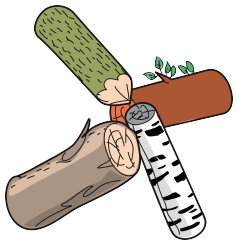
2023 Beaver Computing Challenge (Grades &)

Questions

Answer

(D)

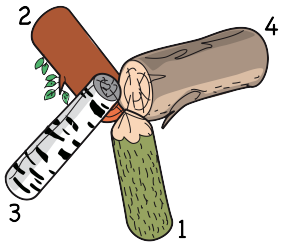


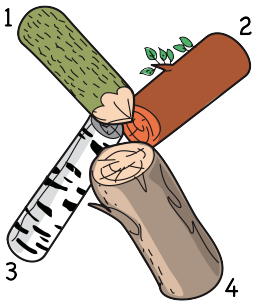
Explanation of Answer

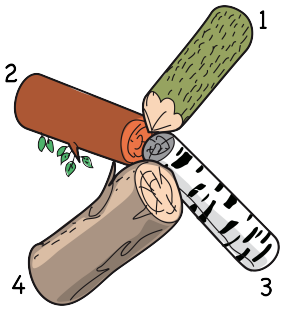
To determine how the logs appear in the photo, it is necessary to shift our view from the side of the logs to the top of the logs. To help make connections between these two views, we can look for relationships between the objects and then determine whether or not these relationships still hold when switching views.

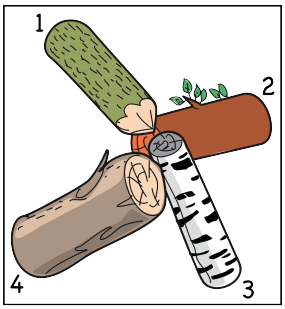
If we first focus on the side view and look at the bases of the logs, we can number the logs as we move clockwise:

We can then assign the same numbers to the same logs in the options for the top view. This will help us identify the one where the relationships still hold.









In only Option D are the logs correctly numbered as we move clockwise.

Connections to Computer Science

This task is concerned about determining the correct arrangement of objects, based on information from a different perspective of those objects. That is, the information from two images needs to be combined for consistency, using patterns in those images.

Detecting recurring patterns in given data is an important skill in computer science, generally referred to as *pattern recognition*. For example, in *image recognition*, pattern recognition techniques assist in identifying objects in images, classifying them into different categories, and even recognizing faces. In *natural language processing*, pattern recognition enables computers to understand the structure of sentences, identify topics, and even generate human-like text.

We can use pattern recognition with less abstract, real-life examples such as images. This task requires *spatial orientation*, which is important in *robotics* and *computer graphics*. Spatial orientation involves understanding and interpreting visual information, such as images, diagrams, or other graphical representations, and the ability to manipulate these representations in order to solve problems or create new solutions. In computer graphics, spatial orientation is used to create realistic animations and renderings, as it helps artists and programmers understand and manipulate complex 3D models.

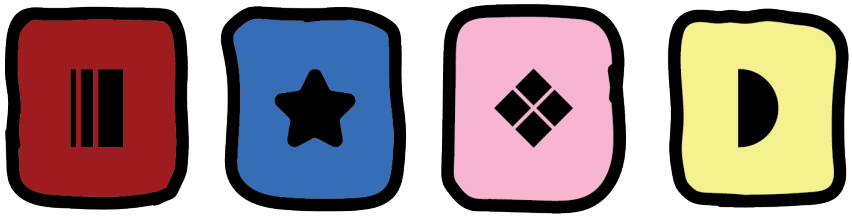
Country of Original Author

Lithuania

Cards

Story

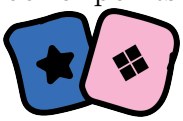
A card game has four types of cards:



The symbol on each card indicates the number of points the card is worth, as shown.

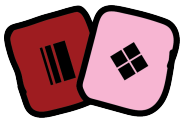
Symbol				
Number of Points	8	4	2	1

A player's score is the total number of points of the cards they have in their hand.

For example, Zita has the cards  and her score is $4 + 2 = 6$.

Question

If Silat's score is 9, what cards could he have in his hand?

(A) 

(B) 

(C) 

(D) 

Answer

(B) 

Explanation of Answer

If Silat has the cards  then his score is $8 + 2 = 10$.

If Silat has the cards  then his score is $8 + 2 + 1 = 11$.

If Silat has the cards  then his score is $8 + 4 + 1 = 13$.

However, if Silat has the cards  then his score is $8 + 1 = 9$ as required.

Connections to Computer Science

In this task, there is a connection between the symbol on the card and the underlying value of the card. All information that a computer works with has an underlying numeric value that is interpreted and can be presented in various ways.

For example, the number 81 can be interpreted as a number that is used in a calculation, but a computer could also interpret it as an *ASCII* code for the letter Q – depending on the context. The *American Standard Code for Information Interchange (ASCII)* is an standard way of translating each Latin-script keyboard character into a distinct number between 0 and 127. This standard allows characters to be transferred to machines and interpreted in the correct way.

This idea of different representations of the same value is an instance of *abstraction* in computer science. We use abstraction to hide unnecessary details, such as what the internal numeric value actually is, in order to more easily focus on the broader concepts, such as how humans view the information. In this task, the colour and pattern are the abstraction of the actual value of the card. Determining the correct level of abstraction allows computer scientists to focus on the fundamental, high-level view of the problem to find a general, scalable solution.

Figuring out what combination of cards results in a 9, we notice that the values of the cards are all powers of 2. The underlying numeric values computers use are represented as *binary sequences* – numbers made up of just 0s and 1s. That is, instead of using our common base-10 representation of numbers, where there is a "units" column, "tens" column, and "hundreds" column in a number, in base-2 representation, there is a "units" column, "twos" column, "fours" column, and "eights" column. These column values are all powers of 2. All base-10 whole numbers can be represented by a sum of powers of 2.

For example, the base-10 number 8 would be represented as 1000 in base 2: one eight, with zero fours, zero twos, and zero ones. As another example, 1111 in base-2 would represent the number $8 + 4 + 2 + 1 = 15$. The reason binary/base-2 numbers are used is that computers can more easily maintain *bits*, which are either "on" (representing 1) or "off" (representing 0).

Country of Original Author

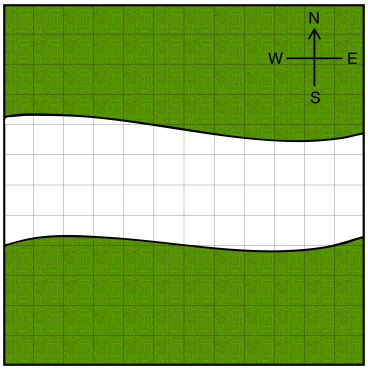
Ireland

Karla's House

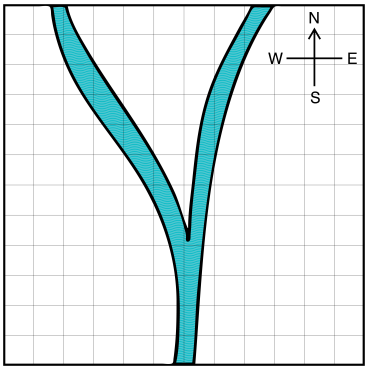
Story

Karla has three maps that all show exactly the same region. One map shows the forests, one shows the rivers, and one shows the houses. Karla's house is in the forest, touches the bank of the river, and is House A, B, C, or D.

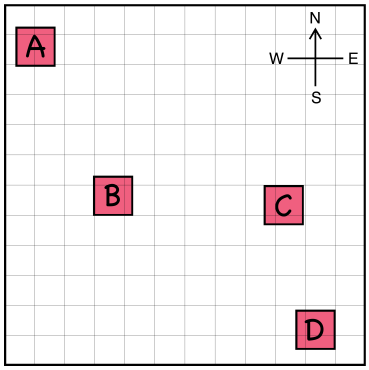
Forests



Rivers



Houses



Question

Which house is Karla's house?

(A) House A

(B) House B

(C) House C

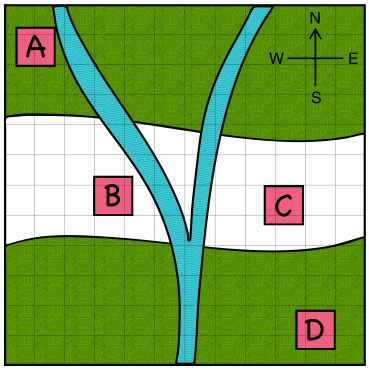
(D) House D

Answer

(A) House A

Explanation of Answer

To identify Karla's house, the information from all three maps must be considered together. This can be done by overlaying the 3 maps:



Since Karla's house is in the forest and touches the bank of the river, it must be House A. Houses B and C are not in the forest, and House D does not touch the bank of the river.

