UNIVERSITY OF WATERLOO

The CENTRE for EDUCATION in MATHEMATICS and COMPUTING

2021 Beaver Computing Challenge (Grades 7 & 8)

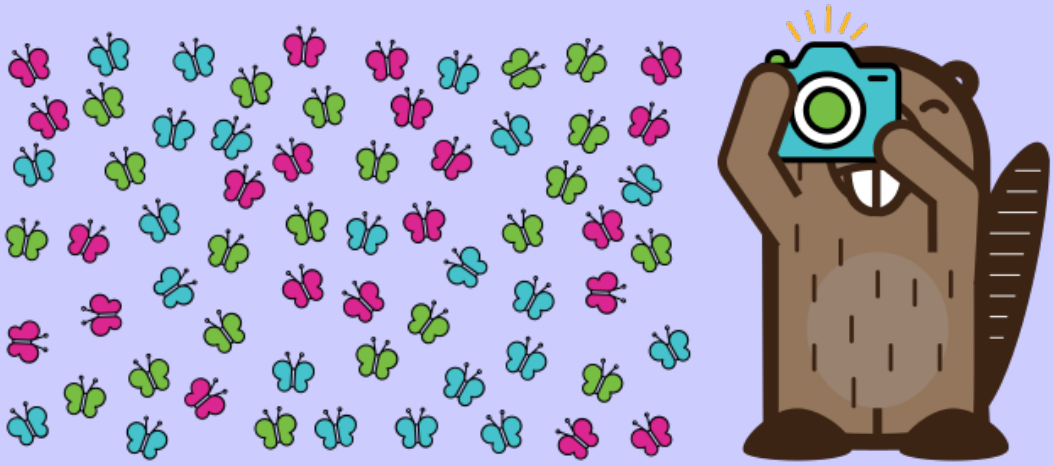
Questions, Answers, Explanations, and Connections

Part A

Butterflies

Story

A beaver is photographing butterflies, but after each photo is taken, half the butterflies fly away.



The first photo has 64 butterflies in it and the last photo has 2 butterflies in it.

Question

How many photos did the beaver take?

(A) 6

(B) 63

(C) 4

(D) 32

Answer

(A) 6

Explanation of Answer

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

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

We see that the photo with 2 butterflies in it is photo number 6. Therefore, the beaver took 6 photos.

Connections to Computer Science

In order to move from 64 butterflies to one butterfly, we need to cut the number in half 6 times. If we started with 128 butterflies, we would need to cut the number in half 7 times before reaching one butterfly, and if we started with 256 butterflies we would need to cut the number in half 8 times before reaching one butterfly.

This process of cutting by half each time decreases the problem size *exponentially*. There are many natural processes that either grow or shrink exponentially: how an invasive species spreads and how a radioactive element decreases its radioactivity are two examples. This idea is used by computer scientists to design *algorithms* which use the *divide and conquer* technique: each major step of the algorithm reduces the size of the problem by half. These sorts of algorithms are very *efficient* because they can take very large inputs and produce an answer very quickly. One famous example of this idea is *binary search* in a sorted list of elements.

Country of Original Author

Canada

Overlapping Coins

Story

Emil has six different coins.

Emil placed the six coins on a table, one at a time. Some coins were placed on top of other coins so that they overlap as shown.

Question

Which coin was the fourth coin that Emil placed on the table?

(A) 

(B) 

(C) 

(D) 

Answer

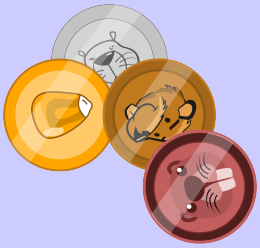
(B) 

Explanation of Answer

To determine the correct answer, we reverse the process.

Notice that the coin  in the bottom-left corner is the only coin that has no other coins on top of it. This means it must have been placed on the table last and was therefore the sixth coin to be placed. Before this coin was placed, the coins on the table must have looked like this:

Now the only coin that has no other coins on top of it is the coin  . This coin must have been placed second to last and was therefore the fifth coin to be placed. Before this coin was placed, the coins on the table must have looked like this:



Now the only coin that has no other coins on top of it is the coin  . This coin must have been placed third to last and was therefore the fourth coin to be placed.

