Bebras Australia
Computational Thinking Challenge

2024 Solutions Guide Round 2

Primary Divisions | **Years 3 to 6**

AUSTRALIAN MATHS TRUST

CSIRO

Bebras

COMPUTATIONAL THINKING CHALLENGE

Bebras Australia
Computational Thinking Challenge

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

Bebras is a fun and engaging computational thinking challenge for students in Years 3 to 12. It is an international challenge that involves over 2.9 million students from 60 countries.

Bebras creates opportunities for students to engage in activities that use and develop their critical and creative thinking and problem-solving skills essential to further learning.

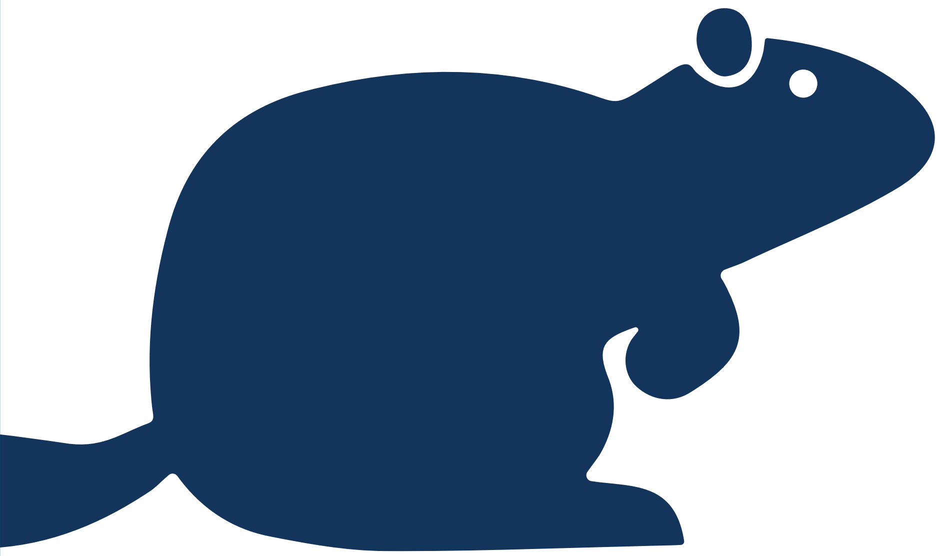
Coding skills are not required to complete the Bebras Challenge. The challenge is open twice a year for three weeks and each round has different questions. Students can participate individually or in teams of up to four.

Lithuanian for beaver, Bebras was the name chosen by the founder of the challenge, Professor Valentina Dagiene from the University of Vilnius, in honour of the animal's collaborative nature and strong work ethic.

The Challenge is coordinated by the International Bebras Committee which meets annually to assess potential questions and share resources. Bebras Australia began in 2014 and was delivered by CSIRO until 2023. From 2024, Bebras Australia is delivered by the Australian Maths Trust.

With support from CSIRO, the AMT provided the Bebras Challenge free of charge for Australian schools in 2024.

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



Contents

What is a Solutions Guide?

Within this Solutions Guide you will find all of the questions and tasks from Round 2 of the Primary divisions of the 2024 Bebras Australia Computational Thinking Challenge.

On each page you will find a question, answer, an explanation, and background information explaining the skills and key curriculum concepts featured.

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.

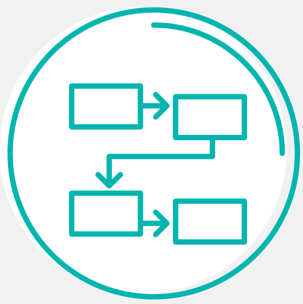
Computational Thinking uses a number of skills, including:

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.

Visit the AMT website for more Bebras resources https://www.amt.edu.au/bebras

Computational Thinking skills alignment

Question	Difficulty	Decomposition	Pattern Recognition	Abstraction	Modelling & Simulation	Algorithms	Evaluation
Middle Primary Year 3 & 4							
Friendly Aliens	Easy	●	●	●			●
Coded Ages	Easy	●		●			●
Carnival Masks	Easy	●	●	●		●	●
Hidden treasure	Easy	●		●	●	●	
Hamburger Shop	Easy	●		●	●	●	●
Walking Logs	Medium	●		●			●
Mazes	Medium	●		●	●	●	●
Treasure Island	Medium	●		●	●	●	●
Travel by Coin	Medium	●		●	●	●	●
Encrypted Message	Medium	●		●		●	●
Favourite Drinks	Hard	●		●		●	●
Maze game	Hard	●		●	●	●	●
Sprinklers	Hard	●	●	●		●	●
Gift selection	Hard	●		●		●	●
Robot on the path	Hard	●		●	●	●	●
Upper Primary Year 5 & 6							
Hamburger Shop	Easy	●		●	●	●	●
Walking Logs	Easy	●		●			●
Mazes	Easy	●		●	●	●	●
Treasure Island	Easy	●		●	●	●	●
Encrypted Message	Easy	●		●		●	●
Favourite Drinks	Medium	●		●		●	●
Maze game	Medium	●		●	●	●	●
Sprinklers	Medium	●	●	●		●	●
Gift selection	Medium	●		●		●	●
Robot on the path	Medium	●		●	●	●	●
Sealed Letters	Hard	●		●			●
Watercolour	Hard	●		●	●	●	●
Snail Compress	Hard	●		●		●	●
Delivering Mail	Hard	●		●	●	●	●
Jumping Together	Hard	●	●	●	●		●

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

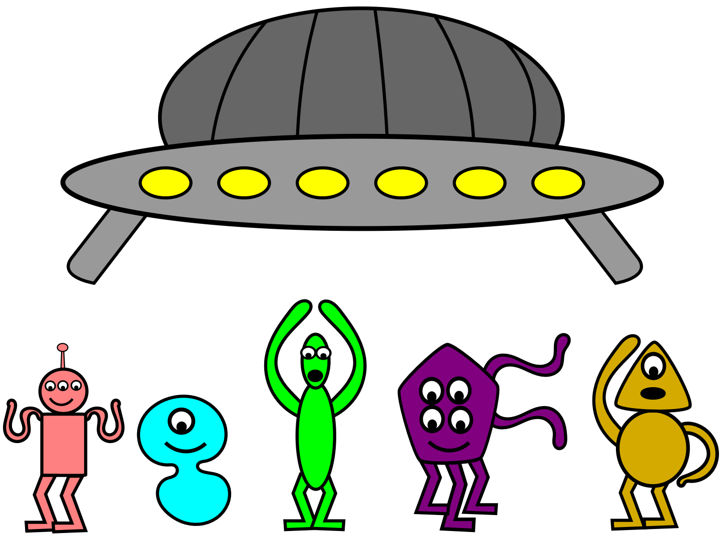
Question	Difficulty	Abstraction	Data Collection	Data Representation	Data Interpretation	Specification	Algorithms	Implementation	Digital Systems	Interactions	Impacts
Middle Primary Year 3 & 4											
Friendly Aliens	Easy	●	●		●		●				
Coded Ages	Easy	●		●			●				
Carnival Masks	Easy	●				●	●				
Hidden treasure	Easy	●					●				
Hamburger Shop	Easy	●		●		●	●		●		
Walking Logs	Medium	●		●	●				●		
Mazes	Medium	●				●	●	●			
Treasure Island	Medium	●		●		●	●	●		●	
Travel by Coin	Medium	●		●		●	●	●		●	
Encrypted Message	Medium	●				●	●	●			
Favourite Drinks	Hard	●	●	●		●	●	●			
Maze game	Hard	●					●	●		●	
Sprinklers	Hard	●		●	●	●	●	●			
Gift selection	Hard	●					●	●	●		
Robot on the path	Hard	●				●	●	●	●		
Upper Primary Year 5 & 6											
Hamburger Shop	Easy	●		●		●	●		●		
Walking Logs	Easy	●		●	●				●		
Mazes	Easy	●				●	●	●			
Treasure Island	Easy	●		●		●	●	●		●	
Encrypted Message	Easy	●				●	●	●			
Favourite Drinks	Medium	●	●	●		●	●	●			
Maze game	Medium	●					●	●		●	
Sprinklers	Medium	●		●	●	●	●	●			
Gift selection	Medium	●					●	●	●		
Robot on the path	Medium	●				●	●	●	●		
Sealed Letters	Hard	●	●			●					●
Watercolour	Hard	●					●	●		●	
Snail Compress	Hard	●		●			●	●	●		
Delivering Mail	Hard	●				●	●	●	●		
Jumping Together	Hard	●					●	●	●	●	

2024 Bebras Challenge
Round 2 | Middle Primary Year 3 & 4

Friendly Aliens

Alice spots a spaceship that has dropped off five friendly aliens. She wants to count how many look similar to a human. She counts an alien if they have:

- two arms, and
- two legs



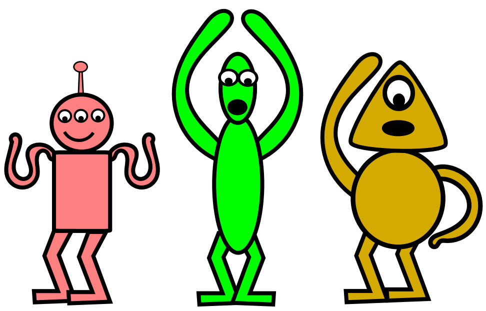
Question

How many aliens would Alice count?

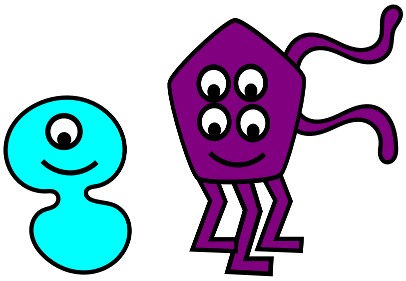
1	2	3	4	5

Explanation

The correct answer is 3.



The three aliens shown here all have 2 arms and 2 legs, and so are counted by Alice.
The other two aliens are not counted because the first one has no arms or legs, and the second one has two arms and three legs.



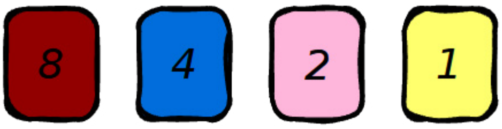
Background information

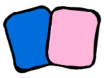
Image recognition is an important Computer Science concept. We often apply technology to try to find certain objects in a picture or video according to criteria about the object. Here, we are wanting to find a subcategory of aliens that have 2 arms and 2 legs while excluding the rest.
In practice, image recognition like this is used from medical imaging to defect detection and security systems.

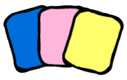
Decomposition: Alice would break down the problem into smaller, manageable parts. In this case, she needs to identify the characteristics of the aliens and determine if they meet the criteria for being counted.
Pattern recognition: Alice needs to recognize the pattern or rule for counting the aliens. The pattern in this case is having two arms and two legs.

Coded Ages

Zoe shows Steven how to use the cards below to tell their ages.

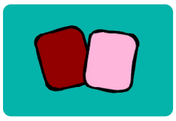
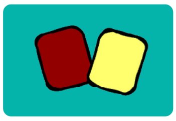
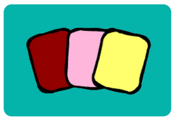
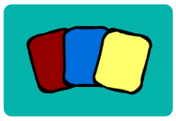


Zoe is 6, so she uses the cards  (4 + 2 = 6).

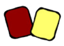
Steven is 7, so he uses the cards  (4 + 2 + 1 = 7).

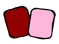
Question

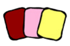
Big sister Kali is 9. What cards will she use?

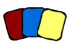
   

Explanation

The correct answer is option  (8 + 1 = 9).

Option  is incorrect because it represents age 10 (8 + 2 = 10).

Option  is incorrect because it represents age 11 (8 + 2 + 1 = 11).

Option  is incorrect because it represents age 13 (8 + 4 + 1 = 13).

Background information

Different representations for numbers is common in computer science. In a single computer there may be many different representations for numbers all being used at the same time. For example, ages (integers), fractions (real values), and dates would all have different representation rules. Understanding the number representation in this task is a stepping stone to understanding positional notation for numbers (decimal or binary, for example) which makes numerical calculations (e.g. addition, multiplication) very efficient with pencil and paper, and also in a computer. Large international organisations such a ISO and IEEE spend a lot of effort agreeing the exact representations for different types of numbers (called number formats) in computers.

Inspired by the binary number system, the numbers on the cards in this task are carefully chosen to be powers of 2 so that only one of each card colour is needed to represent numbers between 0 and 15. However, compared to a positional notation such as binary, the number representation in this task, based on the colour of the cards, is more flexible in that the order of the cards does not matter. But, it has two main disadvantages that makes it impractical. Firstly, humans and computers will find it difficult to distinguish between the different colours if we try to represent very large numbers. Secondly, for efficient numerical calculations we would need to convert the colours into a different representation.

Representation is used in this task as numbers are encoded into a format using coloured cards.

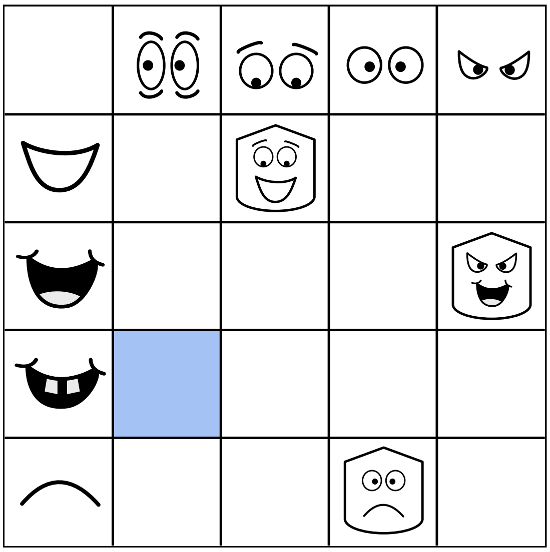
Decomposition can be used in this task to determine one at a time (starting from the largest) which powers of two are present in an arbitrary number. By combining the resulting cards we have the final representation of numbers in this number format.

As with many multiple choice questions, **evaluation** can be used to check (by addition) which of the answer options is the correct one.

Algorithmic thinking is implicit in the task as using Zita's representation system requires an algorithm to encode and decode decimal numbers.

Carnival Masks

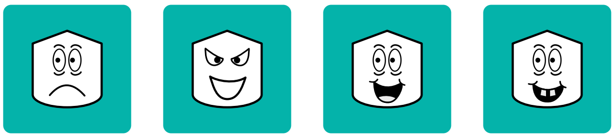
A mask-making machine uses different eyes and mouths to create new combinations. Examples of some of the masks created are shown in the following table:



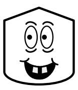
The first row and first column show the eyes and mouths available. The rest of the table shows some of the possible masks.

Question

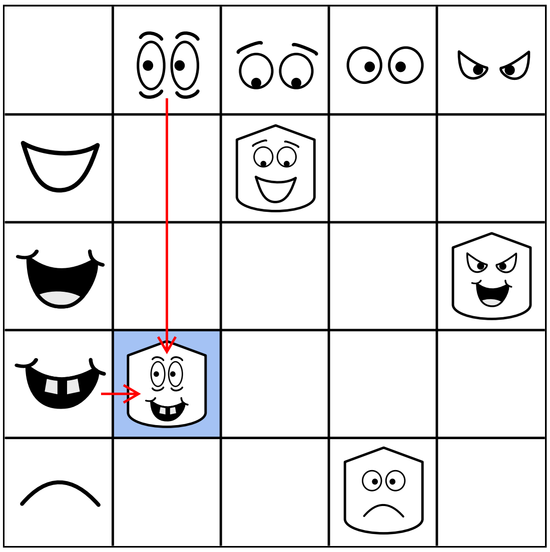
Which mask will be created in the blue square?