Connections to Computer Science

If the information about the forests, the rivers, and the houses were shown on a single map, then it would be easy to identify the house you are looking for.

In a *geographic information system (GIS)*, a wide variety of spatial information about the surface of the Earth is captured, stored and displayed on a computer. In particular, each different type of information, such as vegetation, buildings, water systems, or roadways is stored independently, but can be integrated in various combinations in order to more easily visualize and analyze patterns and relationships. Some applications of a GIS are emergency management when a natural disaster occurs, car navigation systems, and mapping water systems to preserve wetland habitats.

The principle of multiple layers with different image information is also used in many *computer graphics programs*, primarily to place visual objects in front of or behind other objects. An important question is always which layer or which objects are displayed in the foreground. In this task with three layers of images, Karla's map of the houses should be the top layer in order for the houses to not be hidden by the forest areas.

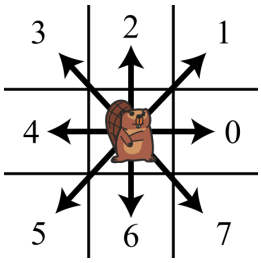
Country of Original Author

Germany

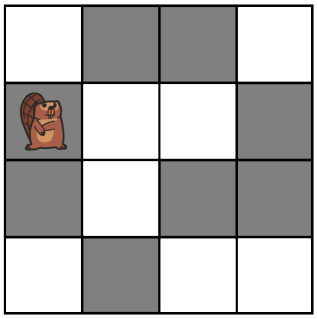
Video Game

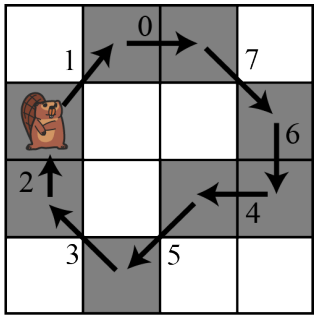
Story

Sasha uses the numbers from 0 to 7 to move her character different directions in a video game, as shown.



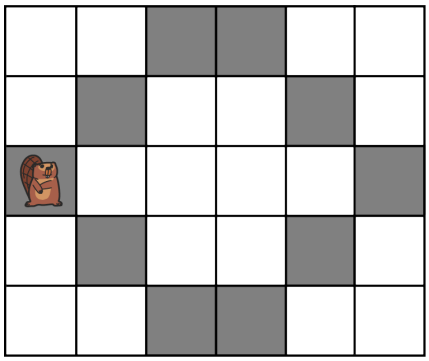
For example, to move her character clockwise along the grey path below and back to its original position, she types the sequence 1, 0, 7, 6, 4, 5, 3, 2.





Question

Which of the following sequences will move Sasha's character clockwise along the grey path and back to its original position?



(A) 1, 1, 7, 7, 5, 5, 3, 3

(B) 1, 1, 4, 7, 7, 5, 5, 0, 3, 3

(C) 1, 1, 0, 7, 7, 5, 5, 4, 3, 3

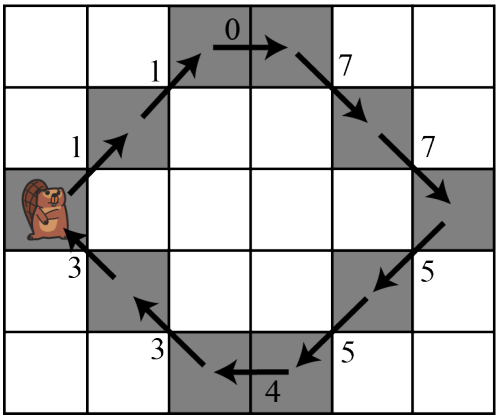
(D) 7, 7, 0, 1, 1, 5, 5, 4, 3, 3

Answer

(C) 1, 1, 0, 7, 7, 5, 5, 4, 3, 3

Explanation of Answer

The image below shows each movement along with its corresponding number.



Thus, the sequence in Option C is correct.

Connections to Computer Science

A *chain code* is method to store an image based on its outline. The chain code of an image is determined by specifying a starting pixel (squares in this task) and the sequence of steps to other squares of the image in one of eight directions: left, right, up, down, or along any of the four diagonal movements. The advantage of using chain codes is that it is *lossless*, meaning that the original image can be reconstructed given the chain code, and it also results in a significant amount of *compression*, meaning that less information needs to be stored about the image, saving memory or bandwidth if the image is transmitted between computers. The first approach for representing images using chain codes was introduced by Freeman in 1961, and is known as the *Freeman Chain Code of Eight Directions (FCCE)*. Chain codes can also be used in *character recognition*, in order to determine if written characters with slight variations are the same character.

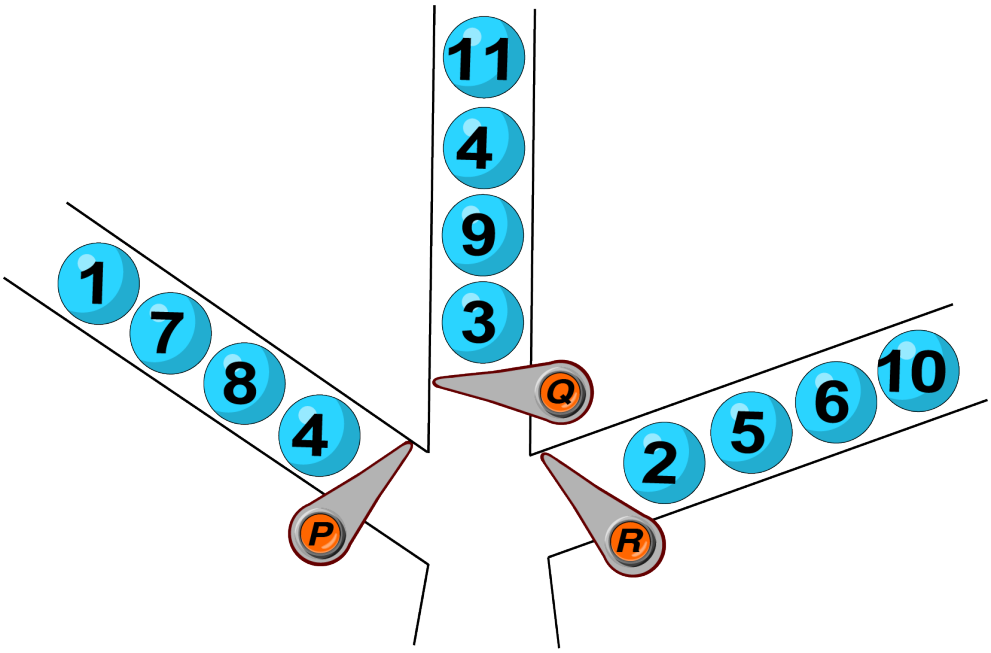
Country of Original Author

Lithuania

Push the Button

Story

Numbered balls are stored in the device shown below. Pushing one of the buttons P, Q or R causes its gate to open and the first ball behind that gate to drop.



Question

What is the maximum number of button pushes that result in balls being dropped in increasing (but not necessarily consecutive) order?

