

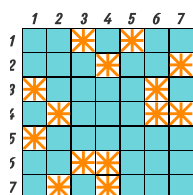
The web has silk joining anchor point 5 with anchor point 1. So the fifth row of the quilt will have crossed fabric in column 1.

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

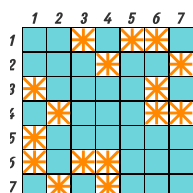
The web has silk joining anchor point 7 with anchor points 2 and 4. So the seventh row of the quilt will have crossed fabric in columns 2 and 4.



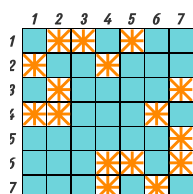
Spider Quilts – continued



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



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



is incorrect because the entire quilt pattern is rotated 90 degrees. We can eliminate this option by recognising that this method of creating a pattern will always create a line of symmetry in the pattern, running from (1,1) to (7,7). This pattern does not have such a line of symmetry and can quickly be discarded.

BACKGROUND INFORMATION

The spider web can be considered as a *graph*, a concept that is often used in computer science.

A graph is composed of *vertices* (the anchor points of the web) and *edges* (the pieces of silk between two anchor points). Graphs are used to represent objects and the relationships between objects. For example, a graph could show people who are friends on social media, or flights between countries.

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

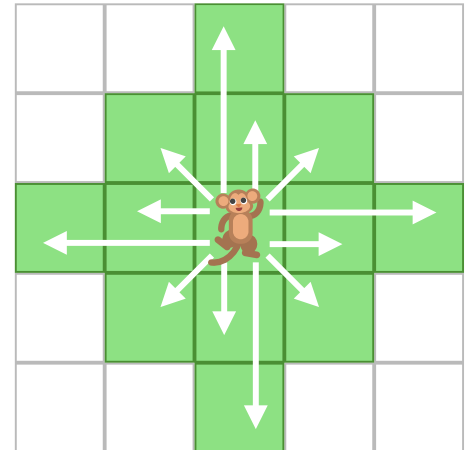


Jumping Jack

Jack the monkey lives in a park. He can jump from one tree to another if it is either up to two cells away horizontally or vertically, or one cell away diagonally, as shown in the diagram on the right.

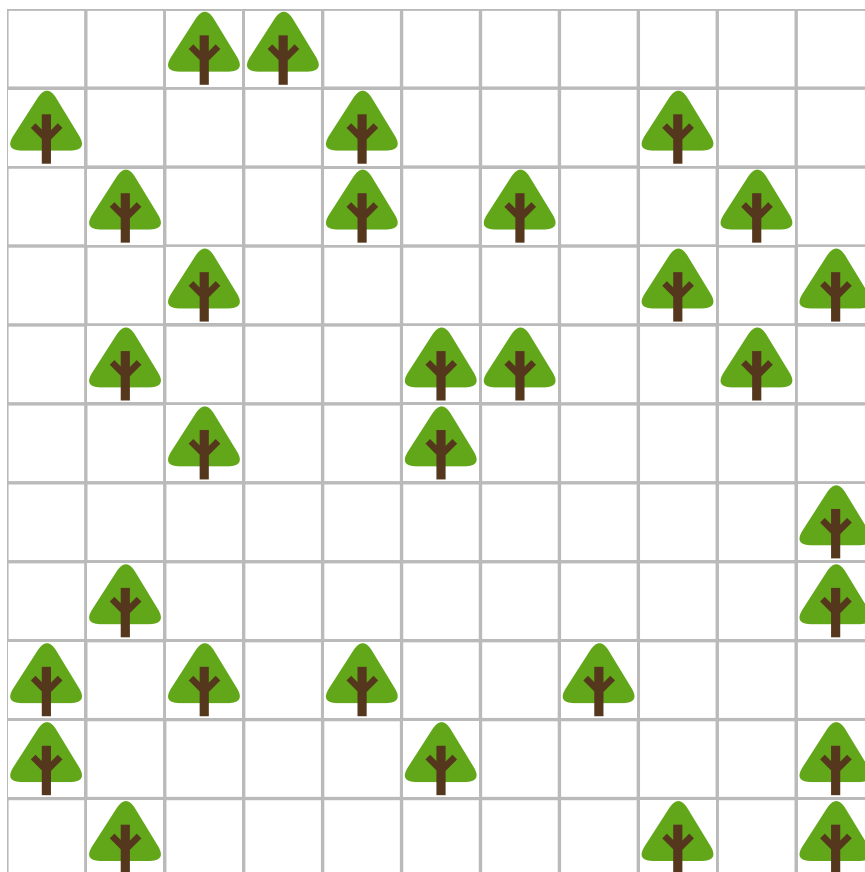
Jack plays a game in which he jumps to as many different trees as possible without touching the ground. He can start from any tree in the park.

In the interactive map below you can click on a tree to change it from one type to another.



Question

Find the biggest number of trees Jack can visit in one go without touching the ground and change them to orange square trees.

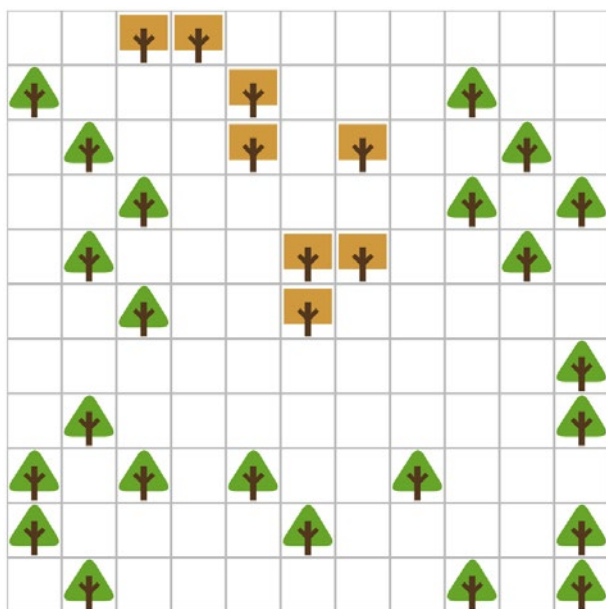




Jumping Jack – continued

EXPLANATION

Answer



Explanation

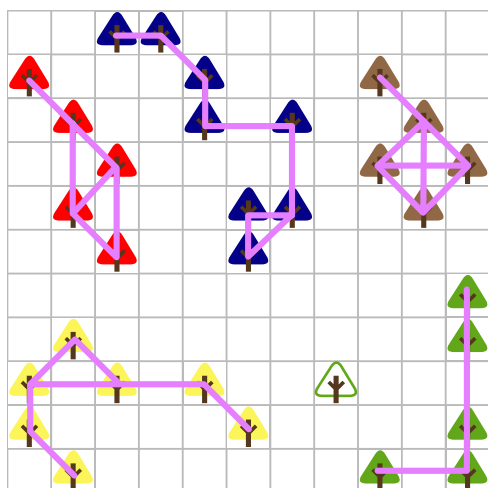
In the diagram below, the groups of trees that Jack can visit without touching the ground have been coloured in different colours.



There are six groups of trees in the park. If Jack starts on a tree coloured in yellow, he can reach all the yellow trees, and no trees of other colours. How do we find such groups? Pick a random tree and colour it in a certain colour. Then use the same colour for all trees that are reachable from it. And all trees that are reachable from those trees, too. And so on, until you cannot reach any other trees. If there are any trees that haven't been coloured yet, take another colour and start again from a random uncoloured tree. This colouring simulates Jack exploring.

The largest group of trees is the dark blue one, which contains 8 trees. The dark blue cluster is the correct answer.

BACKGROUND INFORMATION



From the point of view of computer science, this question involves manipulating a *graph*: the trees are called *vertices*, and two trees are connected with an edge when Jack can jump between them. In the diagram below, the edges are as purple lines between trees.

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

The procedure for colouring is similar to a number of different graph algorithms that deal with searching: *breadth-first search* and *depth-first search*.



Still Life

There is a selection of fruit consisting of a column of red apples, a column of yellow apples, a column of yellow bananas, and a column of red strawberries.

A painter has chosen one piece of fruit to paint from the selection. You are trying to work out which piece of fruit the painter has chosen by asking them yes or no questions.

The painter will answer every question truthfully.

You want to find out which piece of fruit they have chosen in as few questions as possible. The most efficient strategy minimises the number of questions asked regardless of what piece of fruit the painter has actually chosen.



Question

Which of the following questions could be the first one asked as part of the most efficient strategy described above?

Is the fruit a banana?

Does the fruit have a bite taken out?

Is there a worm in the fruit?

Is the fruit yellow?

EXPLANATION

Answer

The most useful question to start with is: "Is the fruit yellow?"

Explanation

This question will eliminate half the fruit with a single question. All fruits of one colour will be eliminated, leaving 8 fruit regardless of which answer the painter gives. This guarantee of eliminating half the fruit is the best outcome of any of the question options.

The second question would be: "Is the fruit a banana/apple/strawberry?" which will again cut the number of fruit left in half, with 4 remaining of one kind of fruit.

The third question will be: "Does the fruit have a bite taken out?"

And with one last question we will know exactly which fruit the painter will choose. We will only need 4 questions, regardless of which piece of fruit the painter chose.

Continued on next page



Still Life – continued

The remaining three options are not correct because while they may reduce the number of questions for some choices, they are less efficient for other choices and therefore are not part of the *most efficient* strategy:

- Option B is not correct because you will only eliminate 4 fruits (the bananas) if the answer is no. You could potentially need 5 questions in total to get to the right fruit.
- Option C is not correct because you will only eliminate 4 fruits if the answer is no. You could need as many as 6 questions to get to the right fruit.
- Option D is not correct because you will only eliminate 1 fruit if the answer is no. You could need 5 questions in total to get to the right fruit.

BACKGROUND INFORMATION

This question is about *expert systems*. Expert systems are systems that have a lot of knowledge about a specific topic. An expert system will use previous knowledge and ask questions that help to find the solution to a certain problem as fast as possible. The first expert system was built in the 1970s. They are the first successful form of artificial intelligence (AI). In this task, the person asking questions is the expert system. Asking questions to the painter, together with the known possibilities, will give the desired information as fast as possible.

You could obviously ask ‘Have you chosen this piece?’ for each individual piece of fruit, but it could take you up to 16 questions! If you ask smarter questions, you can find your solution much faster. Remember that you don’t know beforehand what the painter will answer, so you are not trying to ask questions based on guesswork. Instead, you are trying to ask questions so that both answers will eliminate as many fruits as possible.


If you ask a question where half the objects fit the description, and the other half does not, you will guarantee that half the number of possibilities will be eliminated no matter what the answer to the question is.