Explanation

The correct answer is  .

Each of the masks from row 3 (without counting the header) have the same mouth and thus share that characteristic. Each of the masks from column 1 (again, not taking the header into account) share the same eyes and thus that is the characteristic they share. This presents a way to organise elements according to their shared characteristics and work with them to create new combinations.



Background information

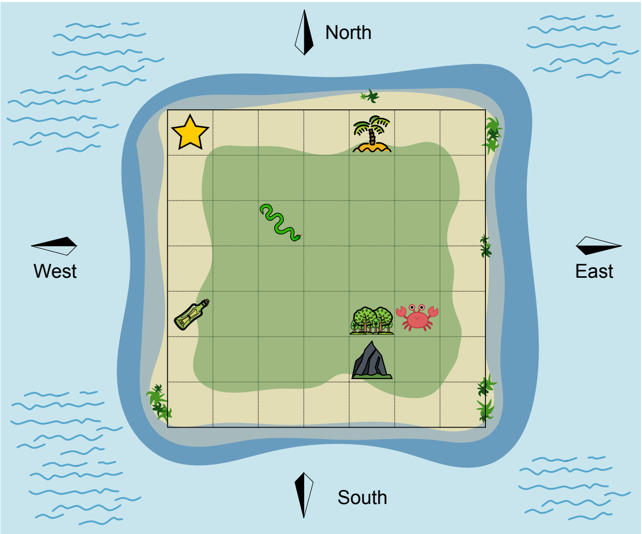
Database design in information system design involves extracting properties that compose the entire system from a real-world problem. In the task, facial features such as eyes and mouths are selected as properties and stored in the database. These properties, sometimes referred to as dimensions, allow for the production of diverse faces through their combination. In this task, two dimensional database is represented as a table, organized into rows and columns. In more complex real-world scenarios, particularly in data warehousing and online analytical processing, a multi-dimensional database is constructed and utilized to manage and analyze data effectively.

This task works on pattern recognition, which is a key concept in computational thinking. Pattern recognition refers to the ability to find similarities or regularities in a data set. It is an analysis process that uses computational techniques to identify patterns in large amounts of information and extract useful insights from them. Pattern identification can be applied in a wide variety of areas, from financial fraud detection to speech recognition and image classification. In short, it is a powerful tool to understand and analyze information in a more efficient and effective way.

Hidden treasure

A group of adventurers are looking for a very special treasure. They found a map and instructions for how to find it.

Start	
Go 4 squares South	
Go 4 squares East	
Go 1 square South	

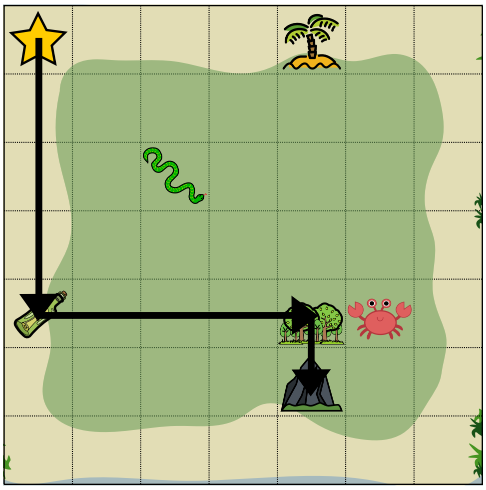


Question

Where is the hidden treasure?

Explanation

The adventurers follow the sequence of instructions: They start from the star. Then, they go down 4 squares where they find the bottle with a message. Afterwards, they move 4 squares to the right and come to the forest. Lastly, they go one square down reaching the gray mountain.



Background information

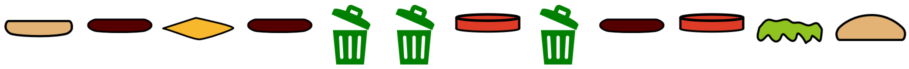
Sequences, order, and algorithms are all very important for computer science. Sequences are a set of things that are arranged in a specific order. Order is important because it helps us make sense of things. For example the letters in the words.

Algorithms are like sets of instructions that tell a computer what to do. In our task, we also use a set of instructions (e.g. "Go 4 squares to South"). Thus, the archaeologists have to apply the given algorithm to find the tresure.

Algorithms are important because they help computers do all sorts of things, from basic math to complex tasks like recognizing faces. Computers need to be able to make sense of information and algorithms are the instructions that tell the computer how to do that. By following the steps in an algorithm, a computer can take in information, process it, and give you a result.

Hamburger Shop

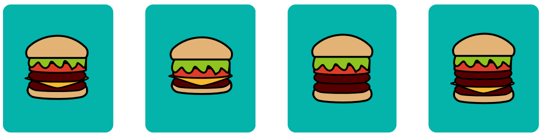
Masahiro wanted to make a tasty hamburger for his mum and tried an experiment. Yuki observed him and recorded the order in which Masahiro added and took away items. Here is what she noted down:



The picture  means that Masahiro took away one item from the top of the hamburger.

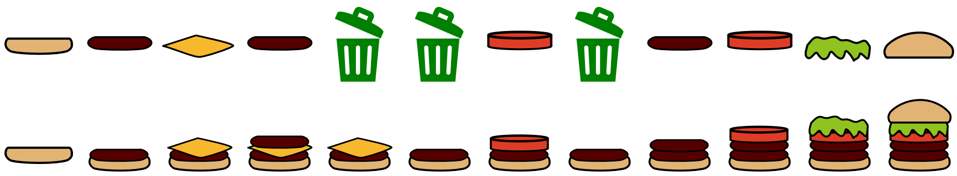
Question

Which of the following hamburgers did Masahiro make?



Explanation

The table shows all steps of the hamburger creation. The top row shows the sequence of actions. The bottom row shows how the hamburger looks after each action in the top row.



Therefore the answer is  .

Masahiro added 9 items and has taken away 3 items, so the hamburger has to consist of 6 items. The answer a) is wrong because this hambunger has only 5 items.

The answer b) is wrong because this hamburger contains cheese but cheese has been taken away.

The answer d) is wrong because this hamburger contains two tomatoes but one of the tomatoes has been taken away.

Background information

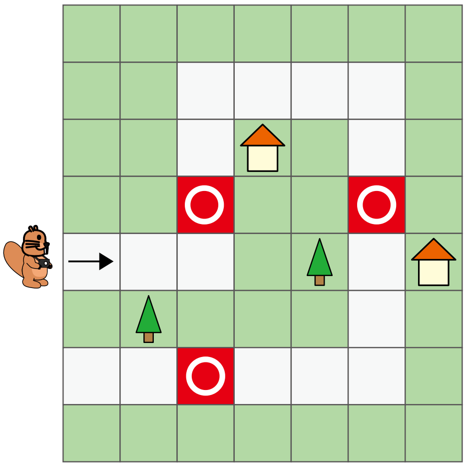
In this task, a hamburger is made using two operations; adding an item and removing the topmost item. The data structure managed by these two operations is called a "stack" in computer science. The main characteristic of the stack data structure is LIFO (Last In First Out), where the last object put in is the first object taken out. The stack data structure is used in many types of computer programs, such as providing the redo/undo functionality in editing software.

This task is algorithmic, the solver has to find the final state after running the given algorithm.
Abstraction is also present in this task. The icon to take away an ingredient represents a more general instruction, rather than the ingredient icons which represent their own specific item to add.

Walking Logs

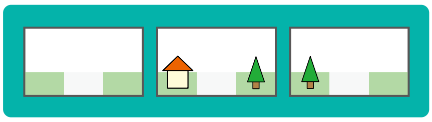
Wendy goes for a walk along the same path every day. One day Wendy took a camera on their walk and stopped 3 times to take photographs of the path ahead. Each photograph shows the square directly in front of Wendy, and the objects to either side of that square.

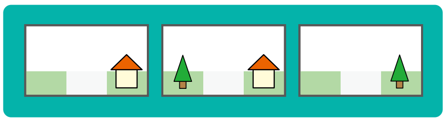
This map shows the path that Wendy walks along. Wendy starts on the square marked with an arrow, moving in the direction indicated until they reach the end of the path. The places that they took photographs are marked on the map with a white circle.

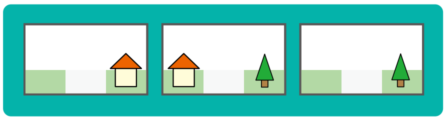


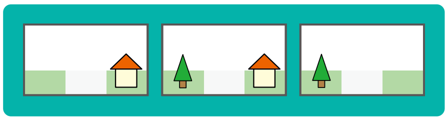
Question

Select which 3 photographs were taken by Wendy.







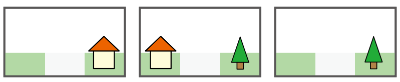


Explanation

Wendy would be facing north in the first photo-taking spot, so the photograph should show the house on the right side of the photograph.

Wendy would be facing south in the second photo-taking spot, so the photograph should show the house on the left side of the photograph and tree on the right.

Wendy would be facing west in the third photo-taking spot, so the photograph should show the tree on the right side of the photograph.

Therefore the answer is .

Background information

Object recognition is an important topic in computer science. An artificial intelligence system can be trained to recognize objects like houses, trees and road surfaces. Probably your smartphone camera is able to recognize faces and indicates them with a frame. An autonomous car must recognize people, cars, traffic signs and other objects in its environment, which are important for its decisions where to go. To help blind and visually impaired persons, modern web media include Automatic Alt Text (AAT). This technology uses object recognition to automatically create verbal descriptions (alt texts) of images.

This question can be approached using abstraction, as there is a lot of additional information in the question that is not needed to find the answer.

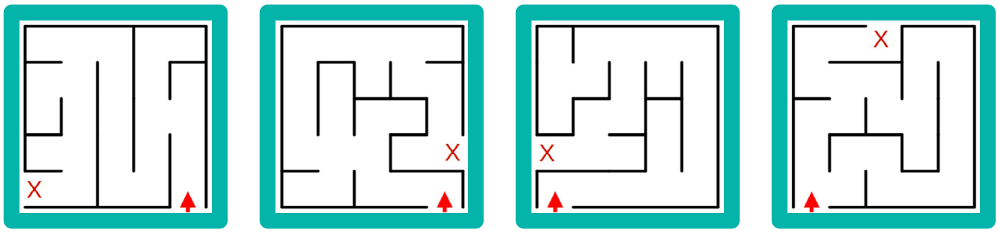
It can also be approached using modeling and simulation skills, with students mapping out the path the beaver walks, and considering what can be seen from each location.

Mazes

Ben likes to explore garden mazes. However, every time Ben wonders whether he has gone through every path in the maze. One day, he learns “The right hand” rule. This rule means that while walking through a maze, you always have your right hand touching the right hand wall. The right hand rule guarantees that when you enter a maze, you can always find an exit. But, the right hand rule does not guarantee that you get to visit every path in the maze.
Below you see four mazes, with an arrow indicating where Ben enters the maze and an X showing the exit.

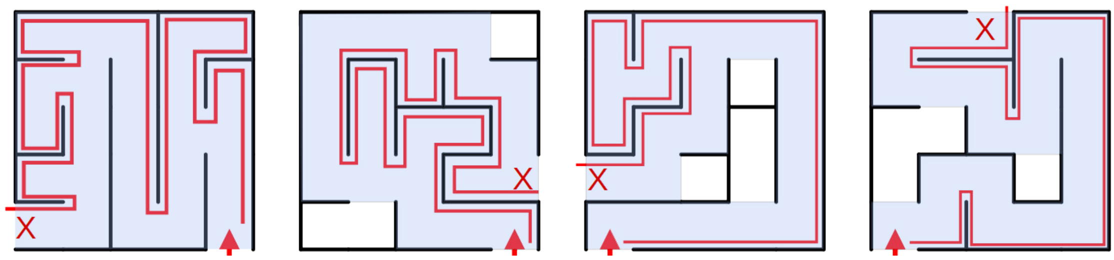
Question

Select the maze where Ben will walk on every path of the maze when following the right hand rule.



Explanation

The correct answer is A. Following the right hand rule, Ben visits every part of the maze. Below you can see the walls that Ben touches through the “Right hand” rule (red line) and the paths Ben covers using this rule (blue shade). In Option A, Ben walks through all paths.



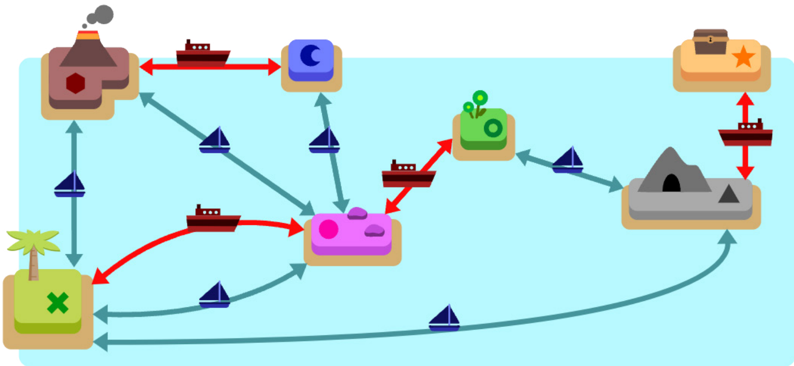
A	B	C	D

Background information

The ‘right-hand’ rule is an algorithm that you can use to go through a maze. The more generic version of this algorithm is called ‘wall follower’ because you basically keep following one wall. The wall follower is a depth-first search algorithm for mazes.
Algorithms are specific sets of instructions to achieve a specific result. In this case the result is: find an exit after you have entered.
There are many fun algorithms to explore when looking at mazes. For example, if you want to know how far each position in the maze is from the entrance you would use a ‘flood fill’ algorithm.

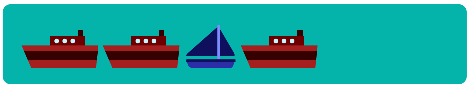
Treasure Island

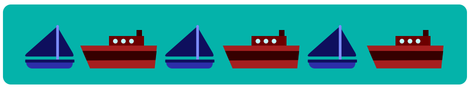
Bill is on vacation in the Island Kingdom. There are two types of ships sailing between the islands: and . You can see their routes in the picture. When people visit the islands, they pick a ship to go to the next island.

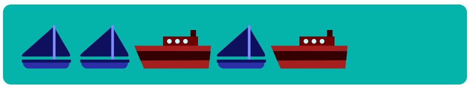


Question

Which of the following trips starting from Island **cannot help** Bill to reach Island ?







Explanation

Given that Bill departs from Island .

The route of option A is:

The route of option B is:

The route of option C is:

It is impossible to finish on island using the path of option D. The route of option D finishes on island or island

For example:

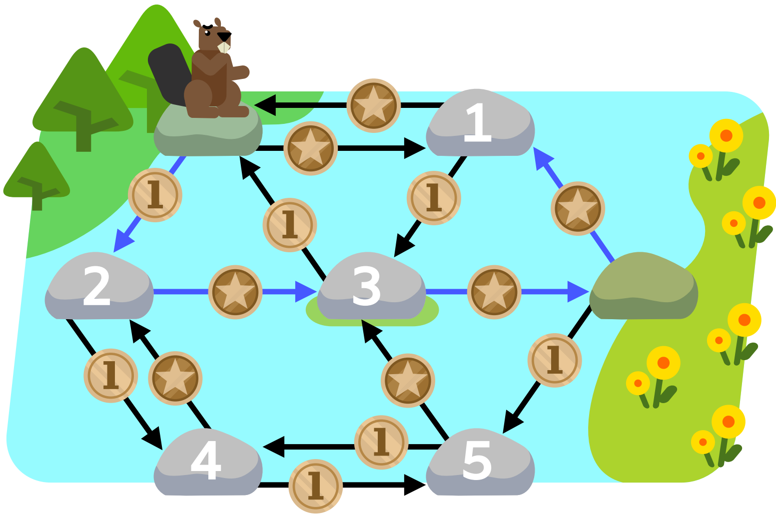
Only the route of option D cannot reach Treasure Island.