(A) 6

(B) 7

(C) 8

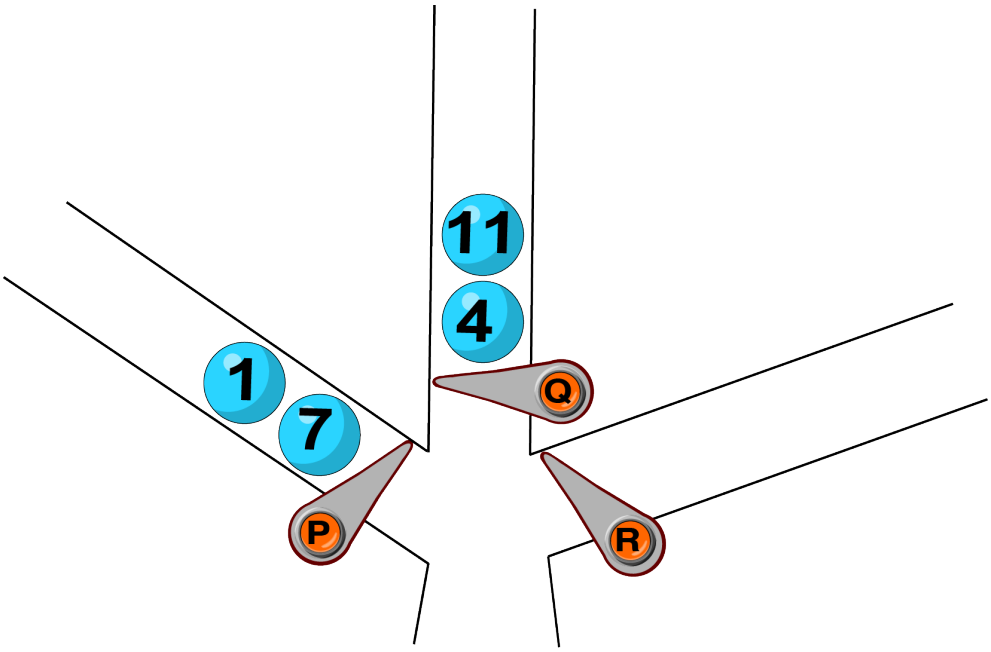
(D) 9

Answer

(C) 8

Explanation of Answer

If the buttons are pressed in the order: R, Q, P, R, R, P, Q, R, then the balls will be dropped in the order: 2, 3, 4, 5, 6, 8, 9, 10. Thus, we have shown a way for eight button pushes to result in eight balls being dropped in increasing order. After this, the remaining balls will be positioned as shown.



Since neither 7 nor 4 is greater than 10, any additional button pushes will result in the dropped balls no longer being in increasing order.

In fact, no matter what order the buttons were pressed in, if the ball numbered 7 drops, the balls will not drop in increasing order because 7 is less than 8 and the ball numbered 7 is behind the ball numbered 8. Moreover, if the ball numbered 7 does not drop, then the ball numbered 1 which is behind it also does not drop. The same reasoning holds for the balls numbered 4 and 11 which tells us there are four balls that cannot drop if the balls drop in increasing order. Therefore, since there are twelve balls stored in the device, at most $12 - 4 = 8$ balls can drop in increasing order. Since we already showed that it is possible for eight button pushes to result in balls dropping in increasing order, the desired maximum number of button pushes is eight.

Connections to Computer Science

This task focusses on the concept of *merging*, which is a crucial component of a well-studied *sorting algorithm* called *mergesort*. The key idea in mergesort is that if we have two sorted lists of numbers, such as $A = [1, 6, 9]$ and $B = [2, 4, 5, 10]$, we can merge them into sorted order by always selecting the smaller of the two values at the front the list, and then making one step into the list that we selected from. For example, in the two lists mentioned above, we select 1 from list A, since it is smaller than 2 and then step to the value 6 in list A. Next we would select 2 from list B since it is smaller than 6. We repeat this process until all the elements have been selected or one of the lists is empty, which would yield the list $[1, 2, 4, 5, 6, 9, 10]$ which is in sorted order. In most *programming languages*, these lists would be stored in a *data structure* called an *array*.

This task alters the merging process to perform a *three-way merge*: the process is similar to merging outlined above, but instead of two, we have three sorted arrays. The smallest unpicked elements from each of these lists are compared and the smallest one is chosen. This process repeats until all elements from all arrays have been picked. This results in a merged, sorted array.

However, this task further alters the merging process because the balls held with buttons P, Q, and R are not all in sorted order: both P and Q are holding elements that are out of numeric order. If we apply the three-way merge process here, the bottom unpicked ball from each of the buttons P, Q, and R is compared. The ball with the smallest label is chosen and moved to the storage area. This process repeats until no ball can be moved. As a result, the balls in the storage area would be sorted in ascending order from bottom to top, which is the required condition.

Country of Original Author

Montenegro

Part B

Eating Carrots

Story

Four carrots are growing in four soil patches as shown.

Rabbits are capable of the following three actions:

Hop to the soil patch immediately to the left of the current soil patch.

Hop to the soil patch immediately to the right of the current soil patch.

Eat the carrot growing in the current soil patch.

Earl the rabbit started in one of the four soil patches, but we do not know which one. We do know that Earl never jumped left of the leftmost soil patch nor right of the rightmost soil patch.

In addition, we know that Earl's sequence of actions was:

Question

Which image below shows the soil patches and the one uneaten carrot after Earl finished his sequence of actions?

(A) 

(C) 

(B) 

(D) 

Answer

(D) 

Explanation of Answer

We will number the soil patches from left to right.

One way to solve this problem is to notice that Earl started by hopping to the right, so he couldn't have started on the fourth soil patch. Afterwards he hopped left three times, so he must have started on the third soil patch.

Alternatively, if Earl started on the first soil patch, then his fifth action would cause him to jump left of the leftmost soil patch. If Earl started on the second soil patch, then his sixth action would cause him to jump left of the leftmost soil patch. If Earl started on the fourth soil patch, then his first action would cause him to jump right of the rightmost soil patch. Therefore, it must be the case that Earl started on the third soil patch.

From the third soil patch, Earl's first two actions would result in eating the fourth carrot. The next two actions would result in eating the third carrot. And the remaining three actions would result in eating the first carrot. The second carrot was not eaten by Earl.

Connections to Computer Science

This task outlines an *algorithm* of how Earl moves through the soil patches. An algorithm is a finite sequence of instructions to accomplish some task. In this algorithm, there are two kinds of instructions: moving and eating. In many *programming languages*, there are instructions to *assign* values to *variables*, to *compare* values, to *repeat* instructions, or to *conditionally* perform instructions if a given condition is true.

In order to determine the initial location of Earl the rabbit, it is necessary to *trace* the given sequence of instructions while observing the *state* of where Earl is. This process of determining the consequences of the algorithm is an aspect of *debugging* a program. Debugging is the process of determining which instruction(s) in a computer program are incorrect, in order to determine why the output or final state of the algorithm is incorrect.

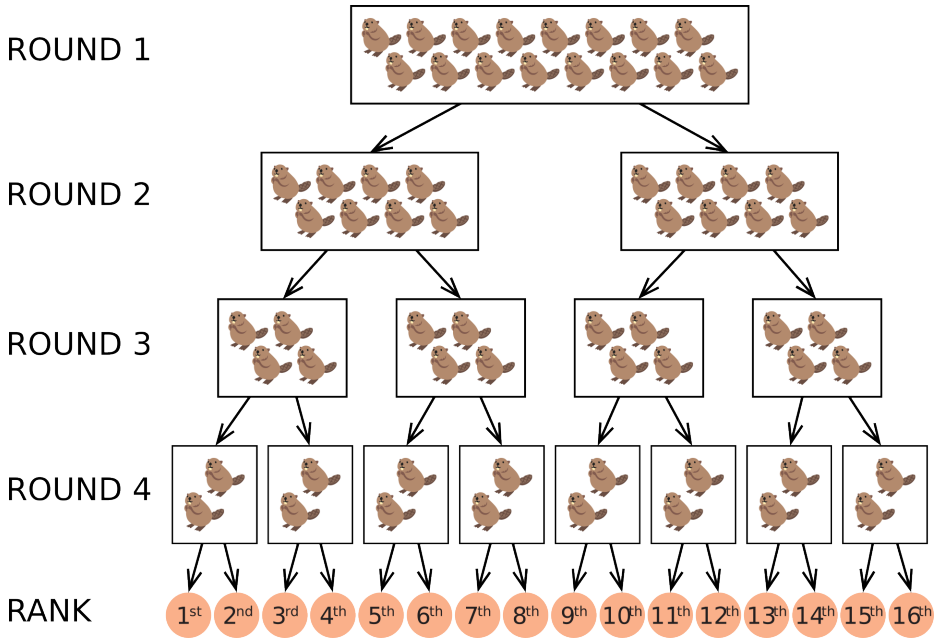
Country of Original Author

Slovakia

Bebras Ball

Story

Players are ranked from 1st place to 16th place based on their performance in a Bebras Ball tournament. The tournament consists of four rounds. All the players are grouped together for the first round, and divided into smaller groups after each round as shown. Winning players follow the left arrow to their group in the next round (or final rank). Losing players follow the right arrow to their group in the next round (or final rank).



For example, a player who wins during rounds 1 and 2, but loses during rounds 3 and 4, will receive a rank of 4th place.