If you ask a question where only one object fits, you have a 1/16 chance to have the correct object now (1 object left), but a 15/16 chance to have 15 objects left. This tactic requires an average of 8 questions to find the correct piece of fruit, which is far less efficient than the 4 questions needed in the strategy outlined in the answer.



Playing with Hats


A beaver likes to play a game by placing circular pebbles on square paving stones.

The beaver moves from left to right, one square at a time .













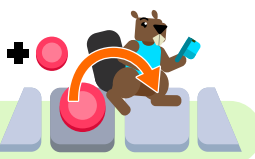


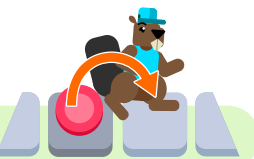
The beaver has a hat and behaves differently depending on whether they have the hat in their hand or on their head.

The rules that the beaver follows are listed below:

- If the beaver has the **hat in their hand** and steps on a square with **no pebble**, they continue on with no change.
- If the beaver has the **hat in their hand** and steps on a square with **a pebble**, they take the pebble and put the hat on their head before moving to the next square.
- If the beaver has the **hat on their head** and steps on a square with **no pebble**, they place a pebble on the square and take the hat off before moving to the next square.
- If the beaver has the **hat on their head** and steps on a square with **a pebble**, they continue to the next square with no change.

The pictures in the table below show the rules of the game. The changes for each situation are shown “before  after”.

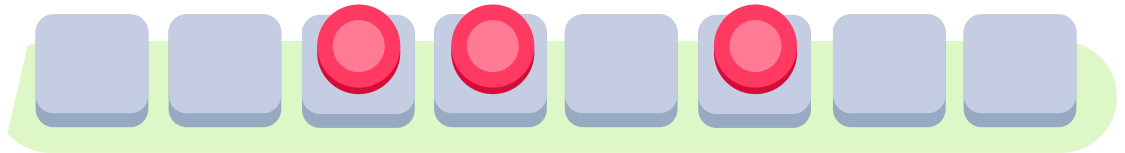
Rules:



Playing with Hats – continued

At the beginning, the beaver has the hat in their hand and three pebbles are on the squares in the positions shown below.



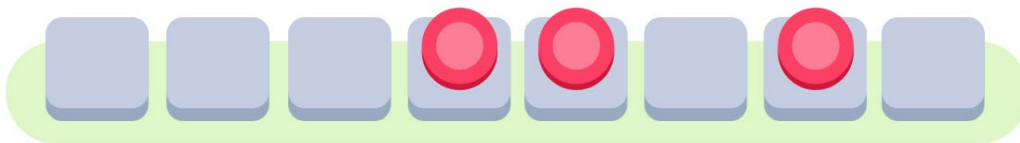
Question

Show which squares have pebbles on them after the beaver has moved over all of them and left the last square.

Click on the squares below to place or remove a pebble.

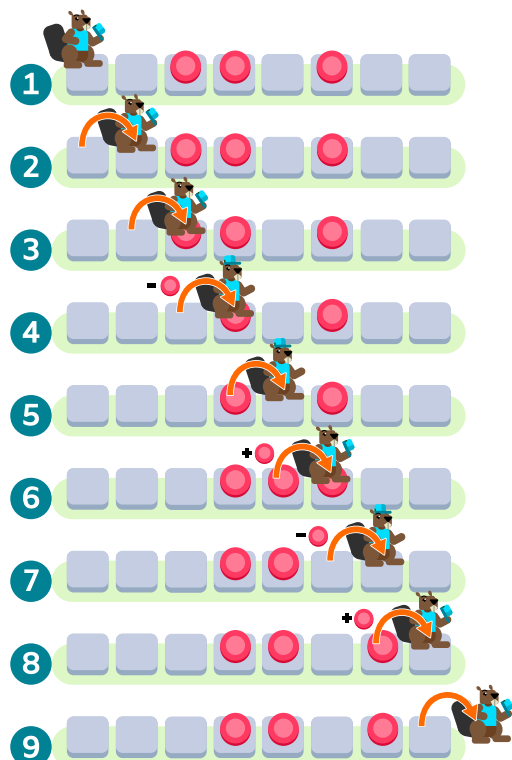
EXPLANATION

Answer



Explanation

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



Continued on next page



Playing with Hats – continued

BACKGROUND INFORMATION

The beaver has two states:

- hat in hand
- hat on head

Depending on its state, the beaver behaves differently. The beaver with its rules behaves like a *Turing machine*. A Turing machine is a useful model for computation in computer science. Although it is very simple, it is as powerful and as efficient as any programming language. This means any software program can be converted into a Turing machine and, conversely, any Turing machine into a program. It was first described in 1936 by the English mathematician and computer scientist Alan Turing. Turing machines are one of the most important formal models in computer science.

A Turing machine has various necessary components:

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

Bebras Challenge 2022 Round 1

Years 11+12



Vaccination Centres

People from a region consisting of six main cities are in the process of getting vaccinated. Due to resource limitations, only two cities can be equipped with a vaccination centre. The map below shows the region and its cities, with the times needed to travel from one city to another (in hours). Not all cities are directly connected by roads.

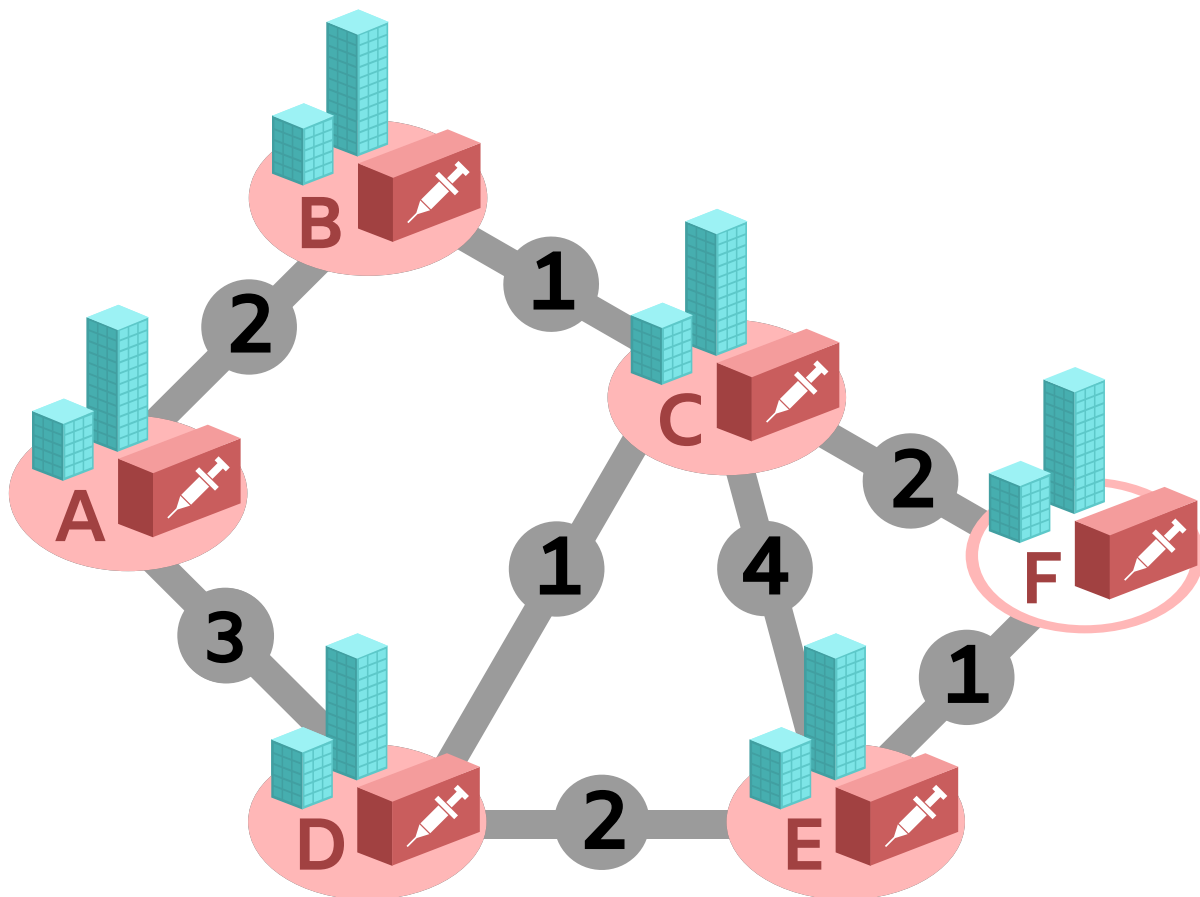
The cities where the two vaccination centres will be located should be chosen so that the time it takes everyone to reach them from their own cities is as short as possible.

This should apply to all cities - this avoids the situation of reducing the travel time for some cities but leaving other cities with long travel times.

Question

Given that one vaccination centre will be placed in the existing hospital in City F, where must the second one be placed?

Select the city to place the centre.





Vaccination Centres - cont'd

EXPLANATION

Answer

The correct answer is City B.

Explanation

Since one hospital is placed in City F, there are five possibilities to place the second one. For each possibility, we have to check the minimum time needed to go from every city to any vaccination centre. In the following table, the columns show each possible placement for the two vaccination centres and the rows show the number of hours needed to reach the closest vaccination centre.

Minimum time go from city...	Locations of vaccination centres				
	A and F	B and F	C and F	D and F	E and F
A	0	2	3	3	5
B	2	0	1	2	3
C	2	1	0	1	2
D	3	2	1	0	2
E	1	1	1	1	0
F	0	0	0	0	0
Max	3	2	3	3	5

The last row shows the largest number of hours needed to reach a vaccination centre. Therefore, placing the second one in City B is the best solution. People in each city would be able to reach a vaccination centre in no more than 2 hours.

BACKGROUND INFORMATION

This task is related to the vertex k -center problem which can be solved by using an algorithm designed by computer scientists. This type of problem can be represented as a graph - a set of nodes that can be connected together with links that have a weight (cities and roads with travel time in this task). The problem consists of choosing k nodes from the graph (2 in this task) so as to minimise the time/distance from any node of the graph to any of the k selected nodes.

Such a problem is very common when trying to choose where to place facilities such as fire stations, schools, police stations, etc. (or hospitals in this task). The criterion that has to be minimised can be the time needed to reach the facilities, the distance to be travelled, or any other criterion that is available.



Secret of the Diary

Petra and Jana have found a secret diary of their classmate Lucie. They think that she is concealing the name of her new puppy in the diary. Unfortunately for them, Lucie has encrypted the text in the diary using horizontal and vertical lines with the help of the table of letters on the right. The two girls were only successful in deciphering the name of Lucie's brother, PAVEL, from the code below:

⌵ ⌵ ⌵ ⌵ ⌵

A	B	C	D	E
F	G	H	I	J
K	L	M	N	O
P	R	S	T	U
V	W	X	Y	Z

Question

Uncover the name of Lucie's puppy hidden in the code:

⌵ ⌵ ⌵ ⌵ ⌵

EXPLANATION

Answer

The name of Lucie's puppy is JOSEF (noting that the answer is not case sensitive, so Josef and josef are also accepted for example).