Background information

To solve problems using computers, we first need to represent the problems in a way that computers can understand. In this particular case, we are dealing with transportation networks, which can be represented as a graph. A graph is a type of data structure that shows how different things are connected to each other.

By using a graph, computers can help us solve various problems. For example, they can find the most cost-effective route between two places, determine the shortest path, solve the problem in communication network, and much more. In this task, students are asked to finding all possible routes from Island ✖ to Island ★ using the graph representation.

Travel by Coin



Flipper is jumping from stepping stone to stepping stone according to a map he has. He starts at the stone in the upper left corner (see the image to the right) and travels from there according to the result of a coin toss. One side of the coin, called **heads (H)**, has a symbol , another side of the coin, called **tails (T)**, has a number 1 .

Standing on his start stone, if the coin lands on H he will travel to stone 1, if it lands on T he will travel to stone 2.

For each stone Flipper lands on, there are instructions for where to go from there, depending on whether he throws H or T.

Shown on the right in blue is the path Flipper would take if he threw the coin sequence **THHH.**

Question

Which coin sequence would result in Flipper ending his journey at the empty stone all the way on the right, assuming he starts at his start stone?

TTHTHHT | HTHHHT | TTTHTTHH | THTTHTTHT

Explanation

Answer C is the correct answer.

This path can be split into 2 parts. The first part **TTTHT** brings Flipper right back to the starting point. From the starting point a sequence of **THH** will bring Flipper to the finish stone.

Option A stops at start. Option B stops at stone 2. Option D stops at start.

There are many other paths to the finish stone.

Background information

The map shown in the diagram can be considered to be a deterministic finite automaton (DFA). The beaver starts in the **start** state and as he flips coins he moves from stone to stone. Each stone corresponds to a state in the DFA. Usually what controls the movement from state to state in a real DFA is reading in characters one by one from a string. If, after reading the entire string, the DFA is in a finish (accepting) state, then the string is deemed to be accepted. Usually the characters are restricted to a set, for example lowercase alphabetic characters, or in this case just H or T.

DFAs are a natural counterpart to regular expressions which occur frequently in computer science, particularly when searching for strings in a large body of text (for example looking for a 6-letter word beginning with 'b' in a dictionary) or trying to ensure that a field on a web form is of the correct format (i.e. is it a valid email address?).

DFAs are also an important element of computer science theory and can be used as machines to recognize sets of strings of a type called *regular languages.* Regular languages are used a lot to have concise descriptions of sets of strings (called *regular expressions)*: for example, the strings that are some of the possible solutions to this task (with multiple repetitions of the string TTTHT and then THH), could be represented as a regular expression as (TTTHT)*THH because from the starting point, any combination with zero or more occurrences of TTTHT followed by one occurrence of THH will bring Flipper to the finish stone.

Encrypted Message

Ali sends an encrypted message to Zahra using a key consisting of numbers.
For example, Ali's message "My name is beaver" encrypted with the key = [3,2,1,4] would be "nia yeer Mmbe asv". (Spaces are not counted).

1	2	3	4
M	y	n	a
m	e	i	s
b	e	a	v
e	r		

Question

What encrypted message will Zahra receive if Ali sends the message "I wrote a letter" using the key = [3,1,2,4]?

1	2	3	4
I	w	r	o
t	e	a	l
e	t	t	e
r			

- rat wet lter ole
- lter rat wet ole
- rat ole wet lter
- rat lter wet ole
- lter wet rat ole
- rat lter ole wet

Explanation

The correct answer is "rat lter wet ole".

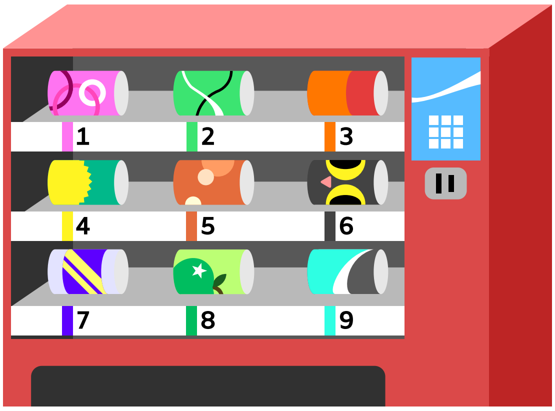
The used key is [3,1,2,4]. So, the first part of encrypted message comes from the 3rd column and is "rat". The second part comes from the 1st column and is "lter". The third part comes from the 2nd part and is "wet". The last part comes from 4th column and is "ole".

Background information

This task is based on columnar transposition cipher. In the first step, the plaintext of the message is written out in rows of a fixed length (spaces are filled with nulls, left blank, or placed by a character out in rows). In the second step, parts of the message are read column by column. The order in which columns should be read is defined by the key. So, this method of encryption scrambles the positions of characters without changing the characters themselves. This type of encryption was used during World War I.

Favourite Drinks

Damian, Nabil and Jasmine always buy their favourite drinks from a vending machine in their work place. Each of them have three favourites from the numbered nine options.



On Monday, only Damian and Nabil worked. Together they bought drinks 1, 2, 3, 5, 8 and 9.
On Tuesday, only Nabil and Jasmine worked. Together they bought drinks 1, 4, 5, 6, 7 and 9.

Task

What drink are their favourites? Put marks in the appropriate cells below by clicking on them.

Click again to remove the mark. Click "Save" when you are done.

	1	2	3	4	5	6	7	8	9
Damian									
Nabil									
Jasmine									

Save Erase

Explanation

	1	2	3	4	5	6	7	8	9
Damian		✓	✓					✓	
Nabil	✓				✓				✓
Jasmine				✓		✓	✓		

Monday: Damian and Nabil, drinks 1, 2, 3, 5, 8 and 9.
Tuesday: Nabil and Jasmine, drinks 1, 4, 5, 6, 7 and 9.
Since Nabil worked on both days, his favourites must be 1, 5 and 9, as these were repeated on both days.
Accordingly, on Monday, Damian bought drinks 2, 3, 8. On Tuesday, Jasmine bought drinks 4, 6, 7.

Background information

The idea of the task is to be able to analyze complex logical statements and draw correct conclusions based on them. The basic logic operations on the number lists are very important in Informatics. That gives a nice meaning to the basic logical operations (identical to the set operations).

It is sometimes hard to know where to start when you are faced with a complicated problem. In solving this task it is important to find an optimal order of checking logical statements. Breaking the problem into many smaller ones (decomposition) can help you to decide where to start.

Logic is one of the computational thinking skills you use when solving this task. You are given information about nine items bought from a vending machine on two separate days by three people who have three favorite items each. From that information, you have to use *deductive reasoning* to figure out what three items are each person's favorites. It is very helpful to use a *truth table* to solve this type of problem. In this case the truth table has a row for each person and a column for each vending machine item. Your task is to decide for each cell is that particular item that person's favorite. Or in logic we would say: Is the statement "this item is this person’s favorite" TRUE or FALSE.

Maze game

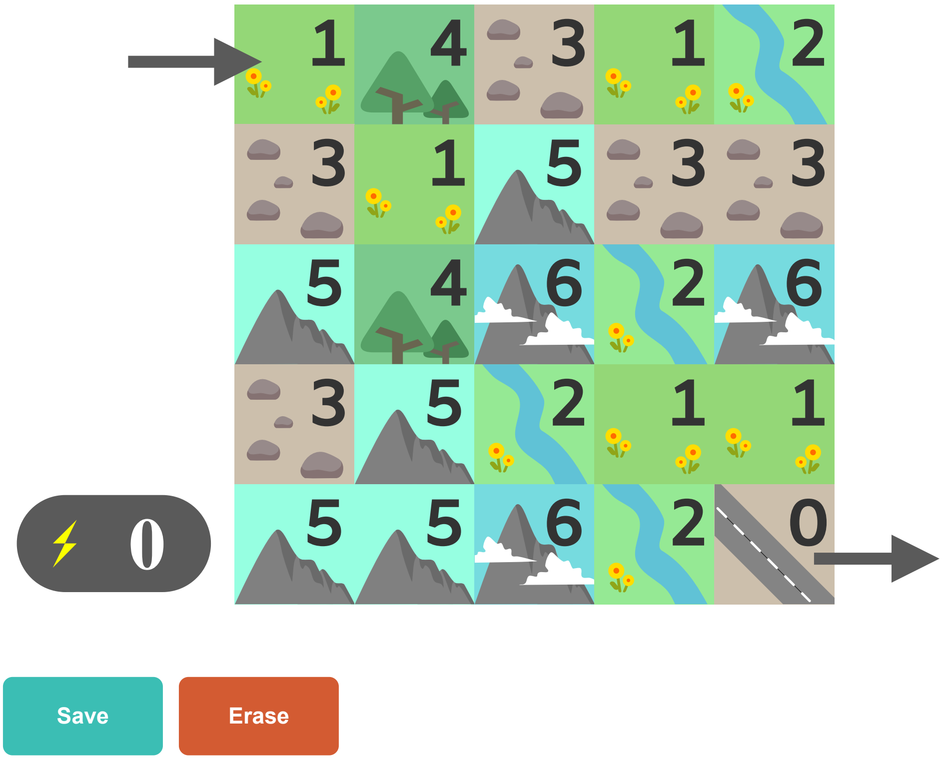
A crocodile is playing a board game. The game is played on a board divided into squares. Numbers in each square show how much energy the crocodile uses when it goes over the square.

The crocodile can only move down and right.

Task

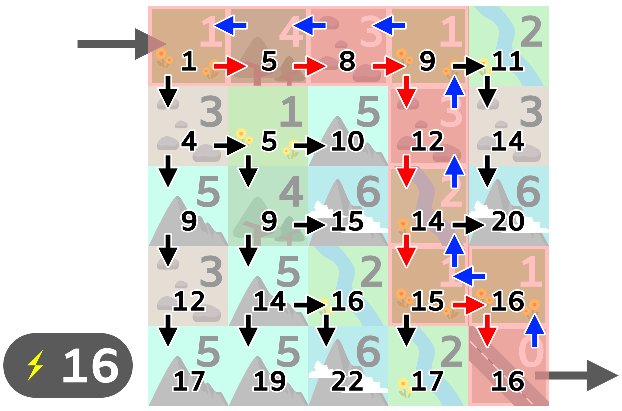
Find the path from the starting arrow to the ending arrow through which the crocodile uses the least amount of energy.

Click on each path section to select it. Press "Save" when you are done.



Explanation

The path from start to end that uses the least amount of energy is highlighted below.



On this path, the amount of energy that the crocodile uses is 1+4+3+1+3+2+1+1+0 = 16.
We should convince ourselves that 16 is the least possible energy used to go from start to end. To do this, we will call any path between two squares *optimal* if it is a path between them that uses the least possible amount of energy.
In the figure, the black numbers in the bottom of each square show how much energy is used in an optimal path to the square. If these numbers are correct, then we know the correct answer is 16. Why are these numbers correct?
Well, consider any square. Any optimal path to this square must begin with an optimal path to the square just above or just to the left. Therefore, once we know the optimal path to both of these squares, the one between them that requires the least energy determines an optimal path to the square under consideration. This is indicated by the arrows in the figure above. These arrows tell us which direction an optimal path to each square comes from.
We can also calculate how much energy is used along all these optimal paths. For example, look at the fifth square along the path of our answer to this problem (in the second row and fourth column). The amount of energy used in an optimal path to the square above is 9 and the amount of energy used in an optimal path to the square to the left is 10. Since 9 < 10, this means the optimal path to the third square on our path must come from above and the amount of energy it uses is 9+3=12.
We can use this idea to work right and downwards from the starting square to determine an optimal path from start to end and how much energy the crocodile uses to take this path.

Background information

One way to solve this task is the try all possible paths from START to STOP. This is an *algorithm* which uses a technique called *brute force*. It works but it usually takes a long time!
Computer scientists have several different ways to solve this type of task which normally take less time especially in a version of this game that included a much larger terrain with many more squares. The technique used in the explanation of the answer is called *dynamic programming*.
When solving this problem, you might have considered using a different technique. Perhaps you always tried to move to the next square (below or to the right) that would use the least amount of energy in the next step. People call this a *greedy algorithm*. Sometimes greedy algorithms do work but here, this particular idea is incorrect and will not find an optimal path.
Dijkstra's Algorithm is famous among computer scientists. It combines the ideas of dynamic programming and greedy algorithms and can be used to find optimal paths to all the squares.
Computational thinking is the application of methods and techniques derived from computer science in other areas of life. Dynamic programming is applied to optimization problems. Such problems occur in all areas of life. The situation from the task can be related to the movement of the object in certain directions with the smallest possible loss for a certain parameter. For example, it could be a passenger plane taking off or landing with as little fuel consumption as possible. The task also refers to the algorithms hidden in the background of board games.

Sprinklers

Amanda is planning to install new sprinklers to water the crops on her farm. The farm land is divided into squares, a map of which is shown below:

The store sells sprinklers that can water 4×4, 2×2, or 1×1 squares of land. There are four types of crops on Amanda's farm: apples, corn, grapes, and mushrooms. Each type of crop needs to be watered differently to promote ideal growth. Therefore, if a sprinkler waters a particular patch of land, then all the crops on that patch should be of the same type. Also, each patch is watered by only one sprinkler.

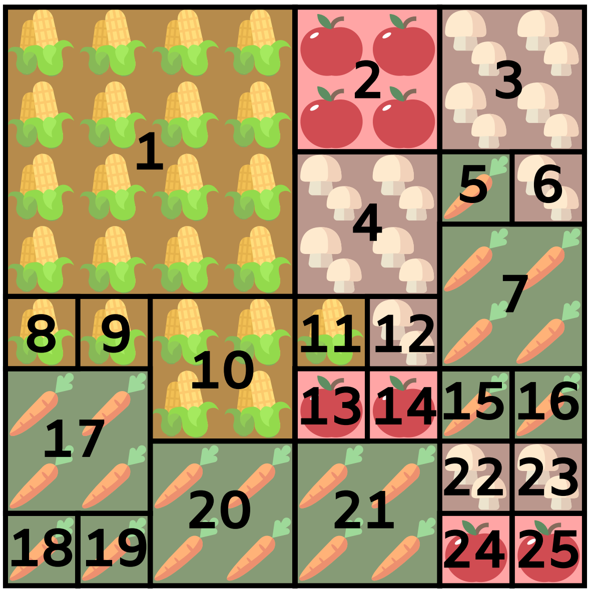
Question

What is the minimum number of sprinklers that Amanda needs in order to water all the crops on her farm?

19 | 21 | 23 | 25 | 27

Explanation

The correct answer is (D) 25. In order to minimize the number of sprinklers needed to water the farm, the patch of land watered by each sprinkler should be as large as possible — while ensuring that (1) all the crops inside a patch are of the same type, (2) the patches do not overlap, and (3) spinklers can overlap the patches they water. The figure below shows one way this can be done:



Maximizing the size of the patch for each crop watered by a sprinkler, given the constraint of the size of a patch (4x4, 2x2, 1x1) that different sprinklers can water, does result in minimizing the number of sprinklers needed to water the farm. Starting with the patch at the top lefthand corner of the farm. For the same crop select the largest patch which can be sprinkled (4x4, 2x2, 1x1). When the appropriate sprinkler has been place, then move to the next free right patch and select the appropriate sprinkler for crops of a similiar type. If the border of the farm is reached, then start the process again with the next row of crops. Repeat this process until all patches have a sprinkler associated with them.