Question

If Noro played in the tournament and lost during exactly one round, which of the following ranks could he **not** receive?

(A) 2nd

(B) 3rd

(C) 5th

(D) 7th

Answer

(D) 7th

Explanation of Answer

Since there are four rounds, and Noro lost during exactly one round, there are four possible scenarios.

Scenario 1: Noro lost during round 1 and won during all others. Thus, Noro would follow the arrows "right, left, left, left" which leads to the rank of 9th place.

Scenario 2: Noro lost during round 2 and won during all others. Thus, Noro would follow the arrows "left, right, left, left" which leads to the rank of 5th place.

Scenario 3: Noro lost during round 3 and won during all others. Thus, Noro would follow the arrows "left, left, right, left" which leads to the rank of 3rd place.

Scenario 4: Noro lost during round 4 and won during all others. Thus, Noro would follow the arrows "left, left, left, right" which leads to the rank of 2nd place.

To be ranked in 7th place, Noro would have to follow the arrows "left, right, right, left" which indicates that Noro would have lost during exactly *two* rounds.

Connections to Computer Science

The diagram in this task, which models the tournament, is a type of structure called a *decision tree*. The top of the tree (or *root*) is where the decision process begins. From there, we select a branch to follow depending on the answer to a decision question. In this task, the decision question is "did I win or lose during the round?" Answers of "win" mean we follow the left branch and answers of "lose" mean we follow the right branch. When we run out of branches we say that we have reached the *leaves* of the decision tree. The leaves represent the final outcomes. In this task, the final outcomes are the ranks.

If you have ever tried to identify someone's secret number by making a guess and having them respond with "higher" or "lower", you have also used a decision tree.

Decision trees are used in computer science in *artificial intelligence*, specifically in *machine learning*. For example, when a program is being designed to distinguish between various images, such as "dog", "fish", or "traffic light", various features such as "rectangular shape", "two eyes", and "fins" are used as decision questions. With repeated *training*, the program develops enough distinguishing features to correctly classify an image.

Country of Original Author

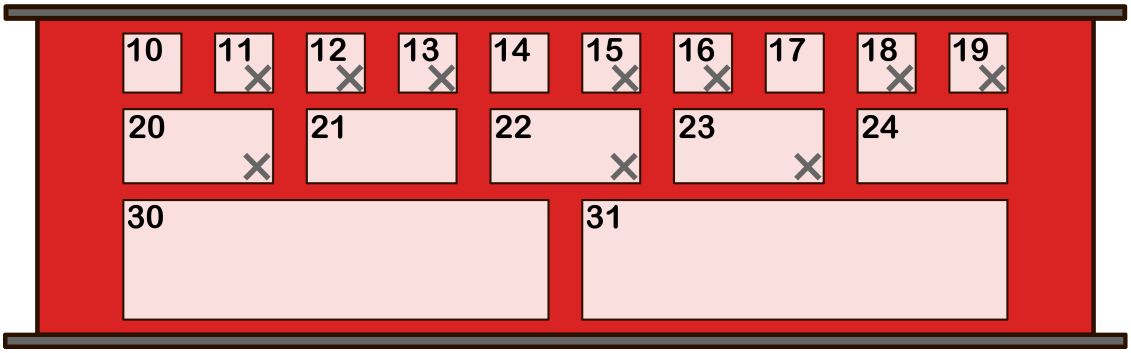
Slovakia

Lockers

Story

When packages arrive at the post office they are placed in lockers to await pick up. The top row of lockers can only hold small packages. The middle row of lockers can hold small or medium packages. The bottom row of lockers can hold packages of any size. Each locker can only hold one package at a time.

The following image shows what the lockers at the post office currently look like. Lockers marked with an X are holding a package.



When a new package arrives, it is placed in the lowest-numbered available locker in which it can fit. When a customer arrives to pick up a package from a locker, the locker becomes available again.

The post office has opened for the day and the following five events occur in this order:

- Four small packages arrive.
- The packages in lockers 11 and 19 are picked up.
- Two medium packages arrive.
- The packages in lockers 20 and 21 are picked up.
- Two small packages arrive.

Question

Then one more small package arrives. In which locker is it placed?

(A) 20

(B) 19

(C) 24

(D) 17

Answer

(A) 20

Explanation of Answer

We can keep track of the available lockers before and after each of the five events.

Event	**Notes**	**Available Lockers**
Post office opens		10, 14, 17, 21, 24, 30, 31
4 small packages arrive	They occupy lockers 10, 14, 17, 21	24, 30, 31
Pickup from lockers 11 and 19	This frees lockers 11, 19	11, 19, 24, 30, 31
2 medium packages arrive	They occupy lockers 24, 30	11, 19, 31
Pickup from lockers 20 and 21	This frees lockers 20, 21	11, 19, 20, 21, 31
2 small packages arrive	They occupy lockers 11, 19	20, 21, 31

The small package that arrives next is placed in the lowest-numbered available locker, which is locker number 20.

Connections to Computer Science

The key computer science concept outlined in this task is *memory allocation strategies*. Memory allocation strategies are used by the *operating system*, which executes various *applications* or *programs* and assigns each application/program an amount of memory in *random access memory (RAM)*. One such strategy is the *best-fit* strategy, where the smallest available block of memory that is large enough for the program/application is the one that is chosen. The best-fit strategy is what is used in this task: the lowest-numbered empty locker that is large enough to hold the package is chosen.

There are other allocation strategies, such as *first-first*, where the first available slot is chosen, and *worst-fit*, where the largest available slot is chosen. It is worth noting that for **any** allocation strategy, there is a sequence of allocations and deallocations that will cause an *out of memory* error, even though there is enough available memory. For example, in this task, if 7 small packages arrive, then all lockers will be allocated. Then, even if all the lockers from 10 to 19 are emptied, there is no room for a medium sized package. When this happens, we say that memory has been *fragmented*. In operating systems, fragmentation can be be resolved by *paging* blocks of allocated memory into smaller units, or by *compacting* blocks to free up memory.

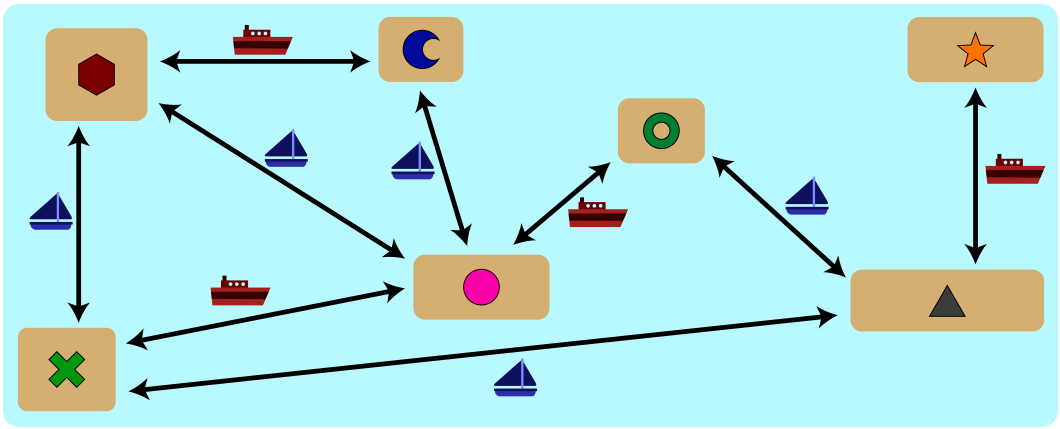
Country of Original Author

Iceland

Island Vacation

Story

Ravi is on vacation in the Island Kingdom. On the map, each island is marked with a different shape, and the routes between islands are marked with the type of boat used on the route. There are two types of boats:  and  .



Ravi started at the island marked with  and traveled to the island marked with  , possibly visiting some islands more than once.

Question

Which of the following sequences of boats could Ravi **not** have taken?