Continuing this process, we find that the coins were placed on the table in the following order:

First Second Third Fourth Fifth Sixth

Connections to Computer Science

The coins in the picture are laid in a *sequence*, and the *order* of the sequence matters.

There are many sequences where the order matters. For example, when getting dressed, putting on shoes should come after putting on pants.

Another example is describing to a computer how to draw a picture. Drawing a circle, then two dots, and then a curved line, will produce a smiley face, as shown in the picture below.

If the order was different, and the circle was drawn last, the two dots and curved line would have been hidden behind the circle.

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

- draw circle at (0,0) with radius 5
- draw dot at (-2,2)
- draw dot at (2,2)
- draw curved line from (-4,1) to (4,1)

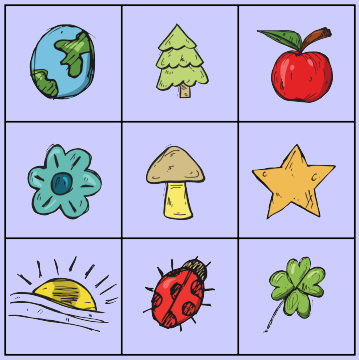
Country of Original Author

Czech Republic

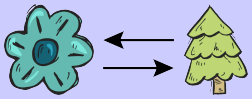
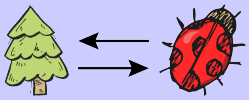
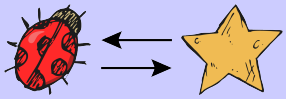
Arranging Objects

Story

A board is divided into squares and a different object is placed in each square as shown.

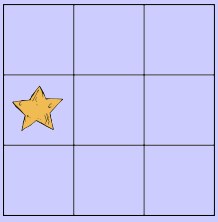


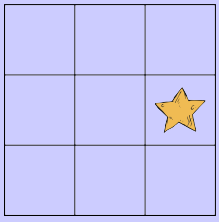
A *swap* exchanges the locations of two objects. Three swaps occur in this order:

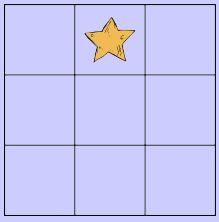
1. 
2. 
3. 

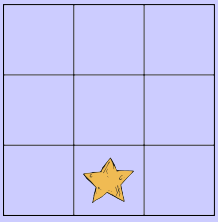
Question

What is the location of  after the last swap?

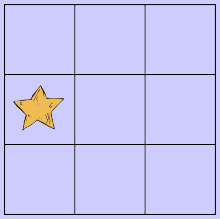
(A) 

(B) 

(C) 

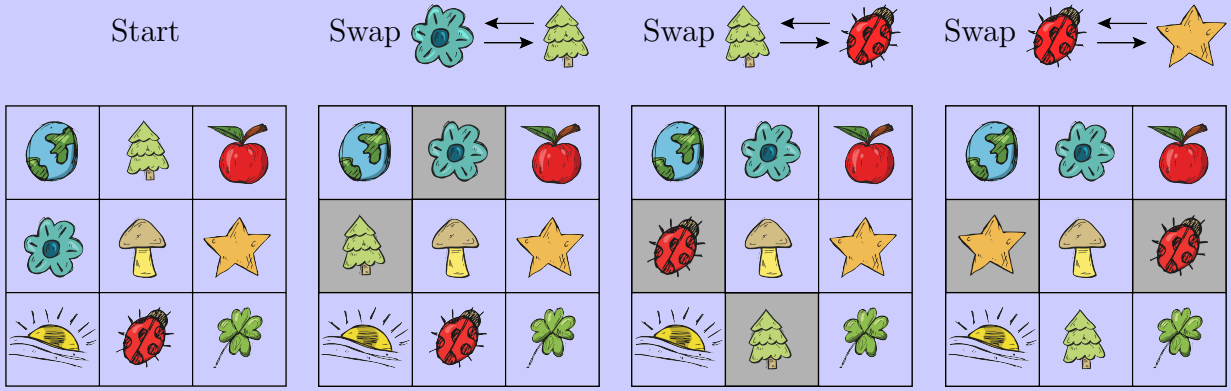
(D) 

Answer

(A) 

Explanation of Answer

Here is the state of the board after each swap:



If you compare the locations of the objects at the beginning, with the locations of the objects after the last swap, you can see that the star is now in the original location of the flower.