If sprinkers are chosen that water 1x1 patches only, then a minimum of 64 sprinkers would be needed. As the farm can be represented by 64 patches.

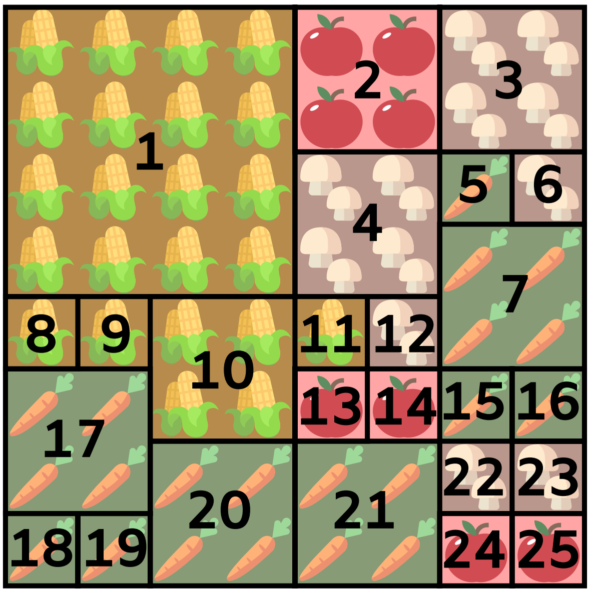
If sprinkers are chosen that water both 2x2 patches and 1x1 patches only. Starting with placing sprinklers that water patches of size 2x2, then placing sprinklers that water 1x1 patches. Then a minimum of 28 sprinklers would be needed. As the farm can be represented by 12 2x2 patches and 16 1x1 patches.

If sprinkers are chosen that water 4x4, 2x2 and 1x1 patches. Starting with placing sprinklers that water patches of size 4x4, then placing sprinklers that water patches of size 2x2, and finally placing sprinklers that watch 1x1 patches. As the fram can bt represented by 1 4x4 patch, 8 2x2 patches and 16 1x1 patches. Then a minimum of 25 sprinkers would be needed.

Background information

This challenge is an example of an *optimization problem*, a task with the goal of maximizing or minimizing a target value given a set of restrictions. Several optimization problems can be solved by following the strategy of always selecting the *local optimum* (the best choice at each step) and hoping that doing so will lead to the *global optimum* (the best result in the end). This problem-solving paradigm characterizes *greedy algorithms*. A greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage. In many problems, a greedy strategy does not produce an optimal solution, but a greedy heuristic can yield locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time.

Aside from the greedy algorithm, the solution touches on another informatics concept known as a *quadtree*. A quadtree is a way to represent an image by dividing it into four equal square regions. A region can further be divided into four regions until each region contains exactly one color – similar to how the solution divides the farm into regions until each region encloses exactly one fruit. For example, look at the picture below:



Quadtrees thus allow images to be stored as a collection of regions rather than individual pixels, resulting in smaller storage requirements and faster operations. This idea can also be extended to other two-dimensional data, such as maps, making quadtrees useful in image processing, game development, and navigation, among other applications.

An important skill in computational thinking is *tinkering*. The big picture of this challenge, which is to minimize the number of sprinklers needed to water a farm, can be a bit overwhelming. Hence, it may be helpful to explore some "what-if" questions first. In this challenge, students may start by tinkering with either small or large patches. Asking "What if I start with 2×2 patches?" motivates combining some patches into 4×4 patches and so on. Meanwhile, asking "What if I start with 8×8 patches?" opens the realization that some patches have to be divided into 4×4 patches and so on. Although these insights may not be necessarily structured at first, they are key to formulating the greedy algorithm for this challenge.

Gift selection

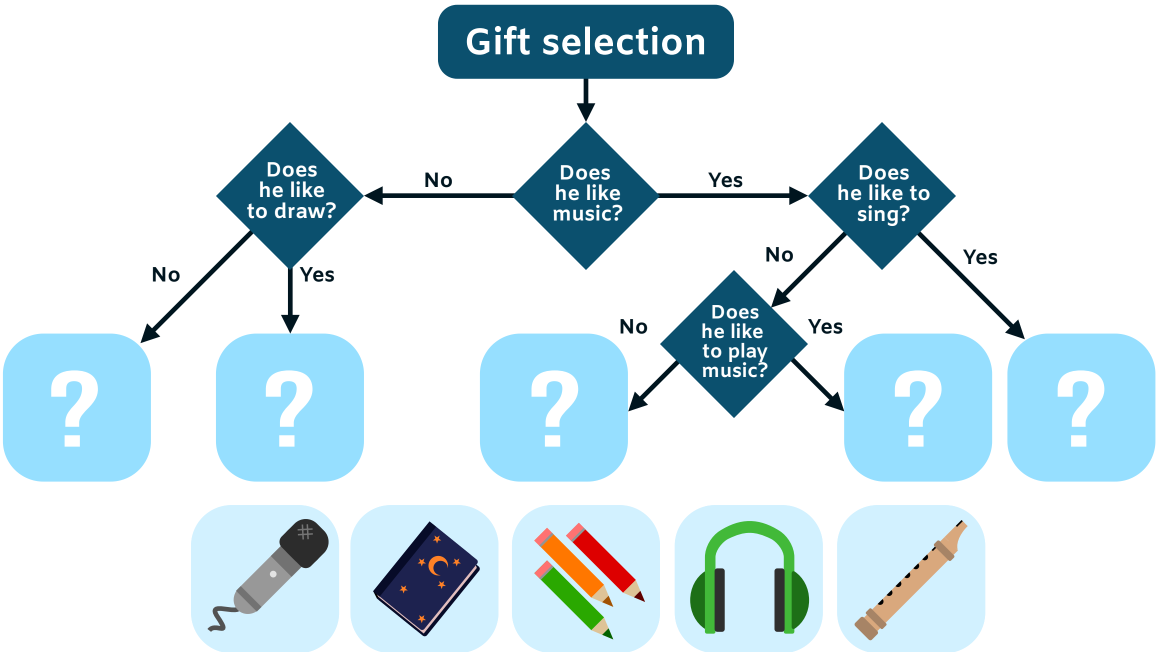
A new student has transferred to Madina's class. They want to surprise him with a gift.

In order to pick the right gift, they ask him a few questions. The flow chart below shows how they ask questions:

Task

Help Madina's class decide which gifts would best fit the new student's possible answers.

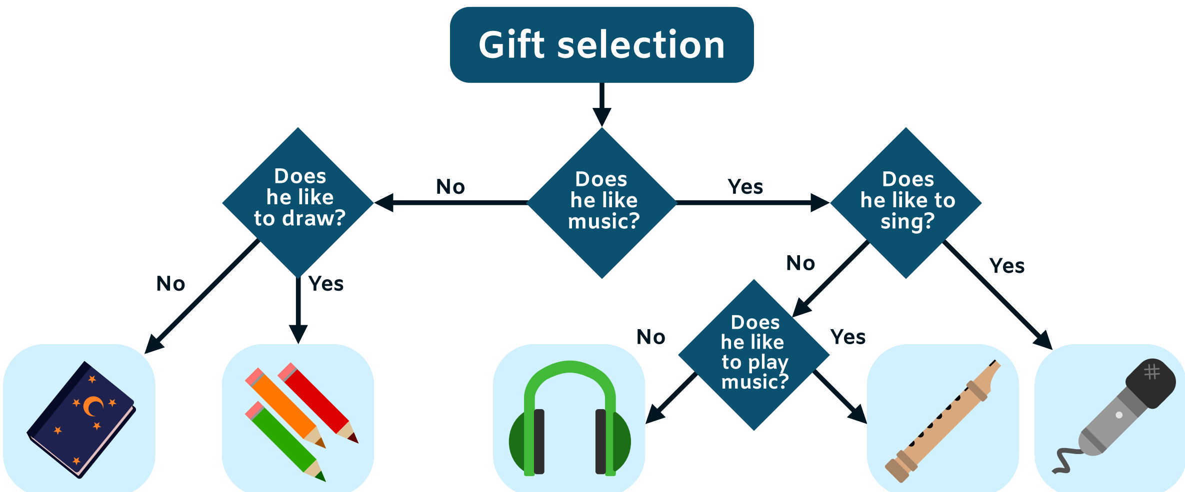
Drag the gifts to the empty boxes and click "Save" when you are done.



Save Erase

Explanation

The following diagram illustrates the correct solution:



If he loves music, they can give him a microphone, recorder, and headphones. If he likes to sing, place the microphone in position E. If he doesn't like to sing but likes to play music, place the recorder in position D and the headphones in position C.

If he doesn't like music but likes to draw, place the colour pencils in position B. Otherwise, place the diary in position A.

Background information

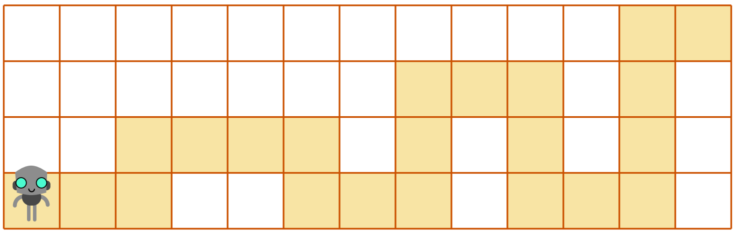
A decision tree is like a map to help you make decisions (in our example we used the decision tree to place gifts based on new student's answers).

It starts with a key question, and each answer can lead to either more questions, or a final answer.

Computer programs can often be visualised as decision trees. A program can be given an input, and then we use code to specify questions to ask of the input, which ultimately leads with a result that can be output.

Robot on the path

We want to program a robot to walk along a path in a park as well as possible. The path is shown in yellow in the picture below. The robot starts at the **left end** of the path, indicated by the little robot image.



The robot knows these **commands**:

- **U** move one square up
- **R** move one square right
- **D** move one square down

The robot's memory holds 5 commands. The robot repeats this sequence until it reaches the right border of the park.

Question

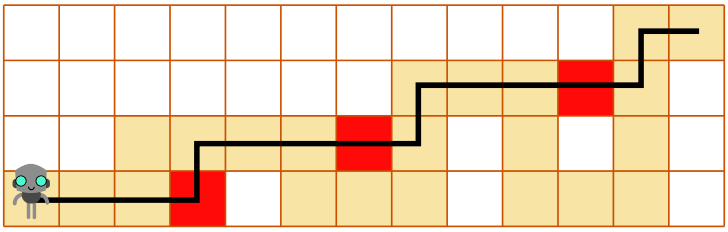
Which sequence of commands will take the robot to the right border of the park while moving over **the fewest number** of white squares?

RRURR | RRRUR | URRRD | RURRR

Explanation

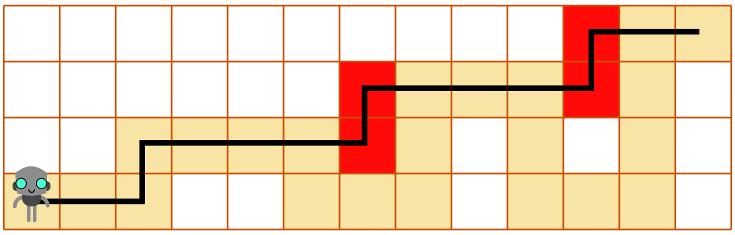
The correct answer is B) **RRRUR**

Here is what happens repeating commands in B) **RRRUR**:

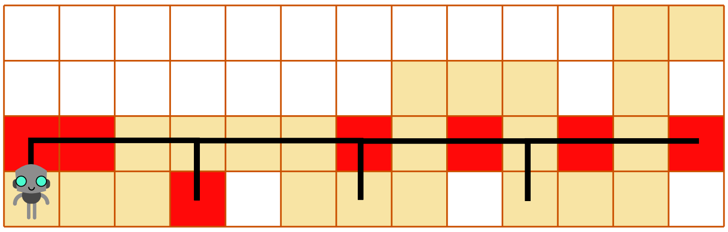


We see that the robot is off the path at **3 squares** (coloured red).

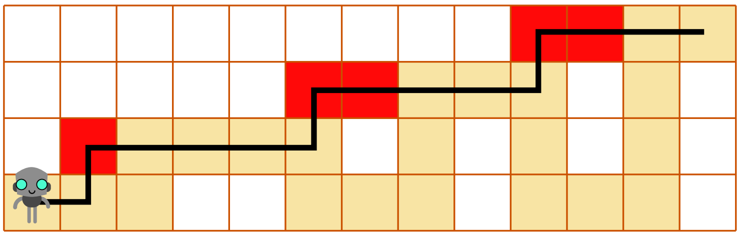
If it were to execute commands A) **RRURR**, it would be off the path at **4 squares**:



If it were to execute commands C) **URRRD**, it would go off the path at **7 squares**:



If it were executing commands D) **RURRR**, it would go off the path at **5 squares**:



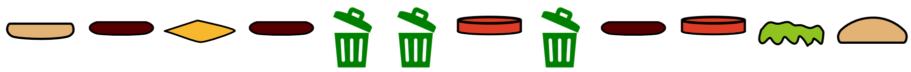
Background information

The path of the robot is described by a sequence of 5 commands that the robot repeats. According to the commands, the robot changes its position. Executing a (repeating) sequence of commands is one of the fundamental ideas in programming.
Commands control the robot in this problem by moving it in one of four directions, so-called "absolute control", without the robot turning in that direction.

2024 Bebras Challenge
Round 2 | Upper Primary Year 5 & 6

Hamburger Shop

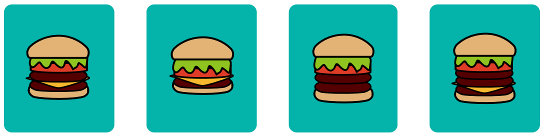
Masahiro wanted to make a tasty hamburger for his mum and tried an experiment. Yuki observed him and recorded the order in which Masahiro added and took away items. Here is what she noted down:



The picture  means that Masahiro took away one item from the top of the hamburger.

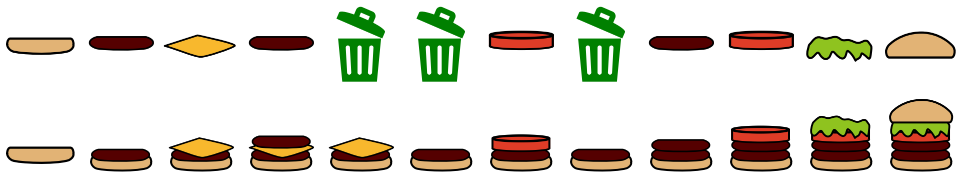
Question

Which of the following hamburgers did Masahiro make?



Explanation

The table shows all steps of the hamburger creation. The top row shows the sequence of actions. The bottom row shows how the hamburger looks after each action in the top row.



Therefore the answer is  .

Masahiro added 9 items and has taken away 3 items, so the hamburger has to consist of 6 items. The answer a) is wrong because this hambunger has only 5 items.

The answer b) is wrong because this hamburger contains cheese but cheese has been taken away.

The answer d) is wrong because this hamburger contains two tomatoes but one of the tomatoes has been taken away.

Background information

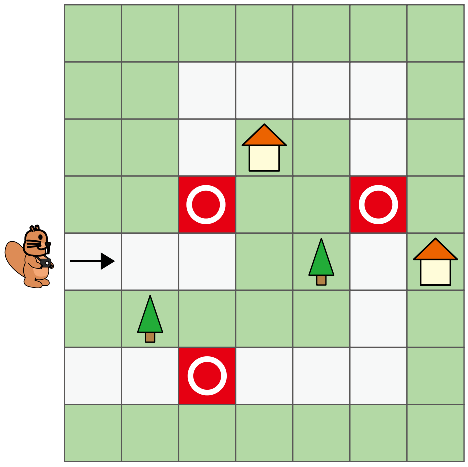
In this task, a hamburger is made using two operations; adding an item and removing the topmost item. The data structure managed by these two operations is called a "stack" in computer science. The main characteristic of the stack data structure is LIFO (Last In First Out), where the last object put in is the first object taken out. The stack data structure is used in many types of computer programs, such as providing the redo/undo functionality in editing software.