(A) 

(B) 

(C) 

(D) 

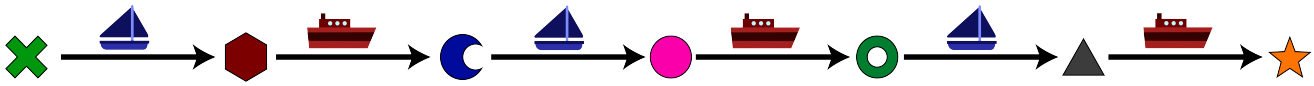
Answer

(D) 

Explanation of Answer

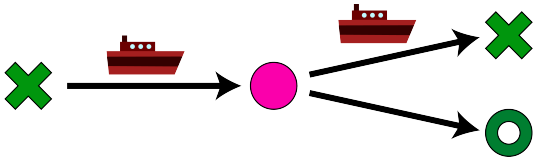
The sequence of boats in Option A can be achieved with the following route:

The sequence of boats in Option B can be achieved with the following route:

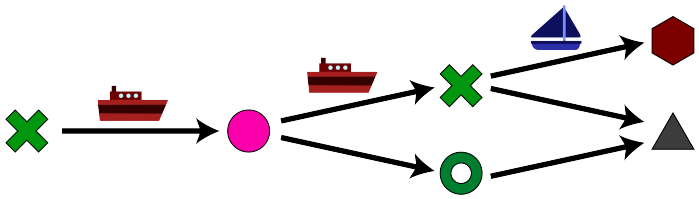


The sequence of boats in Option C can be achieved with the following route:

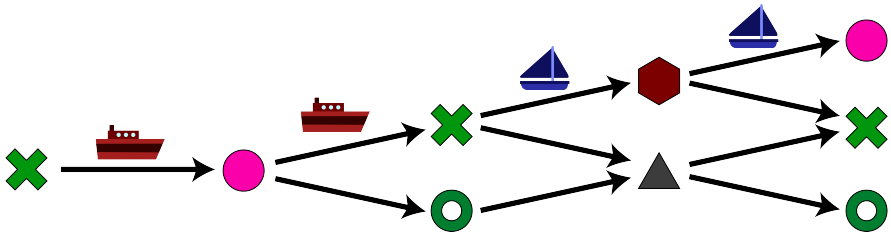
We will try to find a route using the sequence of boats in Option D. From  , Ravi could have taken  to only  . From  , Ravi could have taken  to either  or  .



From  , Ravi could have taken  to either  or  . From  , Ravi could have taken  only to  .



From  , Ravi could have taken  to either  or  . From  , Ravi could have taken  to either  or  .



Using the sequence of boats in Option D, Ravi would have to then take  to  . However, there is no route between  ,  , or  and  . Therefore, it is not possible to travel from the island marked with  to the island marked with using the sequence of boats in Option D.

Connections to Computer Science

The diagram describing the movement of boats between islands is an example of a *graph*. Each island is represented as a *vertex* of the graph. Each *edge* will connect one island to another island. As well, each edge is *labelled* with the type of boat that travels between the two islands.

In this task, we want to determine valid *paths* between the two islands, given a sequence of edges. One way to find such paths is to perform a *breadth-first search*: find all of the *neighbours* of Island  where the edge is valid (i.e., has label ), and then repeatedly determine the neighbours of each of those neighbours, until Island  is reached.

The idea of searching for a path in a graph has many applications, such as mapping out a driving route, determining how to send information through the internet, and determining recommendations for who you may want to connect with on social media platforms.

Country of Original Author

China

Moving Documents

Story

Every day, employees of Beaver Logistics must move documents in boxes from one site to another. It takes 1 minute per box to load the truck, 1 minute per box to unload the truck, and it is a 50 minute round trip between the two sites. The truck can hold at most 20 boxes and so moving more than 20 boxes requires more than one trip back and forth.

At the start of each day, there are 36 boxes of documents to be moved. However, it is possible to spend some time reorganizing to reduce the total number of boxes that then need to be moved.

- On Monday, Alia did a lot of reorganizing before moving the documents on Monday. It took her 2 minutes per original box to reorganize, but when she was done she had eliminated half the boxes.
- On Tuesday, Bala did some reorganizing before moving the documents. It took him 1 minute per original box to reorganize, and when he was done he had eliminated a third of the boxes.
- On Wednesday, Yoko did no reorganizing at all before moving the documents. She had to move all the original boxes.

Question

Who was able to move all the documents (including any time spent reorganizing), and return to the starting site in less than 3 hours?

(A) Alia and Yoko

(B) Alia and Bala

(C) Bala and Yoko

(D) Alia, Bala and Yoko

Answer

(A) Alia and Yoko

Explanation of Answer

For each employee we can calculate how much time is needed to move 36 boxes.

Employee	**Time to Reorganize**	**Number of Boxes**	**Time to Load**	**Time to Unload**	**Number of Trips**	**Driving Time**
Alia	72 min	18	18 min	18 min	1	50 min
Bala	36 min	24	24 min	24 min	2	100 min
Yoko	0 min	36	36 min	36 min	2	100 min

The total time needed for each employee is then:

- Alia: $72 + 18 + 18 + 50 = 158$ min (2 hours and 38 minutes)
- Bala: $36 + 24 + 24 + 100 = 184$ min (3 hours and 4 minutes)
- Yoko: $0 + 36 + 36 + 100 = 172$ min (2 hours and 52 minutes)

Therefore, Alia and Yoko were able to move 36 boxes is less than 3 hours.

Connections to Computer Science

This task has connections to *data compression*, which is an operation that is useful when storing information on a data storage device, such as a personal computer or in a data centre, and also to *data transmission*, which is when information is transferred between a source and a destination, such as when a web server sends information to a smartphone.

One component of compression and decompression is that they both require time to complete and, therefore, it is important to make an appropriate data compression choice before transmitting data. If D is the original data and C the compressed data, it would be appropriate to compress the data if the time required to transmit D would be greater than the sum of the times required to compress D into C on the sender side, transmit C, and decompress C back into D on the receiver side.

In this task, decompression time has not been considered in order to simplify the task. This task is equivalent to, for example, compressing data for backup storage in a *data centre* where the data are never decompressed on the receiver side.

This task is also related to *packet segmentation* because the truck can transport only 20 boxes at a time. Packet segmentation means that when the data is larger than the transmission unit supported by the network, the data will be divided into smaller units for transmission. For example, when a large file is transmitted over the internet, it is divided into many small packets by the sender, transmitted, and reassembled by the receiver.

Country of Original Author

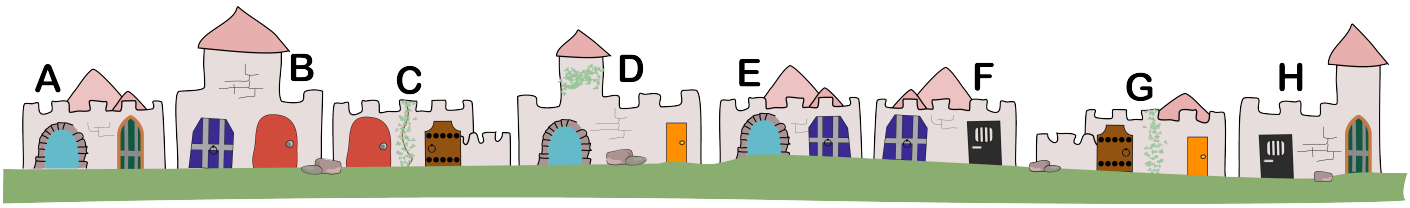
Belgium

Part C

Magical Doors

Story

There are eight buildings, labelled A through H, along a road as shown below.



The only way to travel between the buildings is by using magical doors. There are seven different types of doors:

Each building has two different doors. When you exit a building through one of its doors, you can then enter any of the other buildings that have a door of the same type.

For example, if you exit building A via the leftmost door  , then you can enter either building D or building E, and if you exit building A via the rightmost door  , then you will enter building H.

Question

If you passed through the fewest buildings possible starting in building A and ending in building C, how many **types of doors** did you travel through?

(A) 2

(B) 3

(C) 4

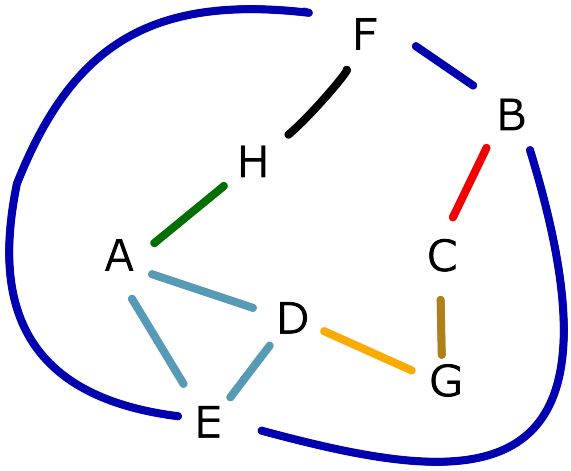
(D) 5

Answer

(B) 3

Explanation of Answer

The following diagram can help us solve and visualize this problem. It consists only of the labels of the buildings and lines connecting two buildings exactly when they have a door of the same type. Notice how the colours of these lines match the colour of the corresponding door.