Explanation

The horizontal and vertical lines found in the encrypted symbols have meaning. The number of horizontal lines corresponds with a row number in the table of letters. Similarly, the number of vertical lines corresponds with a column number. The letter found where the row and column meet is the letter encrypted by the symbol.

For example, the first symbol $\begin{array}{|c|c|c|c|c|} \hline \hline \hline \hline \hline \hline \end{array}$ has 2 horizontal lines and 5 vertical lines. The letter found where row 2 and column 5 meet is J.

Using this process for each remaining symbol, the name JOSEF can be decrypted.

BACKGROUND INFORMATION

Encrypting and decrypting text is a very important part of informatics. It is a common requirement for communicating via the internet in order to be safe and in order for information to be secure.

The method of encryption used in this task is not used in common practice, but the task demonstrates some common computational thinking skills. First, it requires *logical thinking* in order to discover the relationship between the symbols and the letters. Second, it requires *abstraction* in order to identify the important features. You may have noticed that the letter E is encrypted twice (once in PAVEL and once in JOSEF) but the encrypted symbols look different. The length of the lines or where they cross is *not* an important feature and can be abstracted away. What is important is the number of lines, and whether they are horizontal or vertical.



Robo-Rally

Coraline and Tristan are playing a board game called Robo-rally.

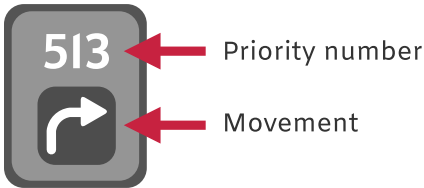
Both players have a token on the board, and they direct their token's movements by playing cards. The cards specify one of the following movements:




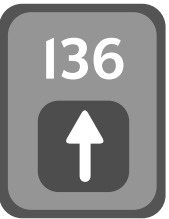






- *Move one space forward,*
- *Turn left (but stay on the same space),*
- *Turn right (but stay on the same space).*

The cards must be arranged by the player in the order they wish the token to move.

Coraline and Tristan each choose 4 cards and then follow the movement instructions in order, one card after the other. All tokens move **simultaneously**. Tokens attempting to move into the same space at the same time are resolved by priority numbers printed on the cards.

The movement on the card with the *higher priority number* will be executed *first*. If a player's token needs to be moved to a space used by the other token, the token on that space will be shifted in the moving token's direction.



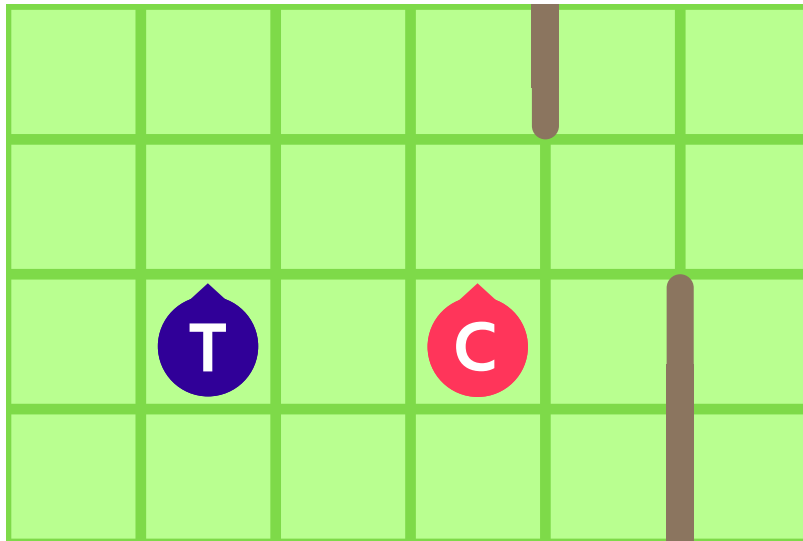
	   
Caroline's cards	
	   
Tristan's cards	



Robo-Rally - continued

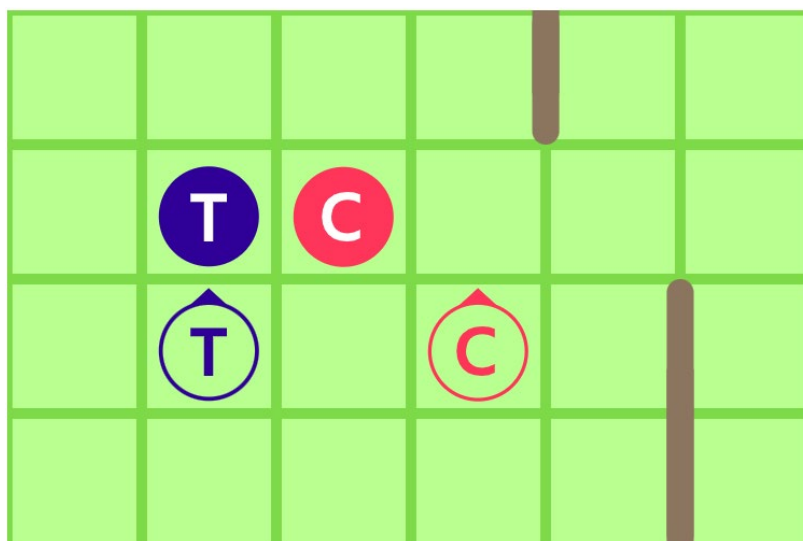
Question

Drag the tokens to their new positions on the board.
The arrows on the tokens below indicate the direction that they are initially facing.



EXPLANATION

Answer

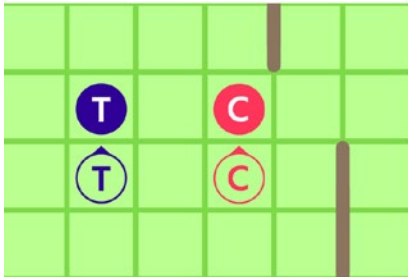




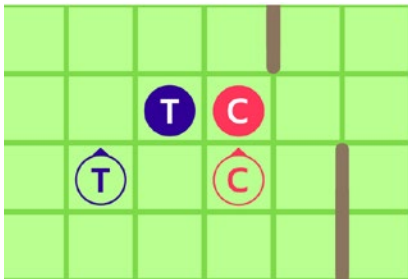
Robo-Rally - continued

Explanation

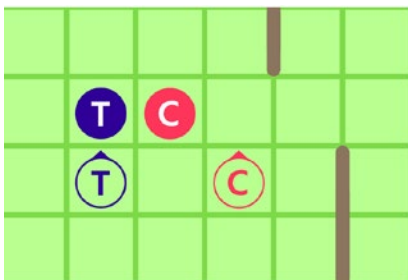
The position of the tokens can be determined by stepwise going through each of Coraline and Tristan's cards, and following the relevant rules.



The first move can be executed simultaneously. The tokens will be moved one space forward.



The second move can be executed simultaneously, too, and the tokens will turn - Coraline's will turn to the left, and Tristan's will turn to the right.



For the third move, both tokens would be moved to the same space. Tristan's token moves first, because his card has the higher value (564). Then Coraline's token will move one space forward – this shifts Tristan's token one space in the direction that Coraline's token moved.

Lastly, the tokens can turn simultaneously, giving the answer.

BACKGROUND INFORMATION

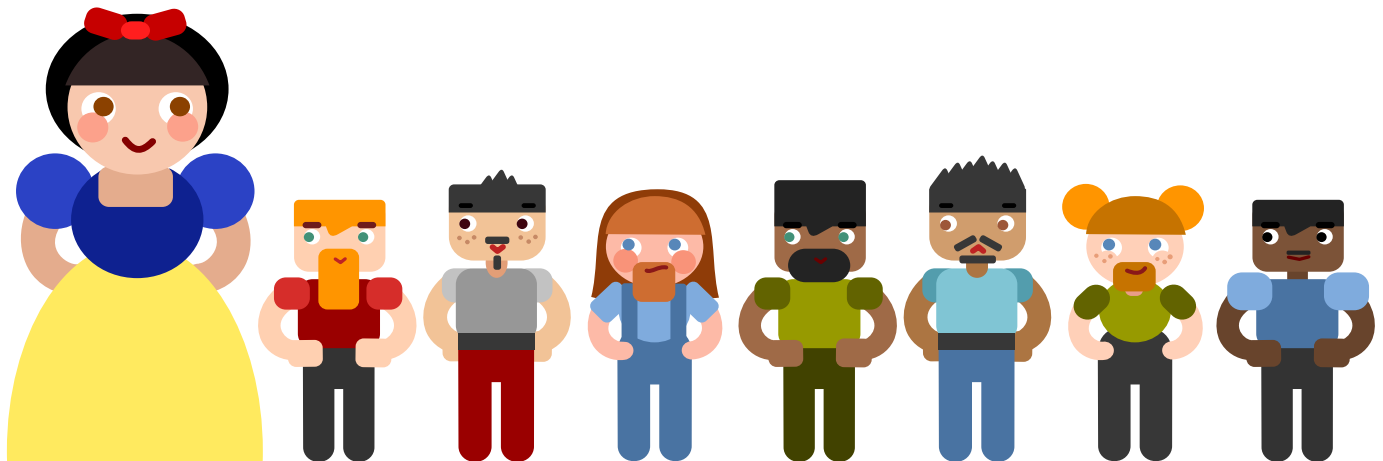
Some computer programs divide a larger task into smaller tasks. These programs let several processes work on the smaller tasks at the same time, and mostly independently. This is called *concurrent execution* of processes. During the concurrent execution, it is possible that a process needs to wait to get access to a shared resource or to affect another process in some way. It is important to define rules for coordinating the processes' behavior in such situations - we saw this in the case where two tokens needed to move into the same space at the same time.



Snow White

The Seven Dwarfs are having an argument. For simplicity, Snow White now refers to them by numbers which reflects who they are friends with:

- 12 is friends with 1 and 2,
- 13 is friends with 1 and 3,
- 23 is friends with 2 and 3, and
- 123 is friends with all.



Snow White has proposed a game to try and distract them. The game has the following five rules:

- If she shouts “2” - 2 comes into the house (or stay there) together with his friends 12, 23, and 123.
- If she shouts “3” - 3 comes into the house (or stay there) together with his friends 13, 23, and 123.
- If she shouts “4” - 1 and his friends change their individual positions: if they were inside the house they go outside, and if outside they come back to the house.
- If she shouts “5” - dwarfs 2 and his friends change their positions.
- If she shouts “6” - dwarfs 3 and his friends change their positions.