This task is algorithmic, the solver has to find the final state after running the given algorithm.
Abstraction is also present in this task. The icon to take away an ingredient represents a more general instruction, rather than the ingredient icons which represent their own specific item to add.

Walking Logs

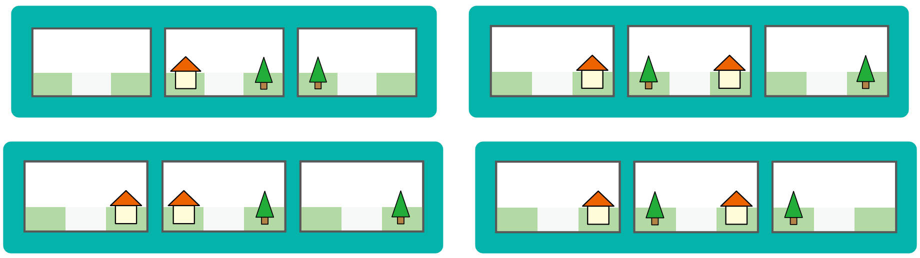
Wendy goes for a walk along the same path every day. One day Wendy took a camera on their walk and stopped 3 times to take photographs of the path ahead. Each photograph shows the square directly in front of Wendy, and the objects to either side of that square.

This map shows the path that Wendy walks along. Wendy starts on the square marked with an arrow, moving in the direction indicated until they reach the end of the path. The places that they took photographs are marked on the map with a white circle.



Question

Select which 3 photographs were taken by Wendy.

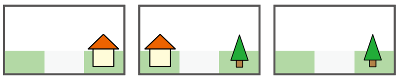


Explanation

Wendy would be facing north in the first photo-taking spot, so the photograph should show the house on the right side of the photograph.

Wendy would be facing south in the second photo-taking spot, so the photograph should show the house on the left side of the photograph and tree on the right.

Wendy would be facing west in the third photo-taking spot, so the photograph should show the tree on the right side of the photograph.

Therefore the answer is .

Background information

Object recognition is an important topic in computer science. An artificial intelligence system can be trained to recognize objects like houses, trees and road surfaces. Probably your smartphone camera is able to recognize faces and indicates them with a frame. An autonomous car must recognize people, cars, traffic signs and other objects in its environment, which are important for its decisions where to go. To help blind and visually impaired persons, modern web media include Automatic Alt Text (AAT). This technology uses object recognition to automatically create verbal descriptions (alt texts) of images.

This question can be approached using abstraction, as there is a lot of additional information in the question that is not needed to find the answer.

It can also be approached using modeling and simulation skills, with students mapping out the path the beaver walks, and considering what can be seen from each location.

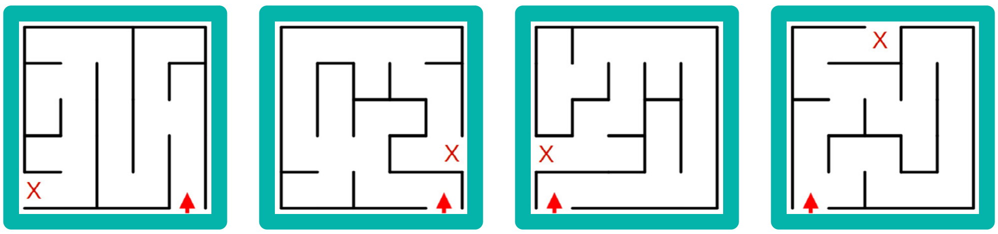
Mazes

Ben likes to explore garden mazes. However, every time Ben wonders whether he has gone through every path in the maze. One day, he learns "The right hand" rule. This rule means that while walking through a maze, you always have your right hand touching the right hand wall. The right hand rule guarantees that when you enter a maze, you can always find an exit. But, the right hand rule does not guarantee that you get to visit every path in the maze.

Below you see four mazes, with an arrow indicating where Ben enters the maze and an X showing the exit.

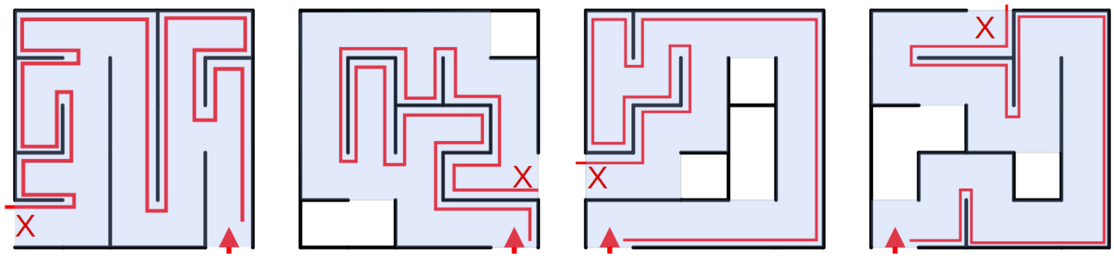
Question

Select the maze where Ben will walk on every path of the maze when following the right hand rule.



Explanation

The correct answer is A. Following the right hand rule, Ben visits every part of the maze. Below you can see the walls that Ben touches through the "Right hand" rule (red line) and the paths Ben covers using this rule (blue shade). In Option A, Ben walks through all paths.



A	B	C	D

Background information

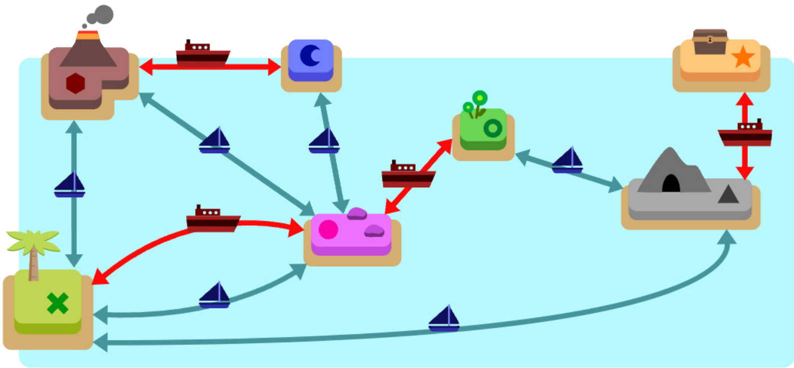
The 'right-hand' rule is an algorithm that you can use to go through a maze. The more generic version of this algorithm is called 'wall follower' because you basically keep following one wall. The wall follower is a depth-first search algorithm for mazes.

Algorithms are specific sets of instructions to achieve a specific result. In this case the result is: find an exit after you have entered.

There are many fun algorithms to explore when looking at mazes. For example, if you want to know how far each position in the maze is from the entrance you would use a 'flood fill' algorithm.

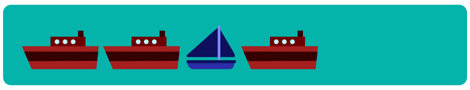
Treasure Island

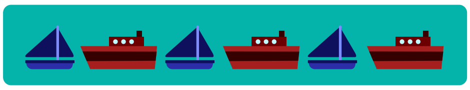
Bill is on vacation in the Island Kingdom. There are two types of ships sailing between the islands: and . You can see their routes in the picture. When people visit the islands, they pick a ship to go to the next island.

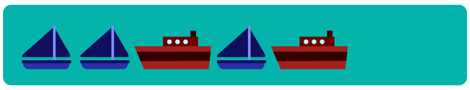


Question

Which of the following trips starting from Island **cannot help** Bill to reach Island ?





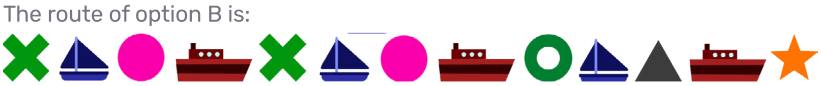


Explanation

Given that Bill departs from Island .

The route of option A is:

The route of option B is:



The route of option C is:

It is impossible to finish on island using the path of option D. The route of option D finishes on island or island

For example:

Only the route of option D cannot reach Treasure Island.

Background information

To solve problems using computers, we first need to represent the problems in a way that computers can understand. In this particular case, we are dealing with transportation networks, which can be represented as a graph. A graph is a type of data structure that shows how different things are connected to each other.

By using a graph, computers can help us solve various problems. For example, they can find the most cost-effective route between two places, determine the shortest path, solve the problem in communication network, and much more. In this task, students are asked to finding all possible routes from Island ✖ to Island ★ using the graph representation.

Encrypted Message

Ali sends an encrypted message to Zahra using a key consisting of numbers.

For example, Ali's message "My name is beaver" encrypted with the key = [3,2,1,4] would be "nia yeer Mmbe asv". (Spaces are not counted).

1	2	3	4
M	y	n	a
m	e	i	s
b	e	a	v
e	r		

Question

What encrypted message will Zahra receive if Ali sends the message "I wrote a letter" using the key = [3,1,2,4]?

1	2	3	4
I	w	r	o
t	e	a	l
e	t	t	e
r			

- rat wet lter ole
- lter rat wet ole
- rat ole wet lter
- rat lter wet ole
- lter wet rat ole
- rat lter ole wet

Explanation

The correct answer is "rat Iter wet ole".

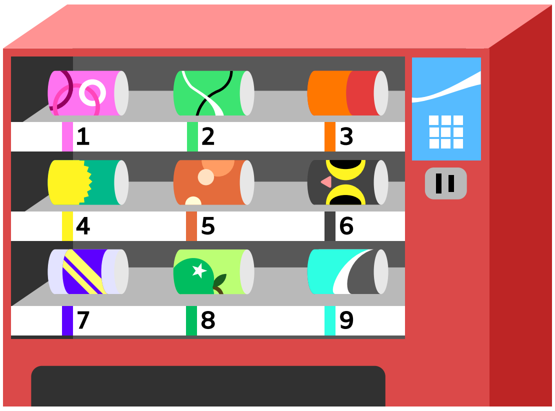
The used key is [3,1,2,4]. So, the first part of encrypted message comes from the 3rd column and is "rat". The second part comes from the 1st column and is "Iter". The third part comes from the 2nd part and is "wet". The last part comes from 4th column and is "ole".

Background information

This task is based on columnar transposition cipher. In the first step, the plaintext of the message is written out in rows of a fixed length (spaces are filled with nulls, left blank, or placed by a character out in rows). In the second step, parts of the message are read column by column. The order in which columns should be read is defined by the key. So, this method of encryption scrambles the positions of characters without changing the characters themselves. This type of encryption was used during World War I.

Favourite Drinks

Damian, Nabil and Jasmine always buy their favourite drinks from a vending machine in their work place. Each of them have three favourites from the numbered nine options.



On Monday, only Damian and Nabil worked. Together they bought drinks 1, 2, 3, 5, 8 and 9.
On Tuesday, only Nabil and Jasmine worked. Together they bought drinks 1, 4, 5, 6, 7 and 9.

Task

What drink are their favourites? Put marks in the appropriate cells below by clicking on them.
Click again to remove the mark. Click "Save" when you are done.

	1	2	3	4	5	6	7	8	9
Damian									
Nabil									
Jasmine									

Save

Erase

Explanation

	1	2	3	4	5	6	7	8	9
Damian		✓	✓					✓	
Nabil	✓				✓				✓
Jasmine				✓		✓	✓		

Monday: Damian and Nabil, drinks 1, 2, 3, 5, 8 and 9.
Tuesday: Nabil and Jasmine, drinks 1, 4, 5, 6, 7 and 9.
Since Nabil worked on both days, his favourites must be 1, 5 and 9, as these were repeated on both days.
Accordingly, on Monday, Damian bought drinks 2, 3, 8. On Tuesday, Jasmine bought drinks 4, 6, 7.

Background information

The idea of the task is to be able to analyze complex logical statements and draw correct conclusions based on them. The basic logic operations on the number lists are very important in Informatics. That gives a nice meaning to the basic logical operations (identical to the set operations).

It is sometimes hard to know where to start when you are faced with a complicated problem. In solving this task it is important to find an optimal order of checking logical statements. Breaking the problem into many smaller ones (decomposition) can help you to decide where to start.

Logic is one of the computational thinking skills you use when solving this task. You are given information about nine items bought from a vending machine on two separate days by three people who have three favorite items each. From that information, you have to use *deductive reasoning* to figure out what three items are each person's favorites. It is very helpful to use a *truth table* to solve this type of problem. In this case the truth table has a row for each person and a column for each vending machine item. Your task is to decide for each cell is that particular item that person's favorite. Or in logic we would say: Is the statement "this item is this person’s favorite" TRUE or FALSE.

Maze game

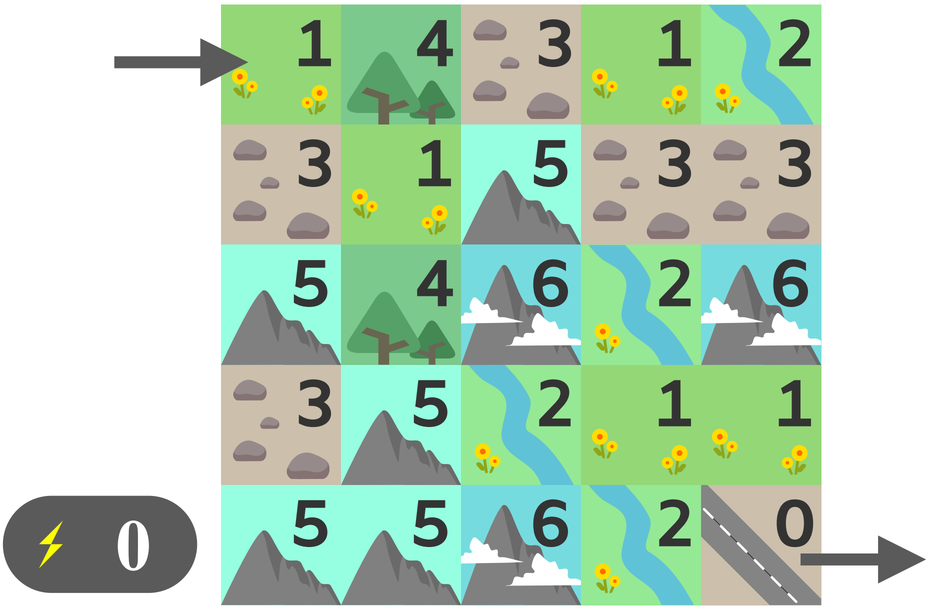
A crocodile is playing a board game. The game is played on a board divided into squares. Numbers in each square show how much energy the crocodile uses when it goes over the square.

The crocodile can only move down and right.

Task

Find the path from the starting arrow to the ending arrow through which the crocodile uses the least amount of energy.

Click on each path section to select it. Press "Save" when you are done.

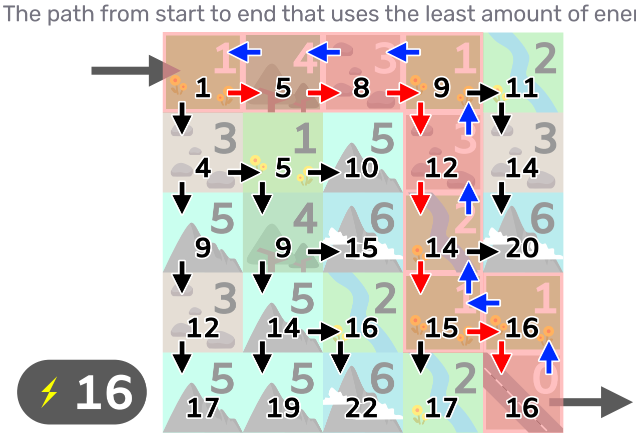


Save

Erase

Explanation

The path from start to end that uses the least amount of energy is highlighted below.