Now it is easier to see that the only buildings you can reach from building A using one door are buildings H, D and E. From there, the only new buildings you can reach using a total of two types of doors are buildings F, G and B. This means that you cannot travel from building A to C using fewer than three types of doors. However, you can travel from building A to building C using a total of three types of doors. One way to do this is to travel from building A to building D and then to building G before using a third type of door to enter building C.

Connections to Computer Science

As shown in the Explanation of Answer, this task involves finding the *shortest path* in a *graph*. Graphs are composed of a set of *nodes* and a set of *edges* between nodes. In this task, the nodes are the buildings, and the edges connect two buildings which have a door of the same type.

One major application of graphs is to represent maps for driving: each street address is a node and roads are edges. When driving, finding the shortest path is the most common problem to solve. There are several *graph algorithms* which can help us find the shortest path. Two examples of shortest path algorithms are *Dijkstra's algorithm* (used in common GPS systems) and the *Warshall-Floyd algorithm* (used in Google Maps).

Country of Original Author

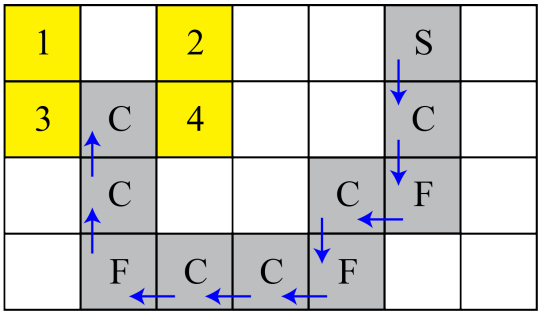
Japan

Closer or Farther

Story

Daniel is playing a game to find treasure buried in a grid of squares. Starting from the square labelled "S", he can only move one step at a time to a neighbouring square. After each step, Daniel receives a signal indicating whether he is now closer to (C) or farther away from (F) the treasure, where the distance is the minimum number of steps it would take Daniel to reach the treasure from his current location.

Daniel plays this game on the following 4-by-7 grid. His path and the signals he receives after each step are shown.



You might notice that Daniel does not always make the best decisions.

Question

In which numbered square is the treasure buried?

(A) 1

(B) 2

(C) 3

(D) 4

Answer

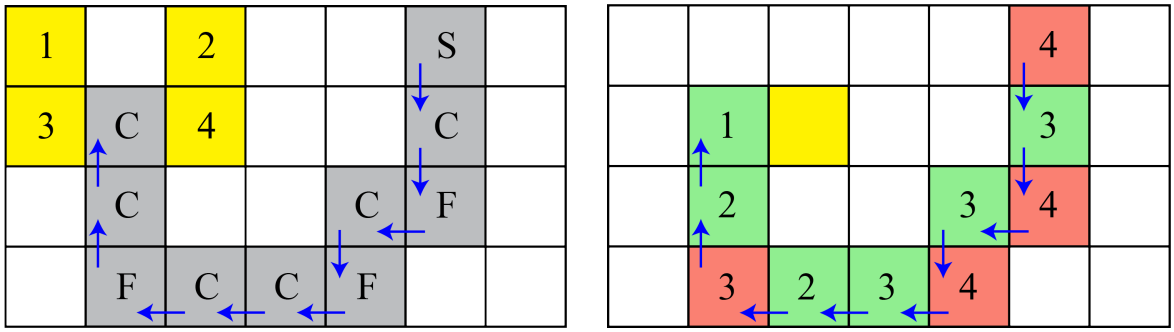
(D) 4

Explanation of Answer

Daniel receives a signal to say that he is closer to the treasure after his first step. Since squares 1 and 2 are farther away after this first step, this means that the treasure cannot be buried at either square 1 or 2.

Now consider the last square at which Daniel received the signal "F". Since square 3 is closer to this square than the previous square in Daniel's path, the treasure cannot be buried in square 3.

Based on the hint Daniel receives, we can conclude that the treasure must be buried in square 4. However, we can also verify that this is consistent with all the signals that Daniel receives. In the new diagram below, the distance from each square Daniel reaches to square 4 is indicated. Every square corresponding to a signal "C" is one where the distance has decreased from the previous square, and every square corresponding to a signal "F" is one where the distance has increased from the previous square.



Connections to Computer Science

This task has connections to the area of *artificial intelligence* known as *machine learning*. In particular, one key technique of machine learning is *reinforcement learning (RL)*, which concerns with how *intelligent agents/learners* should take actions in an *environment* in order to maximize the cumulative *reward*. To evaluate the reward, the machine learning algorithm needs to quantify the "goodness" of a particular state by way of a *value function*.

In this specific task, the new location on the playing board after a move represents the environment, while the signal obtained from the detected distance serves as a reward signal (with 'C' indicating a positive reward and 'F' indicating a negative reward). The player, Daniel, who collects reward signals by making optimal decisions for the next step, serves as the agent. He must consider the possibility of a square containing the treasure, which is similar to the value function.

Autonomous driving is an example of an application of RL. In order to operate successfully in an unpredictable environment where vehicles, pedestrians, and obstacles are encountered at random points in time, an autonomous driving system must perform numerous perception and planning tasks. Each of these tasks needs to created based on the current environment and then dynamically altered based on changes in the environment.

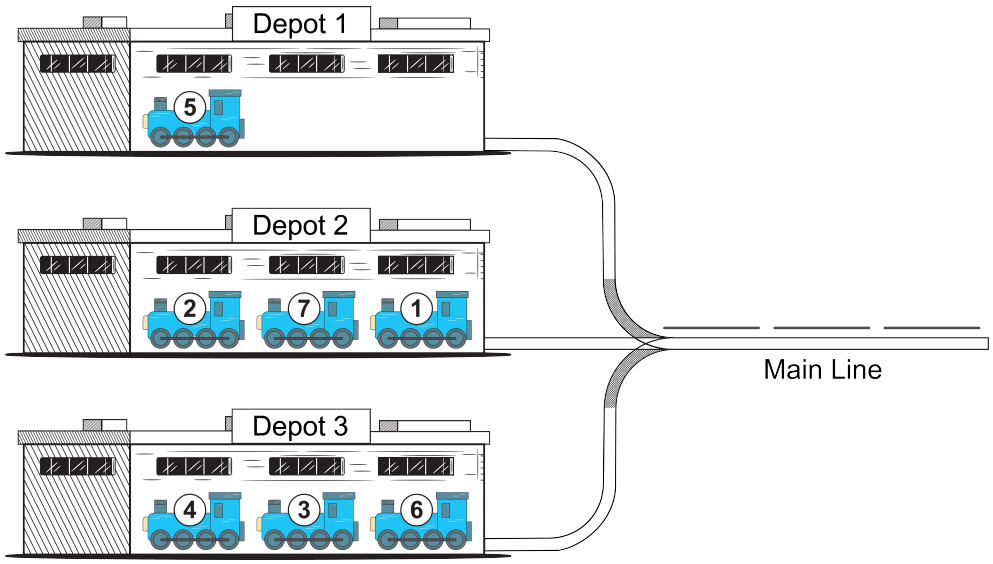
Country of Original Author

Slovakia

Trains

Story

A train station is shown. It currently contains seven numbered trains in three numbered depots. The main line is connected to these depots. The main line and each depot can each hold up to three trains.



Two types of commands result in trains moving between the depots and the main line:

- **OUT(X)**: the rightmost train in Depot X becomes the leftmost train on the main line
- **IN(X)**: the leftmost train on the main line becomes the rightmost train in Depot X

For example, the sequence of commands OUT(3) - OUT(1) - IN(3) - IN(1) will result in trains 5 and 6 exchanging positions.

Question

Which of the following sequences of commands will result in Depot 1 containing trains 1, 2, and 3?