For example, suppose 1, 2, and 12 are in the house while 23 and 123 are outside.

If Snow White shouts “5”, then 2 and 12 go outside, while 23 and 123 enter the house, and 1 stays in the house.

Question

All the seven dwarfs are now in the house, but Snow White would like to stay alone with the prince.

What is the shortest sequence of commands that she can use to send all the dwarfs outside?

Example: 6532 represents the sequence 6, 5, 3, 2.



Snow White - continued

EXPLANATION

Answer

42536 or 43625.

Explanation

Since there are no direct commands to send a particular dwarf out of the house, the best approach is to dismiss the dwarfs through commands “4”, “5” or “6”. The problem is that some of them will be called back using these commands if they are already out of the house, so Snow White needs to make sure that every dwarf affected by the commands “4”, “5” or “6” are brought back into the house before calling that particular command.

Because there is no direct way to call 1 and his friends back in the house, Snow White can start by calling “4” and sending 1 and his friends out. This will leave 2, 3 and 23 in the house. At this point, some friends of 2 and 3 are out of the house. Snow White can then call the combinations of “25” and “36” in either order. Each combination brings either 2 or 3 and their respective group of friends back in the house, and then sends that group of friends out together. The following tables shows the position changes after every command:

- When Snow White calls 42536:

Command	4	3	6	2	5
Dwarfs inside the house	2, 3, and 23	2, 3, 12, 23, and 123	3	3, 13, 23, and 123	-
Dwarfs outside the house	1, 12, 13, and 123	1 and 13	1, 2, 12, 13, 23, and 123	1, 2, and 12	1, 2, 3, 12, 13, 23, and 123

- When Snow White calls 43625:

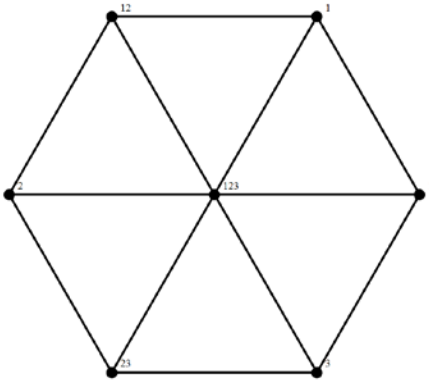
Command	4	3	6	2	5
Dwarfs inside the house	2, 3, and 23	2, 3, 13, 23, and 123	2	2, 12, 23, and 123	-
Dwarfs outside the house	1, 12, 13, and 123	1 and 12	1, 3, 12, 13, 23, and 123	1, 3, and 13	1, 2, 3, 12, 13, 23, and 123

BACKGROUND INFORMATION

The first item concerns *data representation*: dwarfs can be represented as numbers, and these numbers conveniently also represent friendships. Commands can also be represented as numbers. The answer is also a number, representing a sequence of commands - this answer sequence is a (very simple) *program*!

Friendships could also be represented by a simple *graph*, a sort of centered hexagon where 123 is the centre and the edges represent friendships.

More surprisingly, the illustrated problem also can be linked to *computer graphics*. The dwarfs might represent intersecting regions in a plane, which we want to colour with two colours (white=inside the house, black=outside the house). This task then essentially becomes the study of the shortence sequence of commands to colour the entire plane black.





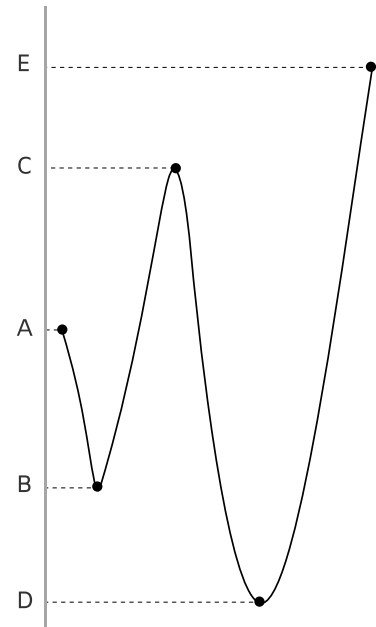
Comfort Temperature

Every day, beaver Theophilus measures his lake's water temperature in some units, which he records in a table. He records the first measurement immediately after waking up, and the last one shortly before bedtime.

Theophilus knows that the temperature changes constantly, so during the day he records only the extreme temperatures:

- a temperature where the previous measurements *were increasing*, and immediately afterwards began decreasing, or vice versa;
- a temperature where the previous measurements *were decreasing* and immediately afterwards began increasing.

For example, if the temperature changed like in the adjacent drawing, Theophilus would have written the numbers A, B, C, D, E in the table. There is exactly one temperature value, the "comfort temperature", at which Theophilus feels best.



Question

What are the limits of the "comfort temperature" if the comfort temperature was met exactly five times yesterday, and yesterday's observations are the following:

5.1, 5.8, 5.5, 5.9, 5.3, 5.7, 5.4, 5.8, 5.6?

Select *two* correct options.

5.1

5.8

5.5

5.9

5.3

5.7

5.4

5.8

5.6

EXPLANATION

Answer

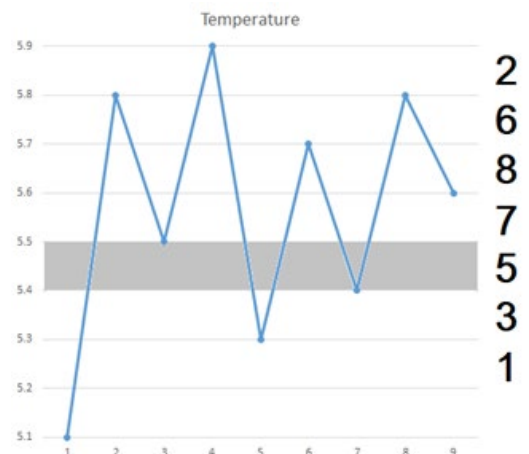
Between 5.4 and 5.5.

Explanation

Using the given data, we can build a graph similar to example given in the task description. The graph looks something like this:

We can then count the number of times where horizontal lines intersect with the graph. We are looking for the range of horizontal lines where there are five intersections with the water temperature graph - this indicates the range where a temperature was reached five times throughout the day, and could therefore be a candidate for the "comfort temperature".

These criteria correspond to the interval (5.4;5.5) (excluding endpoint values), as depicted by the shaded grey region above.



Continued on next page



Comfort Temperature – cont'd

BACKGROUND INFORMATION

In this problem, students need to apply basic *data analysis* skills. While *data visualization* is very helpful here, task can be solved also without it. For example, an approach could be using number intervals, counting how many times a particular interval is covered (in other words, how many times a temperature in that interval appear throughout the day), and adding new endpoints when they appear in measurements.

This approach is shown below:

(5.1, 5.8) 1

(5.1, 5.5) 1; (5.5, 5.8) 2

(5.1, 5.5) 1; (5.5, 5.8) 3; (5.8, 5.9) 1

(5.1, 5.3) 1; (5.3, 5.5) 2; (5.5, 5.8) 4; (5.8, 5.9) 2

(5.1, 5.3) 1; (5.3, 5.5) 3; (5.5, 5.7)..5; (5.7, 5.8) 4; (5.8, 5.9) 2

(5.1, 5.3) 1; (5.3, 5.4)..3; (5.4, 5.5) 4; (5.5, 5.7)..6; (5.7, 5.8) 4; (5.8, 5.9) 2

(5.1, 5.3) 1; (5.3, 5.4)..3; (5.4, 5.5) 5; (5.5, 5.7)..7; (5.7, 5.8) 5; (5.8, 5.9) 2

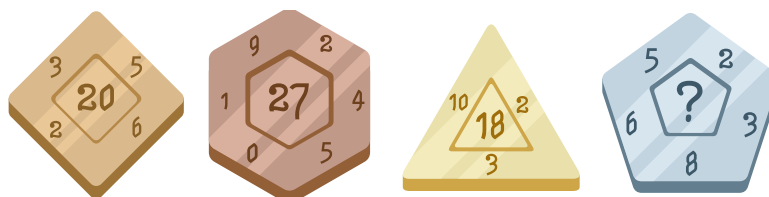
(5.1, 5.3) 1; (5.3, 5.4)..3; (5.4, 5.5) 5; (5.5, 5.6) 7; (5.6, 5.7)..8; (5.7, 5.8) 6; (5.8, 5.9) 2

This strategy utilises *segments*, which are well-known data structures in computer programming and are also used in various applications.



Secret Number

In Bebravia payments are made with special coins. Each coin has its own value written in the centre.



A citizen of Bebravia has the four coins shown above, but the value at the centre of one of the coins is rubbed out. However, each coin's value can be worked out using the same set of rules. From this, the citizen was able to determine the missing value.

Question

What is the missing value?

22

23

26

29

EXPLANATION

Answer

29.

Explanation

The value of each coin is calculated as following: sum of the numbers surrounding the central coin value plus the number of sides the coin has. In this task there are four different coins:

Square: $3 + 5 + 6 + 2 + 4 = 20$

Hexagon: $1 + 9 + 2 + 4 + 5 + 0 + 6 = 27$

Triangle: $10 + 2 + 3 + 3 = 18$

Pentagon: $5 + 2 + 3 + 8 + 6 + 5 = 29$

A student attempting this might at first try to find a relationship between the numbers around the outside and the value in the middle and, after failing to find a simple pattern (and remembering that Bebras tasks do not require any mathematical knowledge beyond simple numeracy) go back to the task and check they have extracted all of the data given. Looking for more data should enable the student to notice that the coins have different numbers of sides and then quickly work out the pattern.



This question comes from
Uruguay

Years 3+4

Years 5+6

Years 7+8

Years 9+10

Years 11+12 Medium



Secret Number – continued

BACKGROUND INFORMATION

The value of each coin is calculated as following: sum of the numbers surrounding the central coin value plus the number of sides the coin has. In this task there are four different coins:

Square: $3 + 5 + 6 + 2 + 4 = 20$

Hexagon: $1 + 9 + 2 + 4 + 5 + 0 + 6 = 27$

Triangle: $10 + 2 + 3 + 3 = 18$





Pentagon: $5 + 2 + 3 + 8 + 6 + 5 = 29$

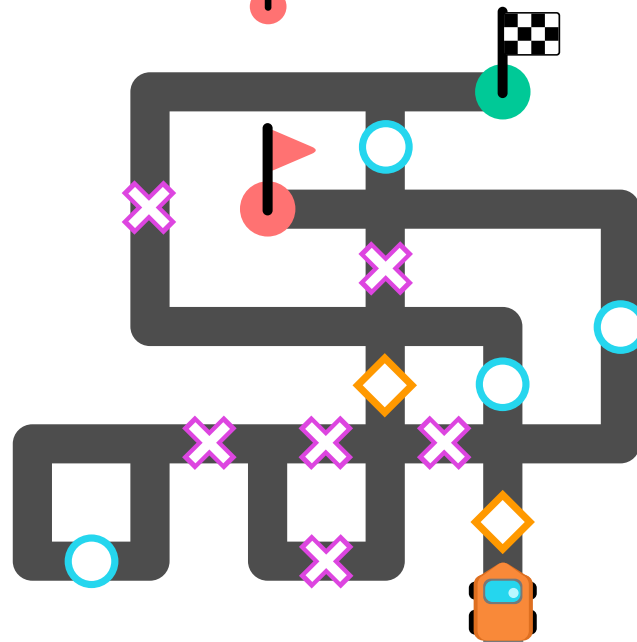
A student attempting this might at first try to find a relationship between the numbers around the outside and the value in the middle and, after failing to find a simple pattern (and remembering that Bebras tasks do not require any mathematical knowledge beyond simple numeracy) go back to the task and check they have extracted all of the data given. Looking for more data should enable the student to notice that the coins have different numbers of sides and then quickly work out the pattern.