On this path, the amount of energy that the crocodile uses is 1+4+3+1+3+2+1+1+0 = 16.
We should convince ourselves that 16 is the least possible energy used to go from start to end. To do this, we will call any path between two squares *optimal* if it is a path between them that uses the least possible amount of energy.
In the figure, the black numbers in the bottom of each square show how much energy is used in an optimal path to the square. If these numbers are correct, then we know the correct answer is 16. Why are these numbers correct?
Well, consider any square. Any optimal path to this square must begin with an optimal path to the square just above or just to the left. Therefore, once we know the optimal path to both of these squares, the one between them that requires the least energy determines an optimal path to the square under consideration. This is indicated by the arrows in the figure above. These arrows tell us which direction an optimal path to each square comes from.
We can also calculate how much energy is used along all these optimal paths. For example, look at the fifth square along the path of our answer to this problem (in the second row and fourth column). The amount of energy used in an optimal path to the square above is 9 and the amount of energy used in an optimal path to the square to the left is 10. Since 9 < 10, this means the optimal path to the third square on our path must come from above and the amount of energy it uses is 9+3=12.
We can use this idea to work right and downwards from the starting square to determine an optimal path from start to end and how much energy the crocodile uses to take this path.

Background information

One way to solve this task is the try all possible paths from START to STOP. This is an *algorithm* which uses a technique called *brute force*. It works but it usually takes a long time!
Computer scientists have several different ways to solve this type of task which normally take less time especially in a version of this game that included a much larger terrain with many more squares. The technique used in the explanation of the answer is called *dynamic programming*.
When solving this problem, you might have considered using a different technique. Perhaps you always tried to move to the next square (below or to the right) that would use the least amount of energy in the next step. People call this a *greedy algorithm*. Sometimes greedy algorithms do work but here, this particular idea is incorrect and will not find an optimal path.
Dijkstra's Algorithm is famous among computer scientists. It combines the ideas of dynamic programming and greedy algorithms and can be used to find optimal paths to all the squares.
Computational thinking is the application of methods and techniques derived from computer science in other areas of life. Dynamic programming is applied to optimization problems. Such problems occur in all areas of life. The situation from the task can be related to the movement of the object in certain directions with the smallest possible loss for a certain parameter. For example, it could be a passenger plane taking off or landing with as little fuel consumption as possible. The task also refers to the algorithms hidden in the background of board games.

Sprinklers

Amanda is planning to install new sprinklers to water the crops on her farm. The farm land is divided into squares, a map of which is shown below:

The store sells sprinklers that can water 4×4, 2×2, or 1×1 squares of land. There are four types of crops on Amanda's farm: apples, corn, grapes, and mushrooms. Each type of crop needs to be watered differently to promote ideal growth. Therefore, if a sprinkler waters a particular patch of land, then all the crops on that patch should be of the same type. Also, each patch is watered by only one sprinkler.

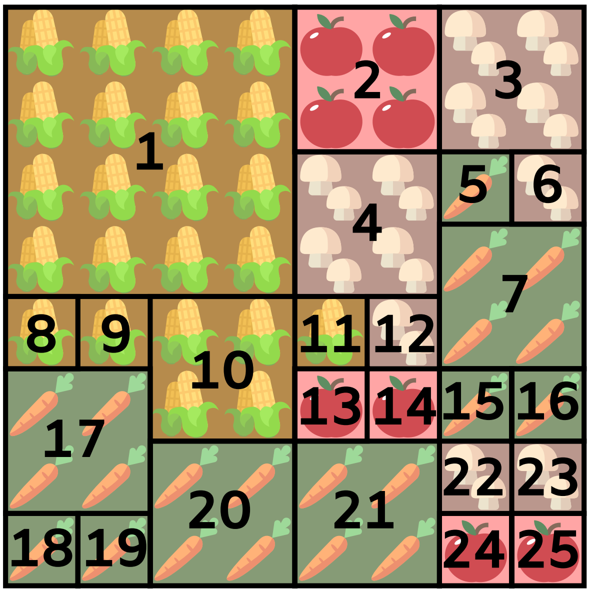
Question

What is the minimum number of sprinklers that Amanda needs in order to water all the crops on her farm?

19 | 21 | 23 | 25 | 27

Explanation

The correct answer is (D) 25. In order to minimize the number of sprinklers needed to water the farm, the patch of land watered by each sprinkler should be as large as possible — while ensuring that (1) all the crops inside a patch are of the same type, (2) the patches do not overlap, and (3) spinklers can overlap the patches they water. The figure below shows one way this can be done:



Maximizing the size of the patch for each crop watered by a sprinkler, given the constraint of the size of a patch (4x4, 2x2, 1x1) that different sprinklers can water, does result in minimizing the number of sprinklers needed to water the farm. Starting with the patch at the top lefthand corner of the farm. For the same crop select the largest patch which can be sprinkled (4x4, 2x2, 1x1). When the appropriate sprinkler has been place, then move to the next free right patch and select the appropriate sprinkler for crops of a similiar type. If the border of the farm is reached, then start the process again with the next row of crops. Repeat this process until all patches have a sprinkler associated with them.

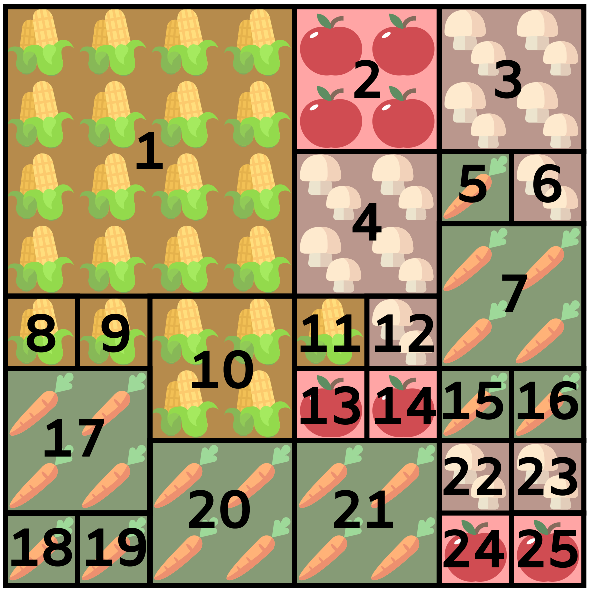
If sprinkers are chosen that water 1x1 patches only, then a minimum of 64 sprinkers would be needed. As the farm can be represented by 64 patches.

If sprinkers are chosen that water both 2x2 patches and 1x1 patches only. Starting with placing sprinklers that water patches of size 2x2, then placing sprinklers that water 1x1 patches. Then a minimum of 28 sprinklers would be needed. As the farm can be represented by 12 2x2 patches and 16 1x1 patches.

If sprinkers are chosen that water 4x4, 2x2 and 1x1 patches. Starting with placing sprinklers that water patches of size 4x4, then placing sprinklers that water patches of size 2x2, and finally placing sprinklers that watch 1x1 patches. As the fram can bt represented by 1 4x4 patch, 8 2x2 patches and 16 1x1 patches. Then a minimum of 25 sprinkers would be needed.

Background information

This challenge is an example of an *optimization problem*, a task with the goal of maximizing or minimizing a target value given a set of restrictions. Several optimization problems can be solved by following the strategy of always selecting the *local optimum* (the best choice at each step) and hoping that doing so will lead to the *global optimum* (the best result in the end). This problem-solving paradigm characterizes *greedy algorithms*. A greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage. In many problems, a greedy strategy does not produce an optimal solution, but a greedy heuristic can yield locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time.
Aside from the greedy algorithm, the solution touches on another informatics concept known as a *quadtree*. A quadtree is a way to represent an image by dividing it into four equal square regions. A region can further be divided into four regions until each region contains exactly one color — similar to how the solution divides the farm into regions until each region encloses exactly one fruit. For example, look at the picture below:



Quadtrees thus allow images to be stored as a collection of regions rather than individual pixels, resulting in smaller storage requirements and faster operations. This idea can also be extended to other two-dimensional data, such as maps, making quadtrees useful in image processing, game development, and navigation, among other applications.

An important skill in computational thinking is *tinkering*. The big picture of this challenge, which is to minimize the number of sprinklers needed to water a farm, can be a bit overwhelming. Hence, it may be helpful to explore some "what-if" questions first. In this challenge, students may start by tinkering with either small or large patches. Asking "What if I start with 2×2 patches?" motivates combining some patches into 4×4 patches and so on. Meanwhile, asking "What if I start with 8×8 patches?" opens the realization that some patches have to be divided into 4×4 patches and so on. Although these insights may not be necessarily structured at first, they are key to formulating the greedy algorithm for this challenge.

Gift selection

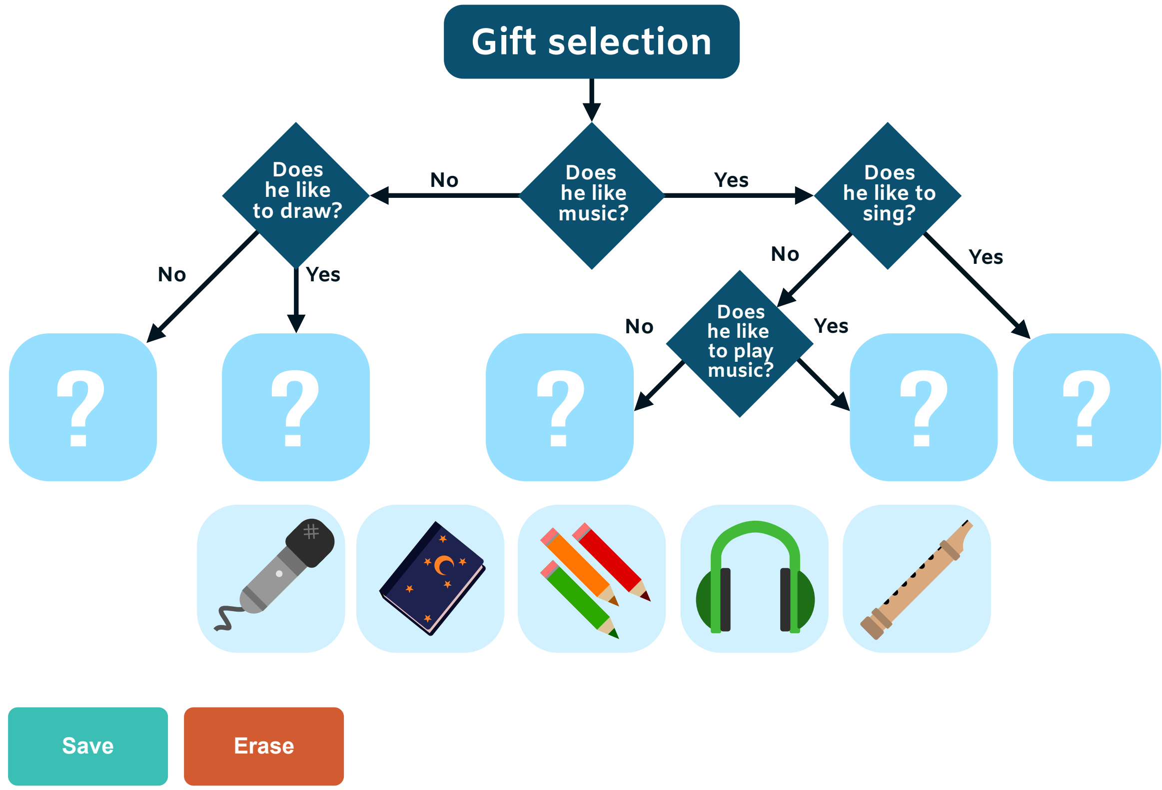
A new student has transferred to Madina's class. They want to surprise him with a gift.

In order to pick the right gift, they ask him a few questions. The flow chart below shows how they ask questions:

Task

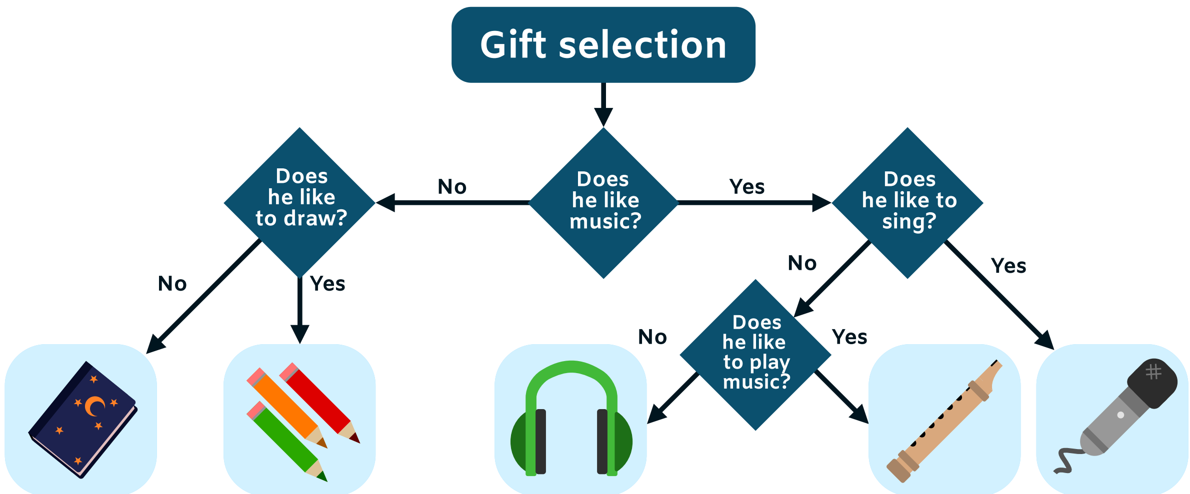
Help Madina's class decide which gifts would best fit the new student's possible answers.

Drag the gifts to the empty boxes and click "Save" when you are done.



Explanation

The following diagram illustrates the correct solution:



If he loves music, they can give him a microphone, recorder, and headphones. If he likes to sing, place the microphone in position E. If he doesn't like to sing but likes to play music, place the recorder in position D and the headphones in position C.

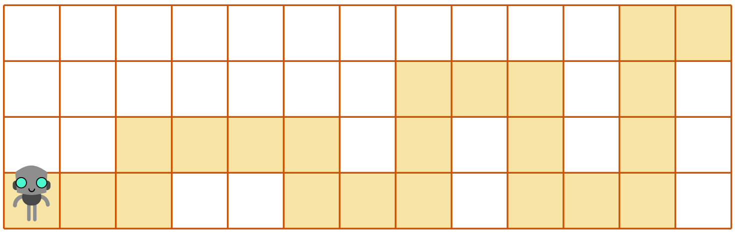
If he doesn't like music but likes to draw, place the colour pencils in position B. Otherwise, place the diary in position A.

Background information

A decision tree is like a map to help you make decisions (in our example we used the decision tree to place gifts based on new student's answers).
It starts with a key question, and each answer can lead to either more questions, or a final answer.
Computer programs can often be visualised as decision trees. A program can be given an input, and then we use code to specify questions to ask of the input, which ultimately leads with a result that can be output.

Robot on the path

We want to program a robot to walk along a path in a park as well as possible. The path is shown in yellow in the picture below. The robot starts at the **left end** of the path, indicated by the little robot image.



The robot knows these **commands**:

- **U** move one square up
- **R** move one square right
- **D** move one square down

The robot's memory holds 5 commands. The robot repeats this sequence until it reaches the right border of the park.