(A) OUT(1) - OUT(2) - IN(1) - OUT(2) - OUT(2) - IN(1) - OUT(3)

(B) OUT(1) - OUT(2) - IN(1) - OUT(2) - OUT(2) - IN(1) - OUT(3) - IN(2) - OUT(3) - IN(1)

(C) OUT(1) - OUT(2) - IN(1) - OUT(2) - IN(1) - OUT(3) - IN(2) - OUT(3) - IN(1)

(D) OUT(1) - OUT(2) - IN(1) - OUT(2) - OUT(2) - IN(1) - OUT(3) - OUT(3) - IN(1)

Answer

(B) OUT(1) - OUT(2) - IN(1) - OUT(2) - OUT(2) - IN(1) - OUT(3) - IN(2) - OUT(3) - IN(1)

Explanation of Answer

The table below shows the trains (in order from left to right) in each depot and on the main line, after each command in Option B is performed.

Command	**Depot 1**	**Depot 2**	**Depot 3**	**Main Line**
	5	2 7 1	4 3 6	
OUT(1)		2 7 1	4 3 6	5
OUT(2)		2 7	4 3 6	1 5
IN(1)	1	2 7	4 3 6	5
OUT(2)	1	2	4 3 6	7 5
OUT(2)	1		4 3 6	2 7 5
IN(1)	1 2		4 3 6	7 5
OUT(3)	1 2		4 3	6 7 5
IN(2)	1 2	6	4 3	7 5
OUT(3)	1 2	6	4	3 7 5
IN(1)	1 2 3	6	4	7 5

Thus, after the sequence of commands in Option B is performed, Depot 1 contains trains 1, 2, and 3.

Option A is incorrect because in order for Depot 1 to contain trains 1, 2, and 3, we must remove the train currently in Depot 1, and add 3 more trains. So our set of commands needs to contain one OUT(1) command, and three IN(1) commands. However the commands in Option A contain only two IN(1) commands, so trains 1, 2, and 3 couldn't all be in Depot 1.

We can see that the first four commands in Option C are the same as in Option B. So using the table above, we see that after the fourth command (OUT(2)), the leftmost train on the main line is train 7. Since the next command in Option C is IN(1), this will result in train 7 moving to Depot 1. Since there is no OUT(1) command after that, we can conclude that train 7 will remain in Depot 1. Thus, Option C is incorrect.

We can see that the first seven commands in Option D are the same as in Option B. So using the table above, we see that after the seventh command (OUT(3)), there are three trains on the main line. Since the next command in Option D is OUT(3), this will result in a fourth train entering the main line. However this is not possible since the main line cannot hold more than three trains. Thus, Option D is incorrect.

Connections to Computer Science

In this task, the trains are moved between the depots and the main line using two operations; OUT(X) and IN(X). The main line and each depot can be thought of as simple *data structures* used in computer science, called *stacks*. A stack follows the *last in, first out* or *LIFO* rule: the item that is placed in last is the first to be removed. There are two main operations for a stack: *push* and *pop*. The push operation adds an element to the *top* of the stack, and *pop* extracts the element at the top of the stack.

In this task, the OUT(X) operation is the pop operation of the depot X stack followed by a push operation on the main line stack. The IN(X) operation is a pop of the main line stack followed by a push of that train to the depot X stack.

Stacks are used frequently in computer science. One such usage is keeping track of operations that can be "undone", such as edits to a document, browsing history in a browser, or searching through a maze by way of *backtracking* if a dead end is reached. When the stacks have fixed depth, as in this task where the depth is at most 3, stacks are useful for problems involving limited resources and physical movement of items such as warehouses.

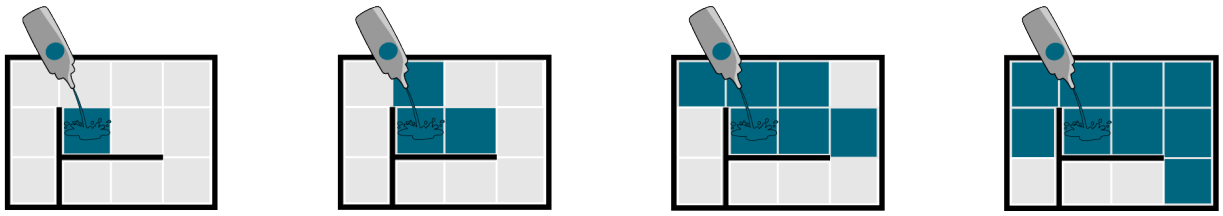
Country of Original Author

Romania

Watercolour

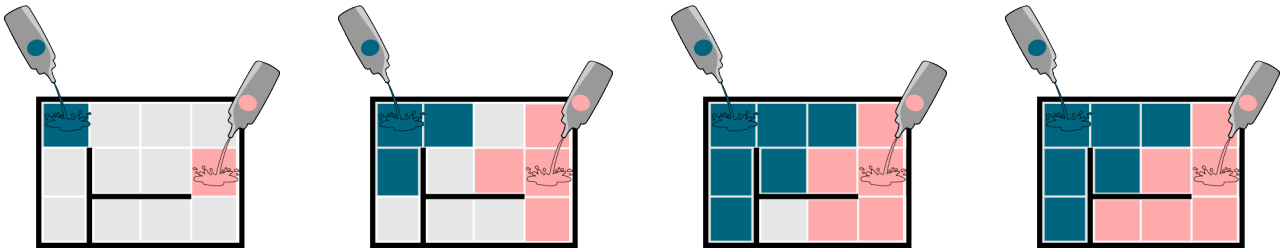
Story

A beaver designs mazes on rectangular grids of squares. To make the mazes more interesting, it can pour watercolour on a square. The colour then spreads. Every second, colour reaches each uncoloured square that shares an edge with a coloured square. However, colour does not spread through walls. Here is an example:



0 seconds 1 seconds 2 seconds 3 seconds

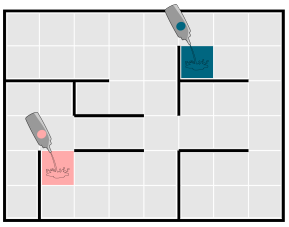
If different colours are used and poured into two different squares, then the first colour that spreads to an uncoloured square will fill it completely and no new colour will be added. If two colours reach a square at the same time, the square takes the darker colour.

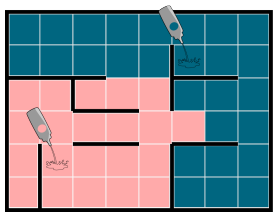


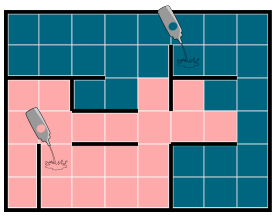
0 seconds 1 seconds 2 seconds 3 seconds

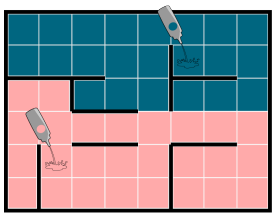
Question

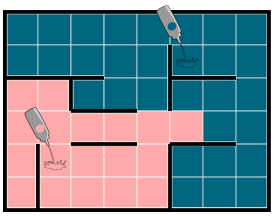
If different colours are poured as shown, what will the maze look like when the maze is filled with colour?



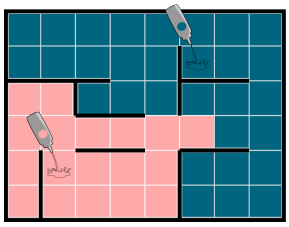
(A) 

(B) 

(C) 