Symbol Reading Robot

A robot starts from the position shown below and moves along the lines.

There are three symbols ,  and  on the lines that decide the direction it should take at the next intersection. The robot must not reach the .



Each symbol has a different meaning and could mean:



turn left at the next intersection, or



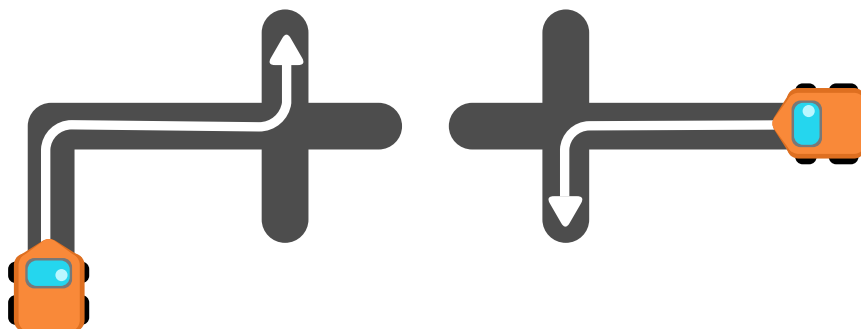
turn right at the next intersection, or



go straight at the next intersection

Unfortunately, we do not know which symbol means what.

The meaning of the symbol remains the same regardless of the direction the robot is moving. For example, the arrow in the picture below shows how the robot would turn, coming from either direction, if a triangle symbol meant “turn left”.

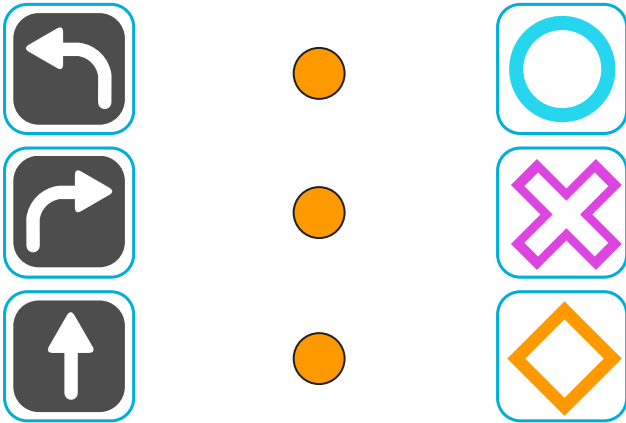




Symbol Reading Robot – cont'd

Question

Help the robot reach  by assigning correct meaning to the symbols.



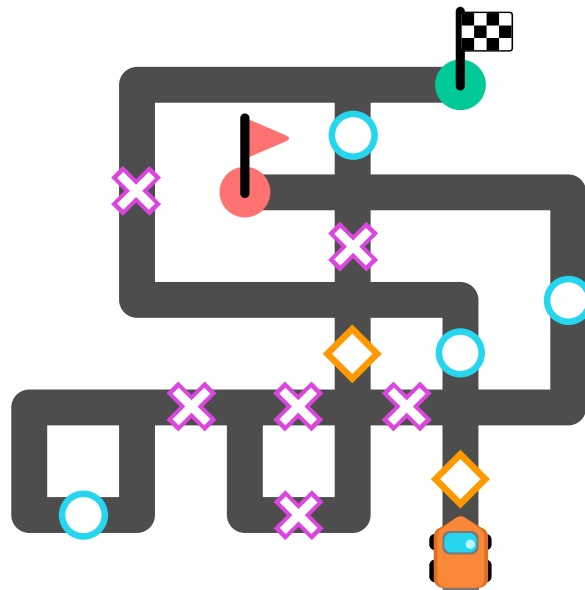
EXPLANATION

Answer



Explanation


The picture shows robot's walk:

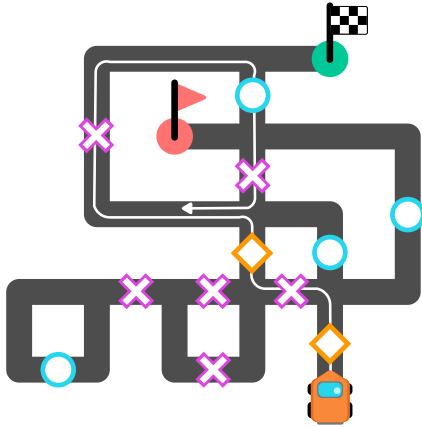
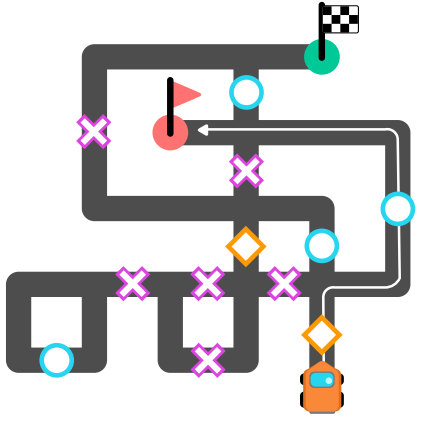
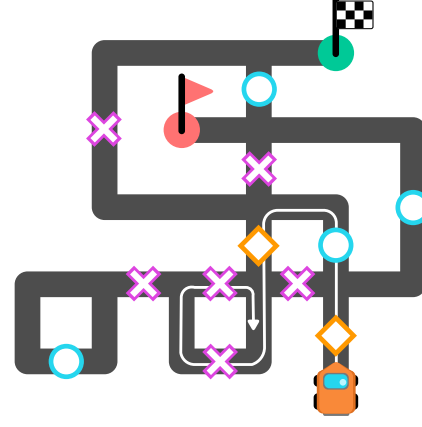



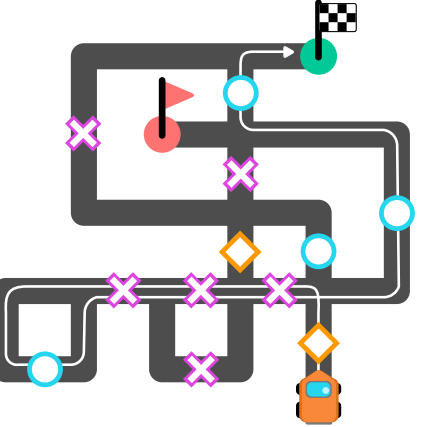
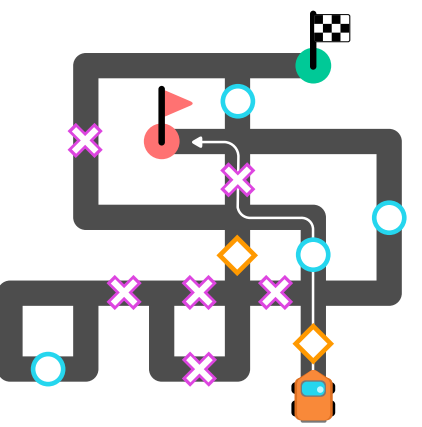
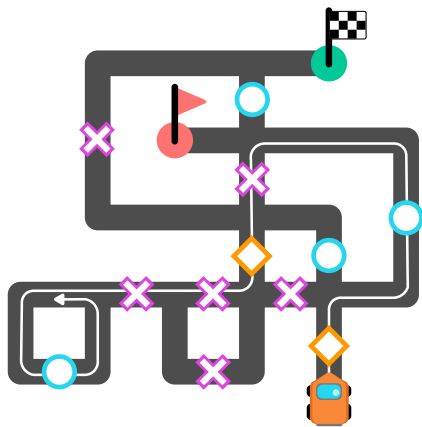







Symbol Reading Robot – cont'd

Years 11+12 Medium

We can use different strategies to solve the problems. One of them is going through every possibility. In this task there are only 6 different ways to interpret the symbols (the pictures show all of them) and only one of them leads to  :

Another strategy is to trace the path the robot could take and assign meaning to symbols as we follow the path. If the robot reaches the red flag or enters a loop, or the meanings cannot be applied to symbols consistently, we retrace to the previous step (in this case intersection) and try another path.



Symbol Reading Robot – cont'd

Years 11+12 Medium

BACKGROUND INFORMATION

The first strategy proposed in the explanation is called *brute force*. This means to consider and check every possibility until one finds the desired result. Sometime there can be many possibilities and considering all of them could be very time consuming and therefore one need to find other strategies.

The second strategy is called *backtracking*. In this technique one incrementally builds candidates to the solutions, and abandons a candidate as soon as one determines that the candidate cannot possibly be completed to a valid solution. The advantage of backtracking with respect to brute force is that you do not have to reconsider the new candidate solutions from the beginning, since you go back only to the step where you made your last choice and continue on from there.

For the given problem brute force may work better, as we have few variables (symbols) and few paths for the robot to take. However in general for larger complex problems, when the number of variables increase and the paths to explore are more, backtracking provides better and elegant solution. Puzzles like Sudoku can be elegantly solved using backtracking.



Cupcakes

Bebras Bakery produces cupcakes for the hard-working hungry beavers in the town. Each cupcake is decorated with three sweet layers. Firstly, each cupcake gets an icing layer, then a toppings layer, and finally a fruit layer. The first example below has red icing, chocolate flakes topping, and a cherry fruit layer. The second example below has blue icing, toasted nuts topping, and a blueberry fruit layer.