Question

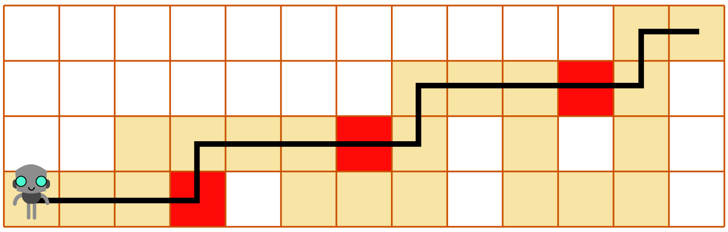
Which sequence of commands will take the robot to the right border of the park while moving over **the fewest number** of white squares?

RRURR | **RRRUR** | **URRRD** | **RURRR**

Explanation

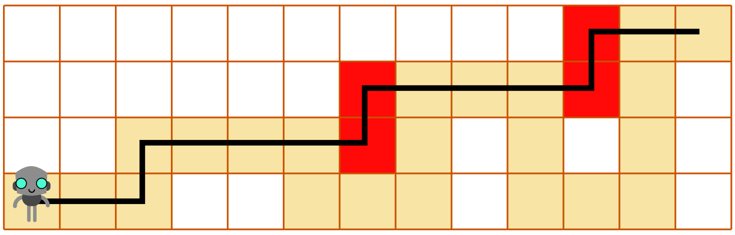
The correct answer is B) **RRRUR**

Here is what happens repeating commands in B) **RRRUR**:

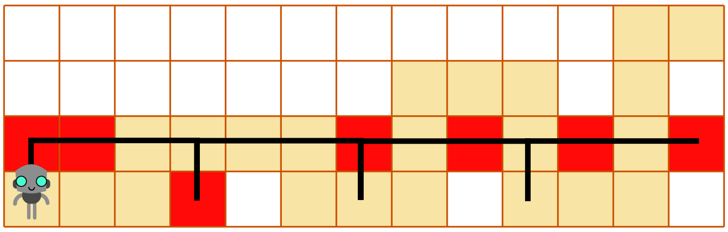


We see that the robot is off the path at **3 squares** (coloured red).

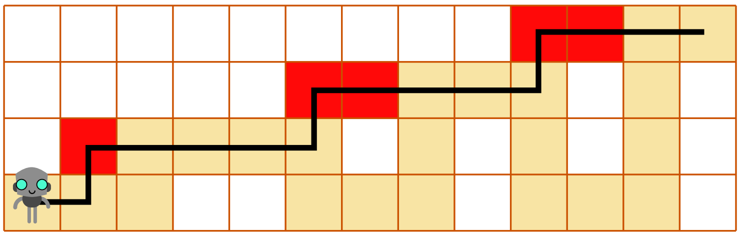
If it were to execute commands A) **RRURR**, it would be off the path at **4 squares**:



If it were to execute commands C) **URRRD**, it would go off the path at **7 squares**:



If it were executing commands D) **RURRR**, it would go off the path at **5 squares**:



Background information

The path of the robot is described by a sequence of 5 commands that the robot repeats. According to the commands, the robot changes its position. Executing a (repeating) sequence of commands is one of the fundamental ideas in programming.
Commands control the robot in this problem by moving it in one of four directions, so-called "absolute control", without the robot turning in that direction.

Sealed Letters

The Republic of Beaveria has a cabinet full of secret letters.

Among the 16 letters in this cabinet, numbered 1 to 16, 10 had been opened, while the other 6 were still inside their sealed envelopes.

One evening, an enemy spy snuck in and opened one of the sealed letters. However, they forgot to seal it again.

The next morning, the Republic of Beaveria goes into an investigation after noticing that there are now 11 opened letters, as shown in the diagram.

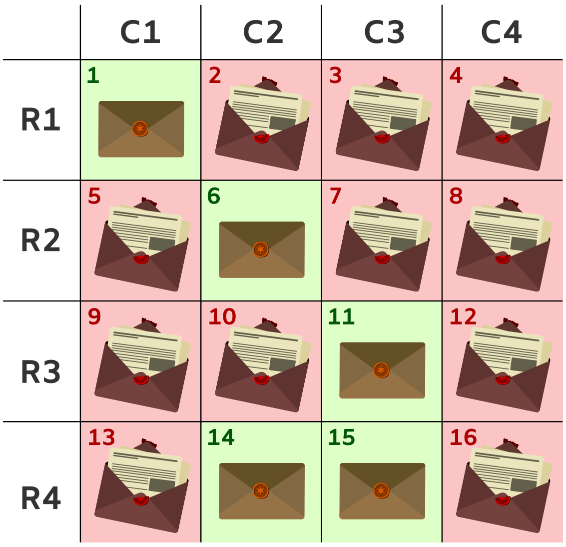
The Republic's guard does not recall all the details, but they are sure that before the enemy spy sneaked in:

- The total number of opened letters in the C2 and C4 columns was even.
- The total number of opened letters in the C3 and C4 columns was even.
- The total number of opened letters in the R2 and R4 rows was even.
- The total number of opened letters in the R3 and R4 rows was even

Question

Which letter was opened by the enemy spy?

Click on the letter and press "Save".



Save

Erase

Explanation

The correct answer is 13.

We follow this line of reasoning:

- There is an even number of opened letters in the C2 and C4 columns combined, matching the guard's recollection. As there is only one letter opened by the spy, this implies that the letter opened by the spy must be either in the C1 or C3 column. There is an even number of opened letters in the C3 and C4 columns combined, matching the guard's recollection. Given the previous statement, this implies that the letter opened by the spy must be in the C1 column.
- There is an odd number of opened letters in the R2 and R4 rows combined, which does not match the guard's recollection. This implies that the letter opened by the spy must be either in the R2 or R4 row.
- There is an odd number of opened letters in the R3 and R4 rows combined, which does not match the guard's recollection. Given the previous statement, this implies that the letter opened by the spy must be in the R4 row.

Therefore, the letter read by the spy is in the C1 column and R4 row, which points to 13.

Background information

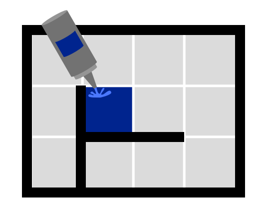
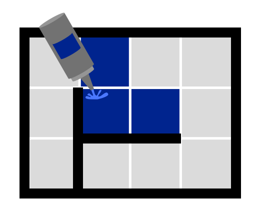
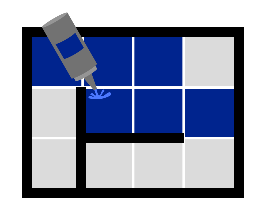
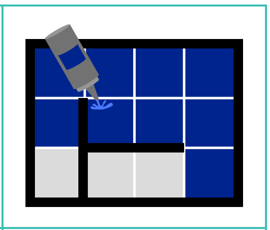
This challenge introduces the concept of *error-correcting* codes. Digital data is essentially a sequence of bits: 1s and 0s. When it passes through a network, it can be corrupted due to noise or due to the action of malicious entities. Data stored on DVDs may also become corrupted if these storage devices are scratched. It is thus important to have a scheme that is capable of not only detecting that corruption has occurred (*error detection*) but also correcting the corrupted bits (*error correction*).

The first modern error-correcting code was introduced by Richard Hamming in 1950. This challenge specifically borrows the idea of the *[15, 11] Hamming code*, with the closed and opened letters corresponding to 0s and 1s, respectively. This coding scheme takes an 11-bit data and inserts 4 *parity bits*. These parity bits occupy slots 2, 3, 5, and 9, while the bits of the original data occupy the remaining slots (except slot 1). The parity bits are set so that there is an even number of 1s in the 2nd and 4th columns, 3rd and 4th columns, 2nd and 4th rows, and 3rd and 4th rows. Finally, the bit at slot 1 is set so that there is an even number of 1s in total.

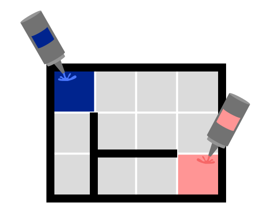
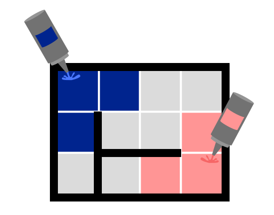
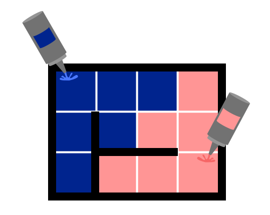
As seen in the presented solution, this scheme allows for the corrupted bit to be identified and subsequently corrected, provided that there is at most 1 corrupted bit. Although more sophisticated error-correcting schemes are necessary to correct 2 or more corrupted bits, the Hamming code's efficiency contributes to its continued usage in NAND flash memory chips embedded in mobile phones and solid-state drives.
An important skill in computational thinking is *deductive reasoning* — logically deriving conclusions from a set of known facts or *premises*. A common method in deductive reasoning is the process of *elimination*. For instance, in this challenge, knowing that the 2nd and 4th columns combined have an even number of opened letters eliminates the possibility that they have the letter read by the spy, thus narrowing down the choices to the 1st and 3rd columns. This ability to reason in a systematic, top-down manner is a core competency in designing algorithms, tracing code, and debugging programs without resorting to guesswork or committing logical fallacies.

Watercolour

When painters pour watercolour into a maze, the colour will spread to neighbouring empty squares every second. The colour cannot spread through the walls, as you can see below.

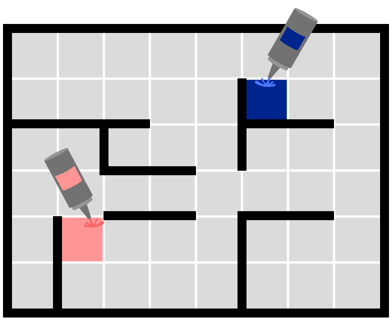
			
0 seconds	1 second	2 seconds	3 seconds

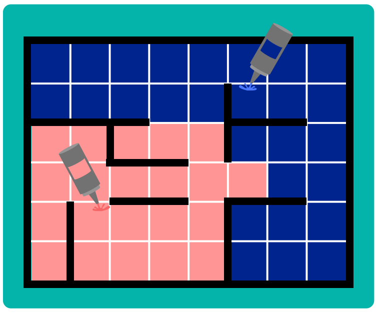
If the painters pour more than one watercolour into the maze, the first colour that reaches a square will fill it completely. When two colours reach a square at the same time, the square takes the darker colour (blue).

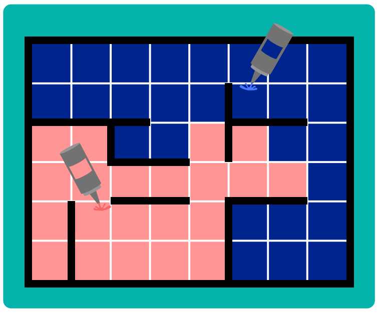
		
0 seconds	1 second	2 seconds

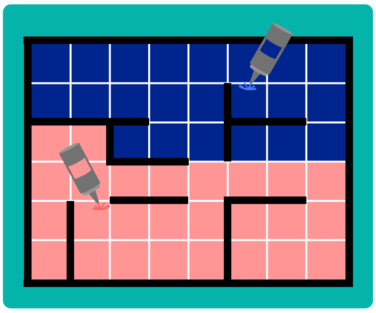
Question

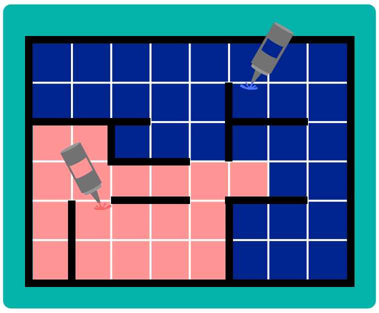
The painters poured two colours into the maze below. What does the maze look like when all the squares are filled with colour?





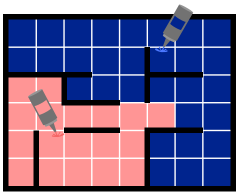




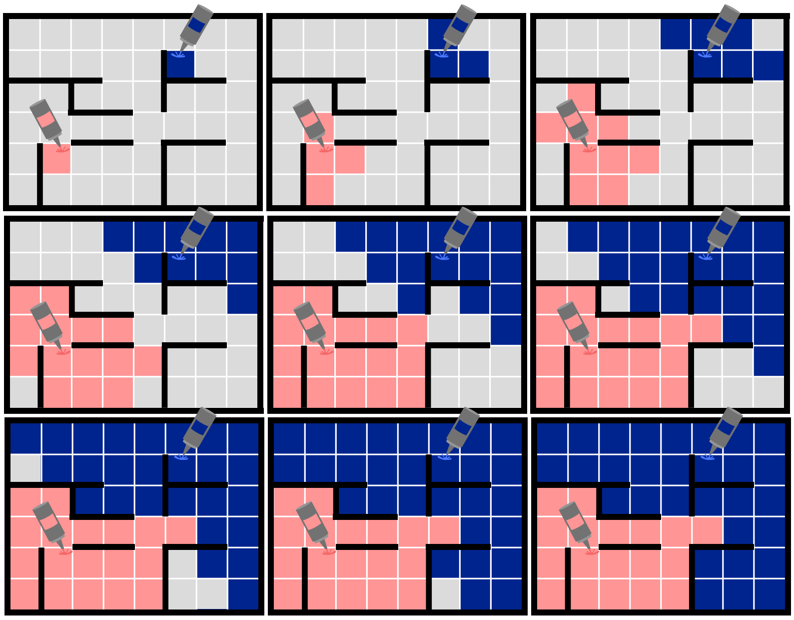


Explanation

The correct answer is



The image below shows the state of the maze second by second:



Background information

The setting of the task resembles a two dimensional array, which basically means a table with rows and columns. This way of representing the data is quite functional to simulate the state of the maze second after second and solve the task.

However, it is also possible to represent it as a graph in which each square is connected to its neighbors. This graph allows us to identify quickly which color will reach a square without simulating the whole scenario second by second. This approach using graphs is the same as performing a breadth-first search.

The breadth-first search (BFS) algorithm is one way of searching a tree or graph for a node that meets a set of criteria. It starts at any given node of a graph and visits all nodes at the current depth level before moving on to the nodes at the next depth level. Breadth-first search can be used to solve many problems in graph theory.

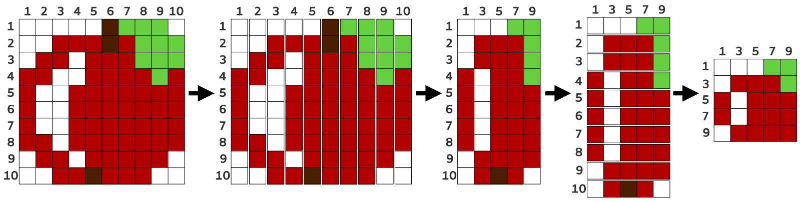
Algorithmic thinking is key in solving this tasks since students have to understand how the setting works in order to simulate its situation second by second.

Snail Compress

Snails have a special technique to shrink their captured images.

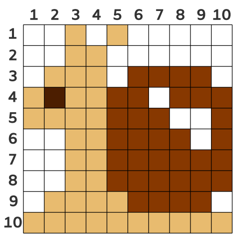
First, they will cut the original image into 10 equally sized strips vertically. Then, they will assemble the odd-numbered vertical strips to create a new image.

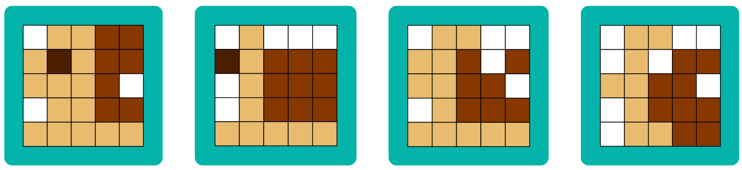
Next, they will cut the new image horizontally into 10 equally sized strips. Then, they will assemble the odd-numbered horizontal strips to create a complete shrunken image.