Another way to see this final result is to think about trading goods at a market. Suppose you bring a flower and you trade it for a tree. Then you trade your tree for a ladybug. Then you trade your ladybug for a star. The item you end up with is the same as if you traded your flower for a star directly.

Connections to Computer Science

This task focuses on the concept of *swapping* values between two *variables*. In computer programming, a *variable* is a memory location that can hold information. *Swapping* involves exchanging the values of two variables.

For example, suppose A is a variable that holds the value "18" and B is another variable that holds the value "42". After a swap, variable A will hold "42" and variable and B will hold "18".

In most *programming languages*, a *temporary variable* is needed to swap values between two variables. For example, if there was a temporary variable T, the following three steps would swap the values between A and B:

1. Give T the value of A.
2. Give A the value of B.
3. Give B the value of T.

One of the most common uses of swapping in computer science is in *sorting*, where a collection of data it put into either ascending or descending order.

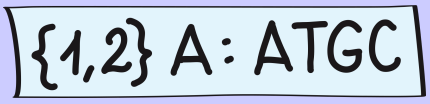
Country of Original Author

India

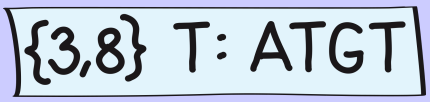
Genetic Data

Story

A genetic scientist is conducting experiments. Each experiment involves a condition followed by a sequence of letters. The condition includes two numbers and a target letter. An experiment is flagged if the number of times the target letter appears in the sequence is between the two numbers (inclusive).



This experiment is flagged because the number of times the target letter A appears in the sequence ATGC is 1 which is between 1 and 2 (inclusive).



This experiment is not flagged because the number of times the target letter T appears in the sequence ATGT is 2 which is not between 3 and 8 (inclusive).

Question

How many of the following four experiments will be flagged?

(A) 1

(B) 2

(C) 3

(D) 4

Answer

(C) 3

Explanation of Answer

The first experiment is flagged because the number of times the target letter T appears in the sequence TTTTTTT is 7 which is between 2 and 8 (inclusive).

The second experiment is not flagged because the number of times the target letter C appears in the sequence AGCTACTAC is 3 which is *not* between 1 and 2 (inclusive).

The third experiment is flagged because the number of times the target letter A appears in the sequence TCGCTGC is 0 which is between 0 and 2 (inclusive).

The fourth experiment is flagged because the number of times the target letter G appears in the sequence GATGTAGCT is 3 which is between 1 and 3 (inclusive).

Three of the four experiments are flagged.

Connections to Computer Science

The concept of *pattern recognition* is a very important one in computer science. The key goal of pattern recognition is to determine if some input, perhaps text or an image, matches or contains a particular pattern. For example, given a large body of text, such a collection of textbooks or entire webpages, determine if a certain word or phrase appears.

DNA can be described as a very long sequence of the letters A, C, G, T. In humans, there are about 3 billion such letters describing DNA. A particular pattern would be a sequence of letters that is known to indicate a certain genetic condition, such as an increased likelihood of a specific disease.

One very common technique to describe patterns in text is the use of *regular expressions*. An example of a regular expression for Canadian Postal codes would be (A-Z)(0-9)(A-Z)(0-9)(A-Z)(0-9). This expression indicates that the first, third, and fifth characters must be uppercase letters, and the second, fourth, and sixth characters must be a single digit. Regular expressions like this are used in text editors for advanced "search and replace" functions.

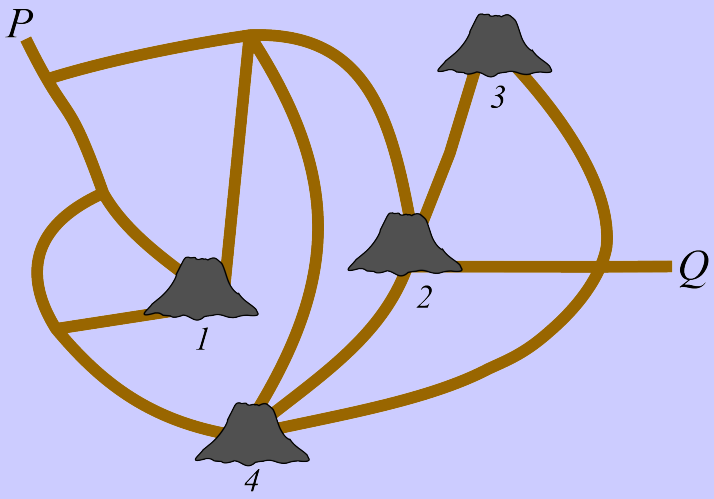
Country of Original Author

Portugal

Volcanoes

Story

In the map shown, Dino can follow roads and can climb up and over volcanoes unless they are erupting.