On the assembly line, each of the layers is changed from one cupcake to the next as follows:

- The icing layer changes with the following pattern: green $\backslash(\backslash\text{to}\backslash)$ white $\backslash(\backslash\text{to}\backslash)$ red $\backslash(\backslash\text{to}\backslash)$ blue $\backslash(\backslash\text{to}\backslash)$ [repeats again starting with green]
- The toppings layer changes with the following pattern: sprinkles $\backslash(\backslash\text{to}\backslash)$ chocolate flakes $\backslash(\backslash\text{to}\backslash)$ toasted nuts $\backslash(\backslash\text{to}\backslash)$ [repeats again starting with sprinkles]
- The fruit layer changes with the following pattern: cherry $\backslash(\backslash\text{to}\backslash)$ kiwi $\backslash(\backslash\text{to}\backslash)$ strawberry $\backslash(\backslash\text{to}\backslash)$ orange $\backslash(\backslash\text{to}\backslash)$ blueberry $\backslash(\backslash\text{to}\backslash)$ [repeats again starting with cherry]

Barry the Beaver plays a trick in the bakery. He changes the pattern of two of the layers:

- Barry changes the fruit pattern so that each time it skips the next two fruits in the pattern. For example, if an orange piece is placed on a cupcake the next cupcake would have a kiwi piece on top.
- Barry reverses the toppings pattern.

Question

If the 1st cupcake has green icing, sprinkles, and a cherry on top, what will the 6th cupcake look like?

Red, chocolate flakes, cherry

Blue, toasted nuts, strawberry

White, toasted nuts, kiwi

White, chocolate flakes, cherry

Blue, chocolate flakes, blueberry



Cupcakes – continued

EXPLANATION

Answer

White, chocolate flakes, cherry.

Explanation

An approach to find the correct combination is to methodically go through each of the topping orders using the new rules.

The first cupcake has green icing, sprinkles, and a cherry on top. We need to find the 6th cupcake.

Starting with the icing layer the order is: green, white, red, blue, green, white - therefore the 6th has white icing

Moving on to the toppings layer we know the Barry has reversed the order here. The first cupcake has sprinkles. The order is then: sprinkles, toasted nuts, chocolate flakes, sprinkles, toasted nuts, chocolate flakes, - therefore the 6th has chocolate flakes

Moving on to the fruit layer. We know that Barry changes the fruit so that it skips the next two pieces in the pattern after putting one on. We know the first one has a cherry on top. The order is then: cherry, orange, kiwi, blueberry, strawberry, cherry, - therefore the 6th has a cherry layer.

BACKGROUND INFORMATION

This task illustrates the computational thinking concepts of *algorithms* and *pattern recognition*, and the computer programming concept of a *linked list*. Pattern recognition is the concept of finding patterns in the problem that can be followed to find the solution, either in the form of loops in the solution, or reusing parts of solutions from previously solved problems. In this task, the sequence of options for each layer forms a pattern, but also there is a pattern in the way that the application of each layer (icing, toppings, fruit) follows the same fundamental algorithm.

An *algorithm* is a list of instructions. Following instructions is a very important concept in computer science. This is how a computer works - we tell it what to do and it follows these steps. For some programming languages, the order of instructions is very important also. By changing the order, we can change the output of the program. The sequence of ingredients in this task is very important for each layer.

When we write computer programs we need to store data within our program. We use data structures to store data in an efficient and organised fashion. There are many different types of data structures, such as the data structure most relevant for this task: a linked list. A *linked list* is a linear collection of data elements where the order is not given by their physical position in memory. Instead, each element points to the next element, and this allows us to access an given element within the list by traversing the list from the start. Linked lists can dynamically increase in size and, unlike arrays, it is easy to insert and delete anywhere within a linked list. For example, to delete an element anywhere in the list we only need to change what the previous element in the list points to.



Bench Workshop

Beaver Albert makes benches. Each bench must have four legs of equal length. The benches come in a variety of different sizes to suit different customer preferences.

Since Albert finds the materials for bench legs in the woods, he does not always manage to find legs of the same length. In exceptional cases, Albert can shorten one or more legs to any length he wants. When shortening a leg, the remainder from what Albert cuts off cannot be used. Albert currently has the following 32 legs in stock:

Length	10	9	8	7	6	5	4	3	2
Count	3	6	3	3	5	3	3	2	4

Question

What is the minimum number of legs that must be shortened to form eight benches?

EXPLANATION

Answer

The correct answer is 6.

Explanation

When summing up the total number of legs, we can observe there is a count of 32. This means that every leg will need to be used to create 8 benches.

With this in mind, there are two non-obvious rules to get the optimal solution for the case when all legs should be used:

- **Rule 1:** If the count of legs for a length is greater than 4, then create as many benches as possible from that length.

If there are already enough legs at a particular length to make a bench, then shortening them will require at least 4 shortenings for no gain in the number of benches created. Excessive shortening of 4-leg sets therefore cannot be optimal.

Applying Rule 1 to Albert’s current stock creates 3 benches at lengths 9, 6 and 2, respectively.

When we take away all those sets, the count is as follows:

Length	10	9	8	7	6	5	4	3
Count	3	2	3	3	1	3	3	2

Bench Workshop – continued

We can now apply the second rule:

- **Rule 2:** All legs of the same length are either used to make a bench or are shortened.

This rule is obvious for the longest and shortest legs. In our case, legs of length 10 should all be shortened since there are no legs which can be shortened to this length, making it impossible to create a bench of length 10. Conversely, all legs of length 3 will be used to make bench since they cannot be shortened any further.

The number of additional legs needed for each length are as follows:

Length	10	9	8	7	6	5	4	3
Count	3	2	3	3	1	3	3	2
Legs to be added	0	2	3	3	1	3	3	2

Legs of at least three different lengths need to be shortened - by shortening legs of just two lengths it is impossible to get the necessary total number of legs to be added ($3+3 < 2+1+3+1+1+2$).

The least number of legs from three lengths are 6 - from lengths 10, 9 and 6. We include length 10 here as '0 legs to be added' and omit length 3 due to Rule 1.

Shortening legs with length 6 to length 5 completes a 4-set of legs with length 5. Using the five longest legs with length 10 and 9 allows Albert to complete 4-sets of legs 8, 7, 4 and 3. So, in the optimal solution 6 legs should be shortened.

BACKGROUND INFORMATION

An *optimal* solution is a feasible solution where the objective function reaches its maximum (or minimum) value – for example, the most profit or the least cost. A *globally optimal* solution is one where there are no other feasible solutions with better objective function values. A *locally optimal* solution is one where there are no other feasible solutions “in the vicinity” with better objective function values – you can picture this as a point at the top of a “peak” or at the bottom of a “valley” which may be formed by the objective function and/or the constraints.

Dynamic Programming (DP) is an algorithmic technique for solving an optimisation problem by breaking it down into simpler sub-problems and utilising the fact that the optimal solution to the overall problem depends upon the optimal solution to its sub-problems. Optimisation problems are widely proposed and needed in many computing fields which make it a very important concept. DP comes in handy when dealing with such problems to give nice solutions with reasonable complexity, which makes it a very important algorithm to know. This problem is a great example for both concepts as students explore the idea that the optimal solution is not always obvious, and often cannot be found using a *greedy algorithm*. This emphasises the importance of algorithms and programming.





Burrow Business

In a field divided into an (8×8) grid, badgers and rabbits have been randomly assigned burrows:

r		B		B	r		
	B	r				r	
B				r			B
	r	r		B		r	
B			B		r	B	
	B	r					
			r		B		r
B		r	B		r		B

In the diagram left, B = a badger, and r = a rabbit.

The badgers and rabbits get along well with each other. However, if a badger or a rabbit finds themselves in a burrow where there are less of their own kind next to them, then given the opportunity, they will always choose to move.

Examples:

- A badger with none of the eight surrounding squares occupied will stay where they are.
- A badger with one badger and one rabbit next to them and 6 unoccupied squares will stay where they are.
- A badger with one badger and two rabbits will choose to move.

The Burrow Officer is trying to organise the field to satisfy all inhabitants, and has the following moving rules:

1. All badgers are asked if they want to move, starting with the one nearest the top left of the field and working across the field and down a row at a time. When all the badgers have made their moves, all the rabbits are asked if they want to move, in the same way.
2. When a badger or rabbit wants to move, they do so immediately but they have to move to the first available square that satisfies their wishes about who can be next to them. The first available square means looking for squares starting at the top left and working across and down.
3. All badgers and rabbits are only allowed to move once.

Question

After all the badgers and rabbits have moved, which of these fields shows the correct pattern?

	B	B	B	B	r	r	r
B	B		B	B	r	r	r
B	B	B			r	r	
						r	
B					r		
	B						
			r				r
B		r			r		

A.

r	B	r	B	r	B	r	B
B	r	B	r	B	r	B	r
r	B	r	B	r	B	r	B
B	r	B	r				

B.

B	B	B	r	r	r	r	r
B	B	B	r	r	r	r	r
B	B						
				B		r	
B			B				
	B						
			r				
B		r			r		B

C.

B	B	B	r		r	r	r
B	B	B	r	r	r	r	r
B			r	r	r		
B			B			B	
	B						
			r		B		
B		r					B

D.



Burrow Business – continued

EXPLANATION

Answer

	B	B	B	B	r	r	r
B	B		B	B	r	r	r
B	B	B			r	r	
						r	
B					r		
	B						
			r				r
B		r			r		

Explanation

	B	B	B	B	r	r	r
B	B		B	B	r	r	r
B	B	B			r	r	
						r	
B					r		
	B						
			r				r
B		r			r		

A.

r	B	r	B	r	B	r	B
B	r	B	r	B	r	B	r
r	B	r	B	r	B	r	B
B	r	B	r				

B.

B	B	B	r	r	r	r	r
B	B	B	r	r	r	r	r
B	B						
				B	r		
B			B				
	B						
			r				
B	r			r		B	

C.

B	B	B	r		r	r	r
B	B	B	r	r	r	r	r
B			r	r	r		
B			B			B	
	B						
			r		B		
B	r						B

D.

Option A is correct. This is a result of following the given algorithm.

Some care is required because later animals move into empty spaces near the top, which makes the layout fairly dynamic. A human can work out a general layout is fairly easily, but finding the exact result is computationally hard.

As this is multiple-choice there is no need to completely work out the correct answer or even check each of the provided answers completely. It is enough to find any placement that cannot be generated by the algorithm and thus disprove the wrong answers.

To find the answer using this approach, we only need to look at the top left square which starts with a rabbit.

Since rabbits move after badgers, a badger cannot end up in this square. That rules out C and D. We then need to determine if the rabbit would stay or move.

Working across the top row the first Badger stays where it is. The second one chooses to move as it only has one neighbour and this is a rabbit. The first available and suitable space is the second square on the top row next to the first rabbit. This now means that once all the badgers have moved the rabbit in the top left square will choose to move, leaving the square empty. No other rabbits will move into it because it is now unsuitable, having badger neighbors but no rabbits. The only answer with the top left square empty is answer A.