Question

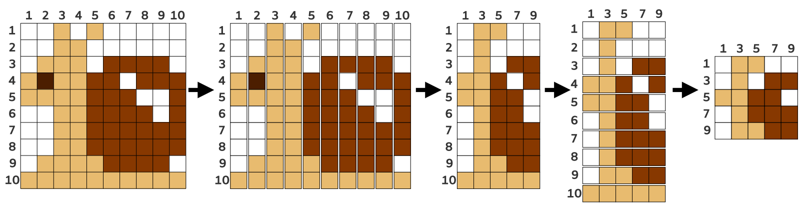
What image will the snails obtain after shrinking the given image using this technique?





Explanation

Following the same procedure as with the picture of the apple, the snail image is compressed in the following way:



First all the even-numbered columns will be erased, than all the even-numbered rows will be erased, which means that only the cells from odd-numbered columns, that are also in odd-numbered rows will remain.

We can see that the correct answer is D, as it is the only image that uses the 9th row for compression of the picture.

Background information

Compression of an image refers to the process of reducing the size of an image file by eliminating or minimising some of the unnecessary data in the image while still maintaining sufficient image quality for use. This process helps to reduce the file size of images, making them easier to store or transmit.
This is lossy compression algorithm, as it completely eliminates some of the rows and columns in the original image. JPG images are also stored using a lossy compression algorithm.
In contrast, a lossless compression algorithm only minimises the data. PNG images are stored using a lossless compression algorithm.

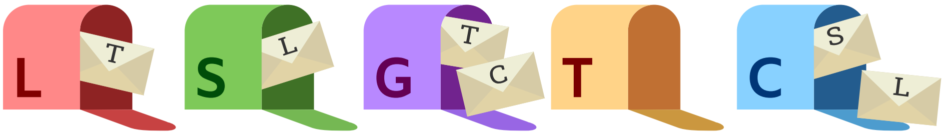
Delivering Mail

Percy delivers mail to five friends. The first letter of their name is printed on their mailbox. When the friends want to send mail to each other, they leave the mail in their own mailbox for Percy to deliver. Percy visits the friends' mailboxes one by one.

At each mailbox, Percy does two things:

- He collects any mail in the mailbox and puts it in his bag.
- He delivers all the mail for that friend from his bag by putting it in their mailbox.

Before Percy delivers any mail, he knows that the friends' mailboxes contain the following mail.



Percy starts with an empty bag, and must only visit each mailbox exactly once. Therefore, he needs to be clever about the order in which he visits them so that he can deliver all the mail.

Question

In order for Percy to deliver all the mail, he must visit the mailboxes in which of the following orders?

- Gina → Cato → Leon → Sue → Theo
- Gina → Sue → Cato → Theo → Leon
- Gina → Cato → Sue → Leon → Theo
- Cato → Gina → Sue → Leon → Theo

Explanation

The correct answer is Option C. Percy must visit the mailboxes in the following order: Gina → Cato → Sue → Leon → Theo.

Since Percy can only visit each mailbox exactly once and delivers all the mail, we can work backwards, using the picture to determine the order in which he must visit the mailboxes.

Since Theo's mailbox is the only mailbox that does not contain any mail, Percy must visit Theo's mailbox last.

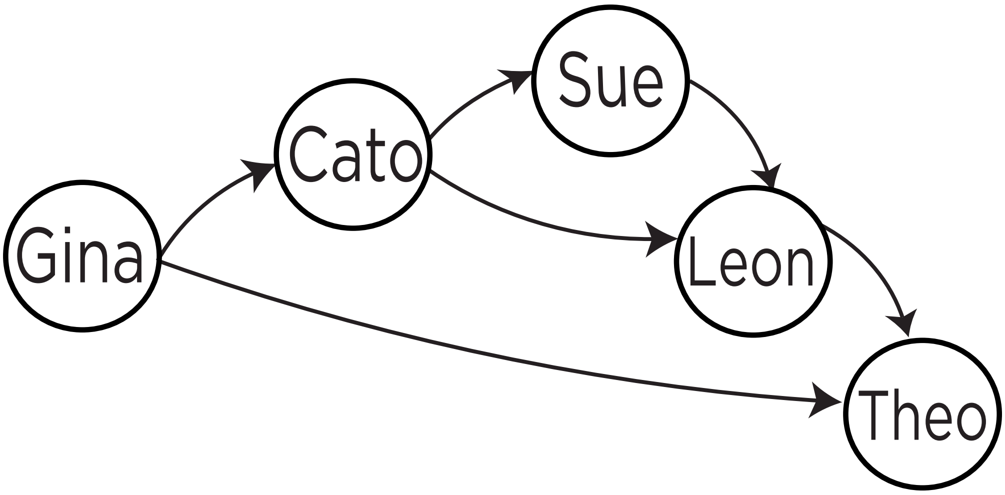
Of the remaining mailboxes, Leon's mailbox is the only one that contains mail only for Theo. Thus, Percy must visit Leon's mailbox fourth.

We are left with Sue's, Gina's, and Cato's mailboxes. Of these, Sue's mailbox is the only one that contains mail just for Leon or Theo. Thus, Percy must visit Sue's mailbox third.

We are left with Gina's and Cato's mailboxes. Since Gina's mailbox contains mail for Cato, but Cato's mailbox does not contain mail for Gina, Cato's mailbox must be visited second, and Gina's mailbox must be visited first. Therefore, Percy must visit the mailboxes in the order given in Option C.

Background information

The mail that is initially in the mailboxes before Percy delivers any mail gives us clues about the order in which Percy can visit the mailboxes. We can put these clues together to create the following diagram consisting of circles named nodes and arrows named directed edges. An arrow from node A to node B means person A wants to sent a mail to person B.

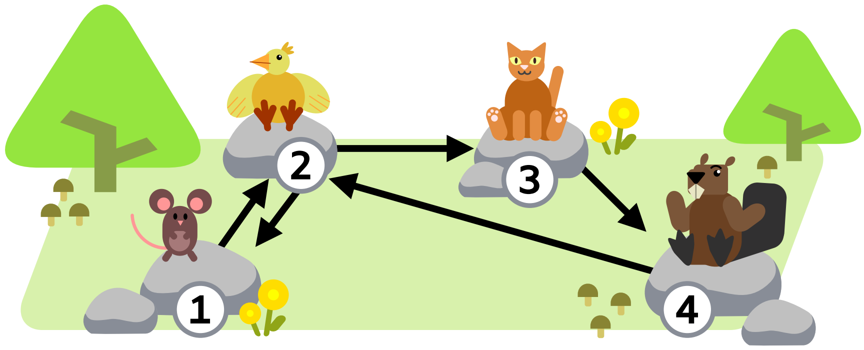


This diagram is called a *directed acyclic graph*, or DAG for short. A graph is "acyclic" because it doesn't contain any cycles or loops. A DAG is commonly used to describe relationships in computer science, such as for scheduling, parallel processing, and data compression. The problem asks to find the order in which Percy must visit the mailboxes. This is called a *topological ordering* of the corresponding DAG, where the names are arranged in a line with all arrows pointing forward. Can you see that there is only one topological ordering for this DAG? That is, can you see why Percy *must* visit the mailboxes in the order given by Option C?

DAG are used everyday by us to solve task scheduling problems. For example cooking usually requires multiple tasks. We make a mental list of the order of the tasks: some tasks cannot start before until others are completed, while others can start at any time!

Jumping Together

Beaver lives by rock number 4. In his neighbourhood there are three other rocks where his friends Bird, Cat and Mouse live. The animals have decided to have a party at Beaver's rock. In order to do this the animals can jump from rock to rock, following the arrows, according to the drawing below. Because they all want to be equally tired when the party starts, they all want to jump around and arrive on Beaver's rock having all made the exact same number of jumps (including Beaver).



Question

What is the minimum number of jumps that the animals must make so they all arrive on Beaver's rock with the same amount of jumps?

Fill in the number and click "Save" when you are done.

Answer:

Save

Explanation

By making a single jump, the cat can reach Beaver, but Beaver would be gone, and both the mouse and the bird have not yet arrived.

Please note the following observations:

- Beaver and *mouse* are both one jump away from rock 2. Their second jump could always be the same. So any number that works for Beaver will also work for mouse.
- The lowest numbers for *cat* are 1, 4 and 6.
- The lowest numbers for *bird* are 2, 4, 5 and 6

Looking at it from the point of view of Beaver:

- Beaver could circle around in three jumps, but *cat* cannot do three.
- Beaver could also do five jumps (4 -> 2 -> 1 -> 2 -> 3 -> 4), but *cat* also cannot do five.

It turns out that every animal can do it in six jumps:

- Beaver can do six jumps (4 -> 2 -> 3 -> 4 -> 2 -> 3 -> 4)
- Cat can do six as well: (3 -> 4 -> 2 -> 1 -> 2 -> 3 -> 4)
- Mouse follows Beaver with six jumps (1 -> 2 -> 3 -> 4 -> 2 -> 3 ->4)
- Bird can also make six jumps: (2 -> 1 -> 2 -> 1 -> 2 -> 3 -> 4)

Obviously the animals can also do it in more jumps, they could do their loop of 6 jumps and then add another 3, 5, 6, 8, 9, 10, 11, ... jumps. It's interesting to verify that 1, 2, 4 and 7 extra jumps are not possible. Can you find out why?

Background information

This task can be represented as a graph in computer science, where each rock corresponds to a **vertex**, represented by a numbered circle, and the arrows indicate **directed edge**s in the graph. The objective is to find a **directed walk** from a starting vertex to the destination vertex.

In this task, it is allowed to visit the same edge multiple times in order to reach the destination vertex. The table below displays the shortest path from each vertex from the start vertex to the destination vertex, along with the length of the shortest path:

Starting vertex	The shortest path to vertex 4	Length of the shortest path
1	1→2→3→4	3
2	2→3→4	2
3	3→4	1
4	-	0

Furthermore, the table below shows the cycles, which represent the shortest path starting from each vertex back to the same vertex:

Starting vertex	Cycle starting from the node	Length of the cycle
1	1→2→1	2
2	2→3→4→2	3
3	3→4→2→3	3
4	4→2→3→4	3

By combining the shortest path from each vertex to the destination and the cycles of the intermediate vertices, all possible directed walks from a vertex to the destination can be enumerated. For example, the table below illustrates the directed walks from each vertex to vertex 4 with a length of 6:

Starting vertex	The directed walk of length 6 to vertex 4 by combining the above paths/cycles
1	1→2→3→4, 4→2→3→4
2	2→1→2, 2→1→2, 2→3→4
3	3→4, 4→2(2→1→2)→3→4
4	4→2→3→4, 4→2→3→4

This approach is a kind of recursion, which is a concept or process depends on a simpler version of itself. To find the pattern of repeating, the length of the directed walks from starting vertex to the destination can be calculated without needing to illustrate all the solutions.

2024 Round 2 All

ID	Question	Year Level	Decomposition	Pattern Recognition
2023-TR-02	Friendly Aliens	3/4 Easy	○	○
2023-IE-05	Coded Ages	3/4 Easy	○	
2023-UY-02	Carnival Masks	3/4 Easy	○	○
2023-UY-04	Hidden treasure	3/4 Easy	○	
2023-JP-05	Hamburger Shop	3/4 Easy, 5/6 Easy	○	
2023-JP-03a	Walking Logs	3/4 Medium, 5/6 Easy	○	
2023-NL-04	Mazes	3/4 Medium, 5/6 Easy	○	
2023-CN-03	Treasure Island	3/4 Medium, 5/6 Easy	○	
2023-IE-04	Travel by Coin	3/4 Medium	○	
2023-AZ-03	Encrypted Message	3/4 Medium, 5/6 Easy	○	
2023-UA-02	Favourite Drinks	3/4 Hard, 5/6 Medium, 7/8 Easy	○	
2023-AZ-04a	Maze game	3/4 Hard, 5/6 Medium, 7/8 Easy	○	
2023-PH-03	Sprinklers	3/4 Hard, 5/6 Medium, 7/8 Easy	○	○
2023-HU-03	Gift selection	3/4 Hard, 5/6 Medium, 7/8 Easy, 9/10 Easy	○	
2023-SK-06	Robot on the path	3/4 Hard, 5/6 Medium, 7/8 Medium, 9/10 Easy	○	
2023-PH-04b	Sealed Letters	5/6 Hard, 7/8 Medium, 9/10 Easy	○	
2023-VN-04	Watercolour	5/6 Hard, 7/8 Hard, 9/10 Medium, 11/12 Easy	○	
2023-VN-02	Snail Compress	5/6 Hard, 7/8 Medium, 9/10 Medium, 11/12 Easy	○	
2023-CA-03	Delivering Mail	5/6 Hard	○	
2023-PR-01	Jumping Together	5/6 Hard, 7/8 Hard, 9/10 Medium, 11/12 Easy	○	○
2023-HR-01	Frog and Mosquito	7/8 Easy	○	
2023-PT-02	BebrasGPT	7/8 Medium, 9/10 Easy, 11/12 Easy	○	
2023-BR-04	Stamp Machines	7/8 Medium, 9/10 Easy	○	○
2023-LT-03	Correcting error	7/8 Hard, 9/10 Medium, 11/12 Easy	○	
2023-SK-05	Bebras Ball	7/8 Hard, 9/10 Medium, 11/12 Medium	○	
2023-NL-01	Triangle	7/8 Hard, 9/10 Hard, 11/12 Medium	○	○
2023-HU-05	Open It	9/10 Hard, 11/12 Medium	○	○
2023-IT-01a	Channel plan	9/10 Hard, 11/12 Medium	○	
2023-AU-02	Palindrome Passwords	9/10 Hard, 11/12 Medium	○	○
2023-TW-05	Line Up	9/10 Hard, 11/12 Hard	○	
2023-IE-01	Mapping the roads	11/12 Hard	○	
2023-IT-02	Logs to the warehouse	11/12 Hard	○	○
2023-PT-03	Walking in the Forest	11/12 Hard	○	
2019-NL-12	Painting doors	11/12 Hard	○	○

Abstraction	Modelling & Simulation	Algorithms	Evaluation	Abstraction	Data Collection	Data Representation	Data Interpretation	Specification	Algorithms	Implementation	Digital Systems	Interactions	Impacts
●			●	●	●		●		●				
●			●	●		●			●				
●		●	●	●				●	●				
●	●	●		●					●				
●	●	●	●	●		●		●	●		●		
●			●	●		●	●				●		
●	●	●	●	●				●	●	●			
●	●	●	●	●		●		●	●	●		●	
●	●	●	●	●		●		●	●	●		●	
●		●	●	●				●	●	●			
●		●	●	●	●	●		●	●	●			
●	●	●	●	●					●	●		●	
●		●	●	●		●	●	●	●	●			
●		●	●	●					●	●	●		
●	●	●	●	●				●	●	●	●		
●			●	●	●			●					●
●	●	●	●	●					●	●		●	
●		●	●	●		●			●	●	●		
●	●	●	●	●				●	●	●	●		
●	●		●	●					●	●	●	●	
●	●	●	●	●					●	●	●		
●		●	●	●					●	●	●		●
●		●	●	●			●		●	●			
		●	●			●			●	●	●		●
●	●	●	●	●					●	●	●		
	●	●			●		●		●		●		
●			●	●		●				●			
●	●		●	●				●	●		●	●	
●		●	●	●		●			●	●	●		
●		●	●	●					●	●			
●	●		●	●	●			●	●	●	●	●	●
●			●	●					●	●			
●	●	●	●	●	●				●	●	●		
●		●	●	●					●	●	●		