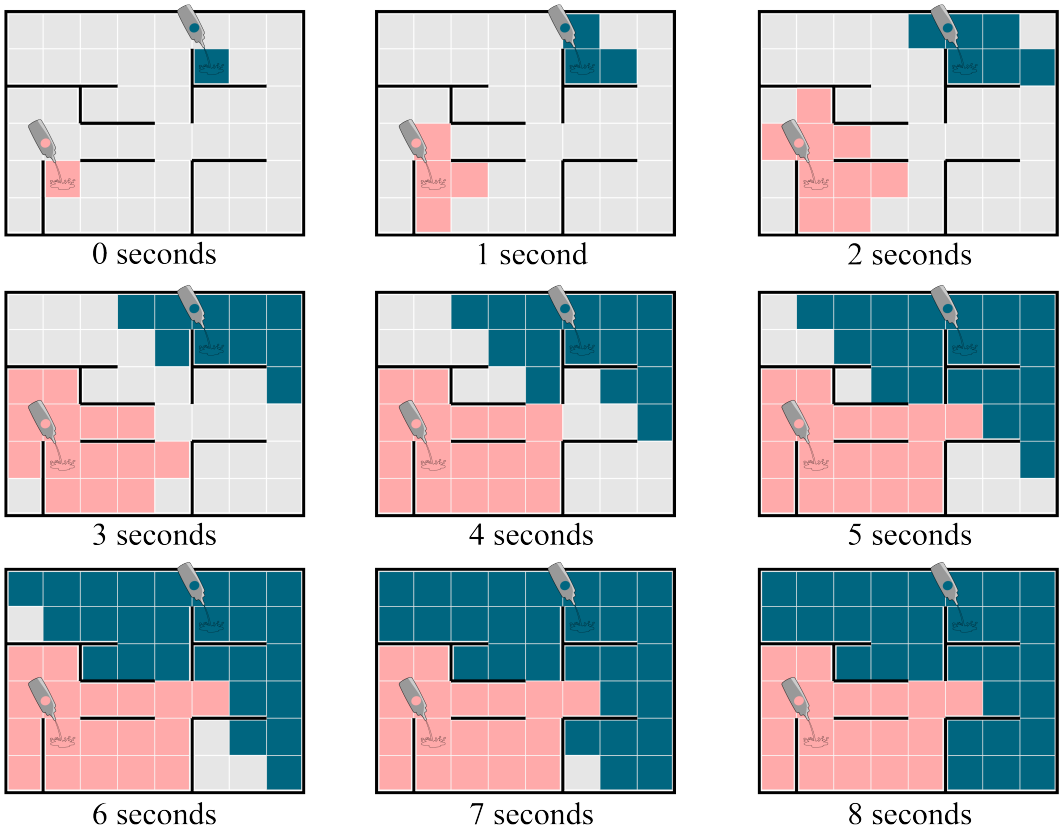
(D) 

Answer

(D) 

Explanation of Answer

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



Thus, when the maze is filled with colour it will look like the maze in Option D.

Connections to Computer Science

In order to solve this task, we can use a technique called *breadth-first search*, and specifically, a variant called the *flood fill* algorithm.

Breadth-first search is an *algorithm* that searches by looking at all immediate "next locations" based on the "current location". In this task, the next locations are the interior spaces behind an exterior wall which is being flooded. This process repeats, with the most recently added "next locations" becoming the "current location". When observing the pictures used in the Explanation of Answer, the entire area is increasingly filled as if it were being flooded from two locations, and thus, the algorithm is known as *flood fill*.

One very common use of breadth-first search is in finding the *shortest path* between two given points, such as when finding the best sequence of directions to drive from one location to another. In this task, we are finding the shortest path from the initial square to the squares that are at the "boundary" between the two colours.

Flood fill algorithms can be found in many graphical drawing programs which contain a "bucket fill" tool that will shade an entire region with a certain colour: the algorithm will colour adjacent *pixels* the same colour until it locates a boundary pixel of a different colour.

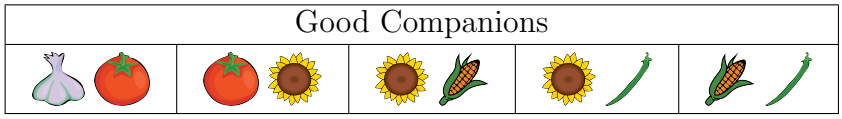
Country of Original Author

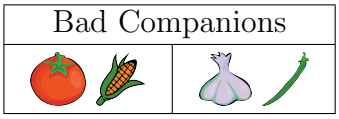
Vietnam

Companion Planting

Story

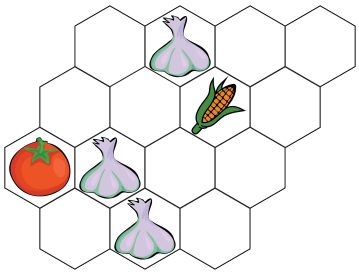
Thalia is planting a garden with garlic, tomatoes, sunflowers, corn, and beans. She wants the plants to help each other grow and knows that some pairs of plants are good companions and some pairs of plants are bad companions:





All other pairs of plants do not affect each other's growth.

There are 15 sections in Thalia's garden bed. She wants to place three of each type of plant in her garden. She has already placed three garlic plants, one corn plant, and one tomato plant as shown.



Question

In how many different ways can Thalia place the remaining plants so that each plant is next to at least one of its good companions, and no plant is next to any of its bad companions?

(A) 1

(B) 2

(C) 3

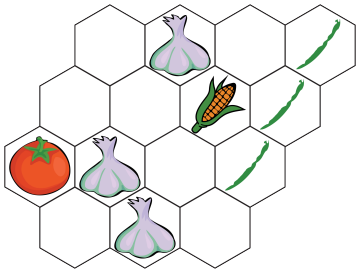
(D) 4

Answer

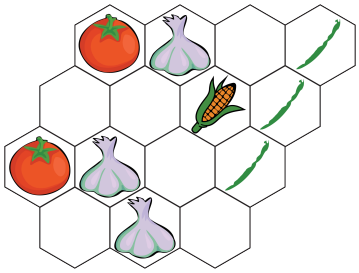
(C) 3

Explanation of Answer

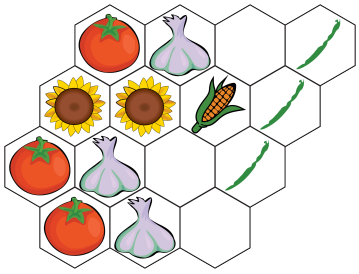
We begin by noticing that there is only one way to place the three bean plants because they cannot be placed next to garlic. This is shown below.



Next, we consider the garlic in the section along the top of the garden. There must be a tomato plant next to it. However, this tomato plant cannot be placed next to a corn plant. Therefore, the top left section must be a tomato plant as shown below.

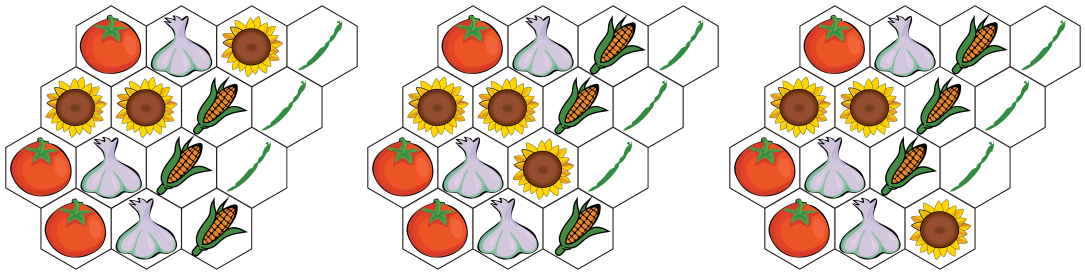


Similarly, the third tomato plant must be placed next to the garlic in the bottom row of the garden. It must be in the bottom left section otherwise there is no way to place the remaining two corn plants so that both are not next to a tomato plant. Since we have only corn and sunflowers left to plant, and corn can't be next to tomatoes, it follows that we must place sunflower plants in the two remaining unoccupied sections of the garden that are next to tomato plants. At this point we have determined that plants must be planted as shown below.



Explanation of Answer Continued

We are left to place two corn plants and one sunflower plant. There are three ways to do this because the sunflower can be placed in any of the three remaining sections. The three ways are shown below and we can verify that in all three cases, the required conditions are met.



Connections to Computer Science

This task can be solved in a variety of ways. One approach is to use the *brute-force algorithm*, where we try placing each of the plants in every square until every condition is met, but this takes a lot of time. Instead, we can reduce the amount of searching to be done by looking for conditions which will reduce the number of possible options.

Underlying this task is the idea of searching in an *implicit graph*. That is, we start with an initial configuration, which is the garden presented in the task. From that point, there are 40 different gardens that can be made by adding one plant. From each of these 40 gardens, there may be many other gardens that can be obtained by adding another plant. As more plants are added, either we move closer to a solution, or a condition will be violated, at which point we *backtrack* to an earlier configuration and not use that particular plant in that particular location.

Many games and puzzles can be solved in this manner, such as sudoku, checkers, or chess. Each board is one *node* in a *graph*, and the placing of a number or piece at a particular location creates a new node. Instead of more standard graphs, there are no explicit *edges* connecting nodes together. Rather, we can deduce the *neighbour* nodes of a particular node by making a "move" in the game or puzzle. Searching in these graphs in an efficient and logical way by using *backtracking algorithms* or other ways of evaluating a "better" node from a "worse" node are used in *artificial intelligence* algorithms for game playing.

Country of Original Author

Australia