Burrow Business – continued

The other options come about through variations of the rules:

- Answer B is simply assigning burrows in an alternating fashion, but this is not what was asked for.
- Answer C is a result of asking each badger or rabbit to move working from the top left down to the bottom of the field a row at a time. There are badgers who have moved to places with starting rabbits, which cannot happen if the algorithm is followed.
- Answer D is a result of asking all the rabbits to move first instead of the badgers. Otherwise all the rules were met.

BACKGROUND INFORMATION

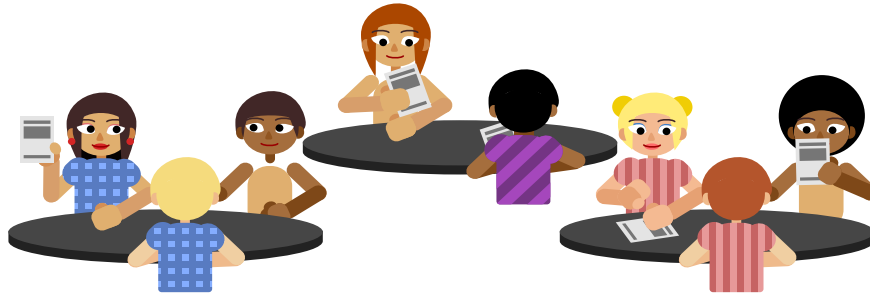
The challenge in this question is about following an algorithm. It is based based on *Schelling's segregation model*. This is a program that models social outcomes given particular choices, namely about where people will live given reasonable choices. It is more sophisticated than the model used in this question and demonstrates that two groups of people will naturally segregate into separate communities without exhibiting prejudice but simply by wanting to have people like them relatively near them. This helps to explain why we have concentrated communities in many cities around the world – groups of people who have the same religion, culture, or even wealth can often be found all living together in areas of a city. If an estate agent chooses to “save themselves some time” by making assumptions and, for example, only offering houses to rich people in areas where most rich people already live, this will make segregation even more extreme.

Although the model used in the question is very simplified it can be seen that using any of the systems in answers A, C or D result in segregation. Only B, which imposes an order, results in an integrated community.



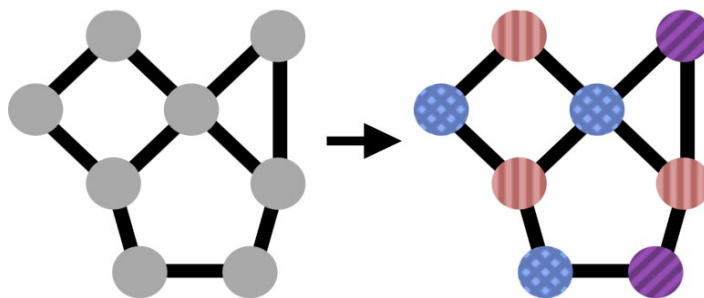
Quiz Night

Eight people usually form three quiz teams:



The teams are organised so that each player knows all of the other players on their team. They also know some of the players on the other teams. Who knows who can be represented by a graph.

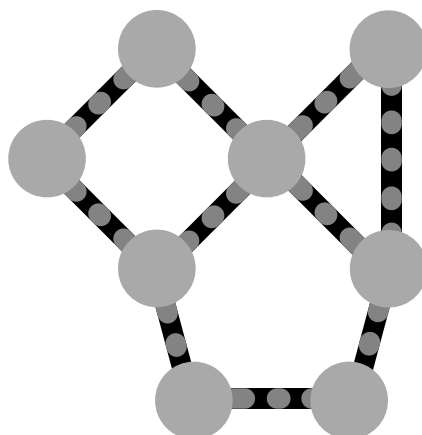
In the graphs below, circles represent people. If there is a line between two people this means that they do not know each other and so cannot be on the same team. The graphs can be helpful when assigning teams by colouring in the circles, for example:



Unfortunately, one of the tables is broken tonight, so only two quiz teams can be formed. At the moment, two teams cannot be formed unless two people are introduced to each other. This introduction can be shown by removing a line on the graph. But who should be introduced?

Question

Select one line from the graph below that, when removed, allows two teams of four to be formed.

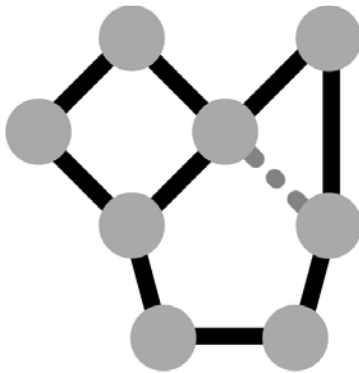




Quiz Night – continued

EXPLANATION

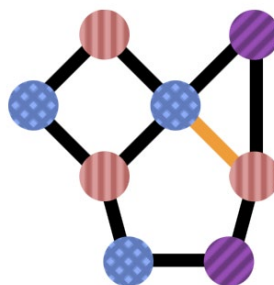
Answer



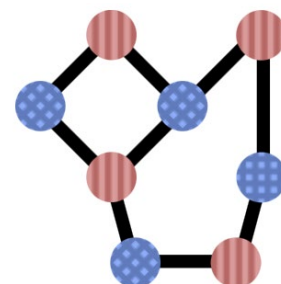
Explanation

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

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

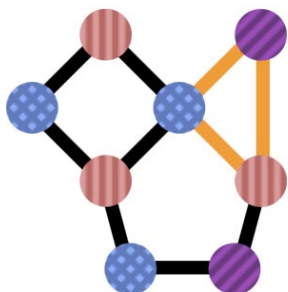


After deleting this edge, we can colour the graph with two colours as shown below right.



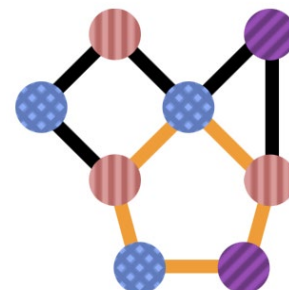
To test that deleting this edge is the only possible choice, we need to consider both the triangle in the upper right and the pentagon on the bottom.

First consider the triangle in the upper right:



If any edge outside of this triangle is deleted, we still need three colours just for the three vertices of that triangle. So one of these three edges needs to be deleted.

Now consider the pentagon on the bottom:



If any edge outside of this pentagon is deleted, then it is impossible to colour all five of its vertices with only two colours. To test this, we can cycle clockwise through the five vertices, alternating colours for each one. But when we reach the last vertex, it will have the same colour as the first vertex because the number of vertices in the cycle is odd.

Therefore we need to delete an edge that destroys both the triangle in the upper right and the pentagon on the bottom at the same time. There is only one edge that is shared by both shapes, leading us to the only possible answer.




BACKGROUND INFORMATION

Many real-world problems can be reframed as coloured vertices on a graph. An example is a graph where the vertices are students and an edge between two students shows that they can't be placed in the same group. If we colour the vertices with k colours, this can be seen as assigning every student to one of k groups. Such a colouring is *proper* if any two vertices directly connected by an edge have different colours. Often, we just say *colouring* when we mean a *proper colouring*. An edge is sometimes called *critical* if deleting it makes a *proper colouring* with fewer colours possible. In the example, this means that if the corresponding two students are introduced and can then work together, then having fewer groups becomes possible.



Counting by Nodding

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

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

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

Question

A skeleton of the control program is shown below.
Complete the program by dragging the appropriate condition blocks from the right to the gaps in the program.

Setcountto0

Repeat until

Wait until

Wait until

Increasecountby1

Delivercounttickets

nose > 0

nose > 0.8

nose > 1.2

nose < 0.8

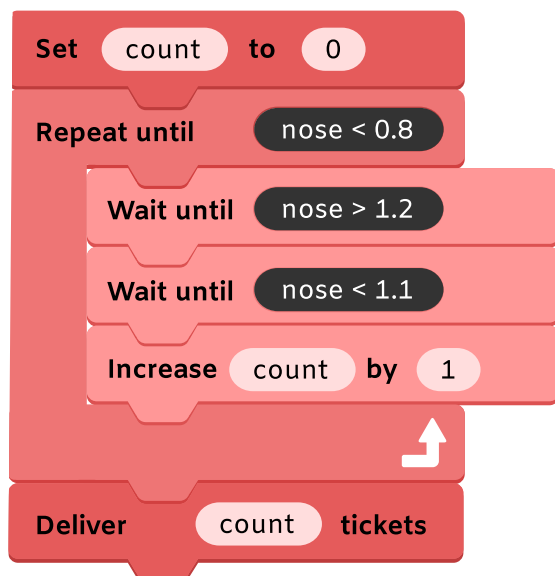
nose < 1.1



Counting by Nodding – cont'd

EXPLANATION

Answer



Explanation

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

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

When the loop is finished (because the person has raised their head such that $nose < 0.8$), the variable *count* contains the number of nods and count tickets are delivered.

The program has to use inequalities, as in real life applications, it will be difficult for the user to get the exact value of $nose = 1$ to signify the end of each nod. Similarly, the values used to trigger an action should be significant enough that it can be considered a deliberate action of the user. For example, if $nose > 1$ is used to measure a nod, minuscule head movements can be counted as one nod even though it is not the intention of the user.

BACKGROUND INFORMATION

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

In the task, the program shown is described as a “skeleton of the control program” because it is not a finished program. It will need further real life testing to check that the variables chosen work in a good variety of situations, that it produces reliable data in a high percentage of occasions, and that there are no bugs that lead to the vending machine behaving in unexpected ways.



Log Sort

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

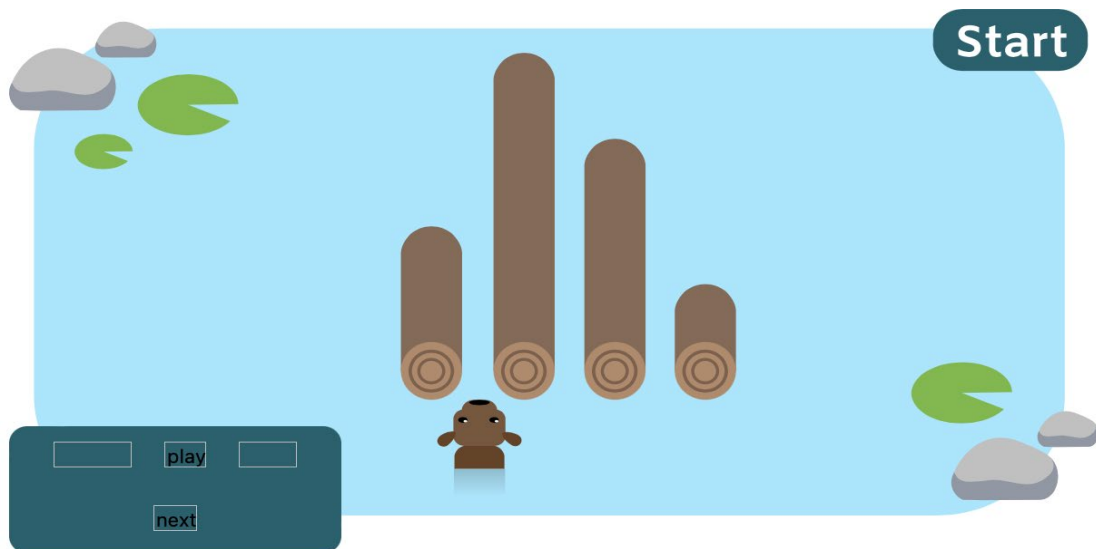
Hamid knows the logs can be sorted in the following way, no matter how the logs come in initially:

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

Repeat the following until you are on the right of the rightmost log:

- If the log on the left is smaller than the right log: move to the right by one log.
- If the log on the right is smaller than the left log:
 - swap these logs;
 - unless you are at the starting position: move to the left by one log.

See how Hamid sorts 4 logs in this way. In this example, Hamid has to move 9 times.



The number of times Hamid has to move to sort a group of logs depends on how the logs come in initially. In the worst case, Hamid would have to move 25 times to sort 6 logs. In the best case (when the 6 logs are already sorted) Hamid would still have to move 5 times.

Question

Which of the ranges below is the *smallest* range that will always include how many times Hamid has to move when sorting all starting arrangements of 60 logs?

0...60

10...90

59...300

59...3,600

59...216,000



Log Sort – cont'd

EXPLANATION

Answer

59...3,600.

Explanation

[0...60] is wrong. Even in the best case, when the logs are sorted, Hamid has to make 59 moves. More moves are required if the logs are out of order quickly taking the moves required to beyond 60.

[10...90] and [59..300] are also wrong. To prove it, we need to explore the worst case, when logs are sorted in the opposite order. In this case, Hamid reaches a log on the k^{th} position and moves it to the first (leftmost) position, then goes for a log on the $(k+1)^{\text{th}}$ position. So for the log on the k^{th} position, we need to move $k-2$ times right to reach it, and $k-2$ times left to put it in the beginning. Thus, we obtain the sum $2(1+2+\dots+58)$ and we need to add 59 moves from the leftmost to the rightmost position at the end of the algorithm. The sum is exactly $592=3481$. This number does not belong to any of these two ranges.

[59...3600] is correct. To see it, we need to prove that the worst case is really the worst one. When Hamid reaches the log in the k^{th} position, all previous logs are already sorted properly, so he needs only to put this new log in the correct position among the previous ones. Then he goes to the log on the $(k+1)^{\text{th}}$ position. So, the smaller is the log on k^{th} position, the greater number of move it requires.

[59...216,000] does include how many times Hamid will have to move when sorting the logs no matter what their starting positions but this range is not the smallest range that does this.

BACKGROUND INFORMATION

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

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

When speaking of an algorithm we always need to consider how 'fast' it is, i.e. the number of operations it requires to sort items in the worst or most disordered case, depending on the number of elements needed to be sorted. For this algorithm, if we have n objects, we need approximately n^2 operations. This is called a quadratic relationship. The other answers in this task represent other possible relationships: constant (independent) for [0...60], linear for [10...90], log-linear for [59...300] and really big (actually increasing by the power of 3) for [59...216,000].

This relationship for a particular algorithm is called the *algorithm complexity* and is studied in computational complexity theory.



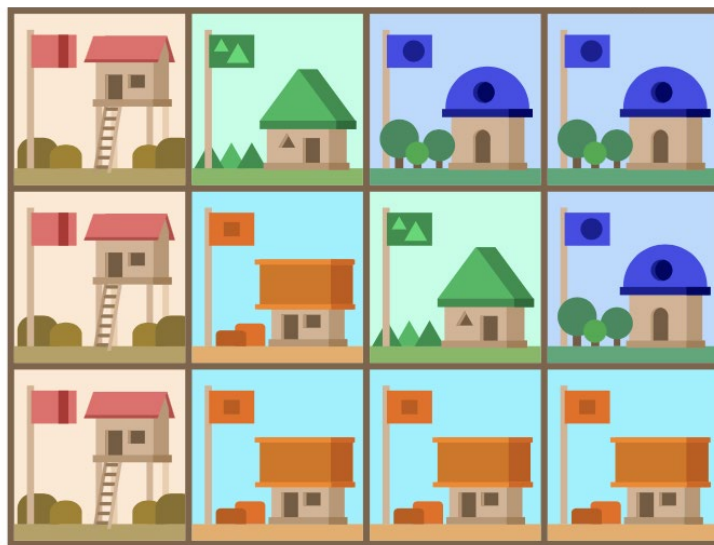
Unification

In ancient Beavaria, there lived four tribes consisting of several villages. Each tribe had their own flag as shown in the picture below.

One day, the tribes decided to unify. However, in order to not cause chaos, it was decided that only two tribes can be unifying at the same time.

The time needed to unify two tribes, in months, is equal to the total number of villages in these two tribes.

After this, the two tribes become one single tribe, and the unification process is repeated until there is only one unified tribe remaining.



Question

What is the minimal amount of months needed for the tribes to unify?

23

24

25

26

27

EXPLANATION

Answer

24.

Explanation

The optimal strategy to minimise the total number of months needed to unify all the tribes is to minimise the number of times each village is included in the unification processes. This can be done by merging the largest tribes last, as the largest tribes with the greatest number of villages will then only be added the least amount of times. In order to do this, each unification step should happen between the two tribes with the fewest villages.



Unification – continued

This is illustrated in the table below:

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

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

BACKGROUND INFORMATION

This challenge is an example of an *optimisation problem*, a task whose goal is to come up with a strategy that maximises or minimises a certain quantity, subject to some constraints. Optimisation problems are ubiquitous in our everyday lives: finding the shortest route to a destination, creating a schedule that accommodates the most number of non-overlapping activities, and so on. There are several ways to approach solving an optimisation problem, and these include *greedy algorithms*. *Greedy algorithms* rest on the assumption that making the best choice at each stage (*local optimum*) will result in the best final outcome (*global optimum*). In this problem, this assumption is satisfied: tribes have to minimise the number of months for each unification in order to minimise the number of months for the entire unification process.

It must be emphasised, however, that the greedy paradigm is not a universal solution to all types of optimisation problems. Nevertheless, it usually provides a decent approximation within a reasonable time.



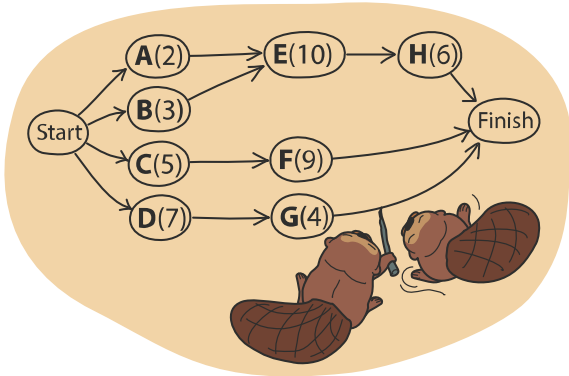
Two Beavers are Working

Two beavers'are building a dam and need to do eight tasks (cut trees, remove branches, float wood, assemble trunks etc). The beavers decided to record these tasks in shorthand as follows:

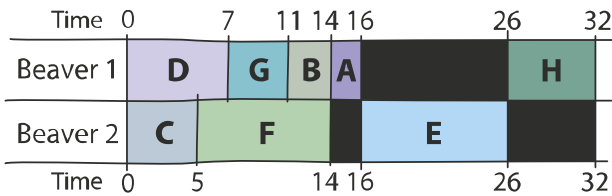
A(2), B(3), C(5), D(7), E(10), F(9), G(4), H(6).

Each number in brackets indicates the number of hours to do that task.

Some tasks must be completed before others can be started, as shown by the arrows in the diagram below. The beavers work in parallel, each taking different tasks.



The beavers try this strategy: Choose the biggest task currently possible.
The beavers end up working on the tasks in this order:



The beavers complete the dam in 32 hours. However, it is possible to complete the dam in a shorter time with a different strategy.

Question

Create the schedule for the *shortest* time for the beavers to build the dam below.
Drag the tasks, from the top row, down to each beaver's task box to create the schedule.
Tasks in grey need other tasks to be completed before they can be assigned.

A

B

C

D

E

F

G

H

A

B

C

D

E

F

G

H

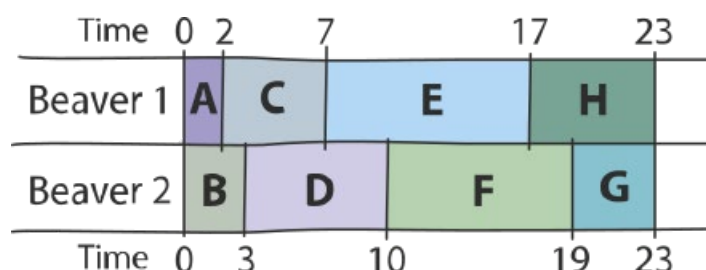
Beaver 1	
Beaver 2	



Two Beavers are Working – cont'd

EXPLANATION

A Possible Answer



Explanation

The picture in the tasks shows one possible schedule of the two beavers. We can see that the first beaver is idle for a relatively long time (8 hours), and the second beaver is idle for 6 hours. It would be better if they could be working all of the time.

The strategy will use here is to make sure that the two largest tasks E(10) and F(9) are not done by the same beaver. Above is the picture of one particular schedule that allows all the tasks to be completed in 23 hours with no idle time for either beaver.

BACKGROUND INFORMATION

For some problem instances, the beavers' strategy ("choose the biggest remaining") will yield the shortest time. For other problem instances (such this one) the strategy of partitioning the largest tasks seems to work. However, for each of these approaches we can find a problem instance for which it will not work so well. This is because, unless there are some special restrictions on the problem, the only guaranteed way of finding the schedule yields the shortest time is to try all possible allowed schedules! This is impractical in real-world situations; it might take more computer resources to find the perfect schedule than for one beaver to build the whole dam themselves!

In this task, a problem instance was carefully chosen for which the beavers' strategy ("choose the biggest remaining") did not work. So it made it quite difficult; in this case one has to consider the whole problem instance instead of blindly following a simple strategy. However, for many problem instances, the beavers' "greedy" strategy can be good enough, and has the advantage that it is very quick to make a schedule and get the beavers working straight away.

Carefully finding a problem instance to cause a strategy to perform poorly (as was done in the preparation of this task) is a real art in computer science, that requires one to deeply understand the strategy. It is a necessary skill if one wants to determine the worst-case running time of a computer program, for example, which is part of algorithm analysis and is used in the field of computational complexity theory.

We would like to thank the International Bebras Committee and community for their ongoing assistance, resources and collaborative efforts. Special thanks to Eljakim Schrijvers, 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