Because two volcanoes are erupting, Dino cannot get from point P to point Q.

Question

Which two volcanoes are erupting?

(A) Volcanoes 1 and 2

(B) Volcanoes 3 and 4

(C) Volcanoes 1 and 4

(D) Volcanoes 2 and 4

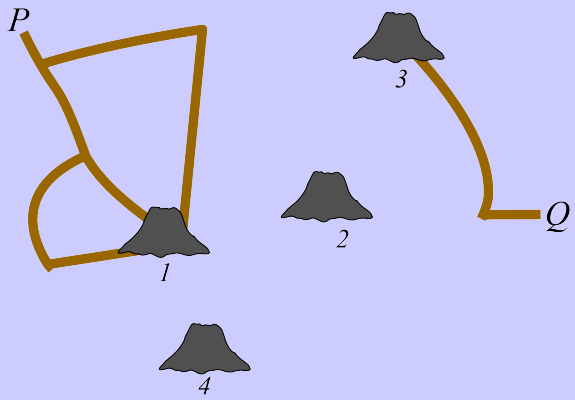
Answer

(D) Volcanoes 2 and 4

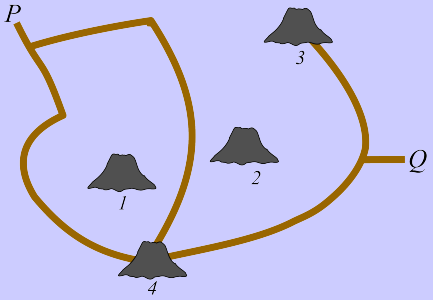
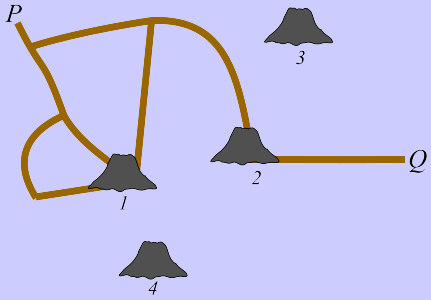
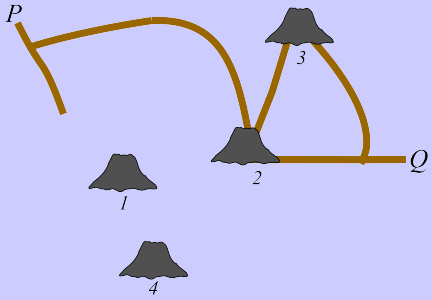
Explanation of Answer

Since Dino cannot climb up and over an erupting volcano, a road to or a road from an erupting volcano is not helpful (at least up to the point where it intersects another road).

In the following image all roads to and from volcanoes 2 and 4 have been removed. Notice that in this situation it is not possible for Dino to get to point Q from point P.



Using the same approach we can see that a route from P to Q does exist for the other three options.

Volcanoes 1 and 2	**Volcanoes 3 and 4**	**Volcanoes 1 and 4**
		
Route over volcano 4 exists	Route over volcano 2 exists	Route over volcano 2 exists

Connections to Computer Science

In computer science, this type of diagram is called a *graph*. The volcanoes are the *vertices* and the roads connecting the volcanoes are the *edges* of the graph.

In the initial graph, points P and Q are *connected*: there is a sequence of edges, known as a *path*, that leads between P and Q.

This task is concerned with finding a set of *cut vertices*, which are also known as *articulation points*. A set of cut vertices has the property that when those vertices and all of the edges attached to those vertices are removed, the graph will become disconnected.

In this problem, no single vertex is a cut vertex: removing just one of the four vertices will not disconnect the graph. However, removing Volcanoes 2 and 4 will disconnect the graph.

One crucial use of finding cut vertices is for determining the *fault tolerance of a network system*: finding the number of routers or servers that would need to go down/offline before some computers in the network can no longer connect to each other.

Country of Original Author

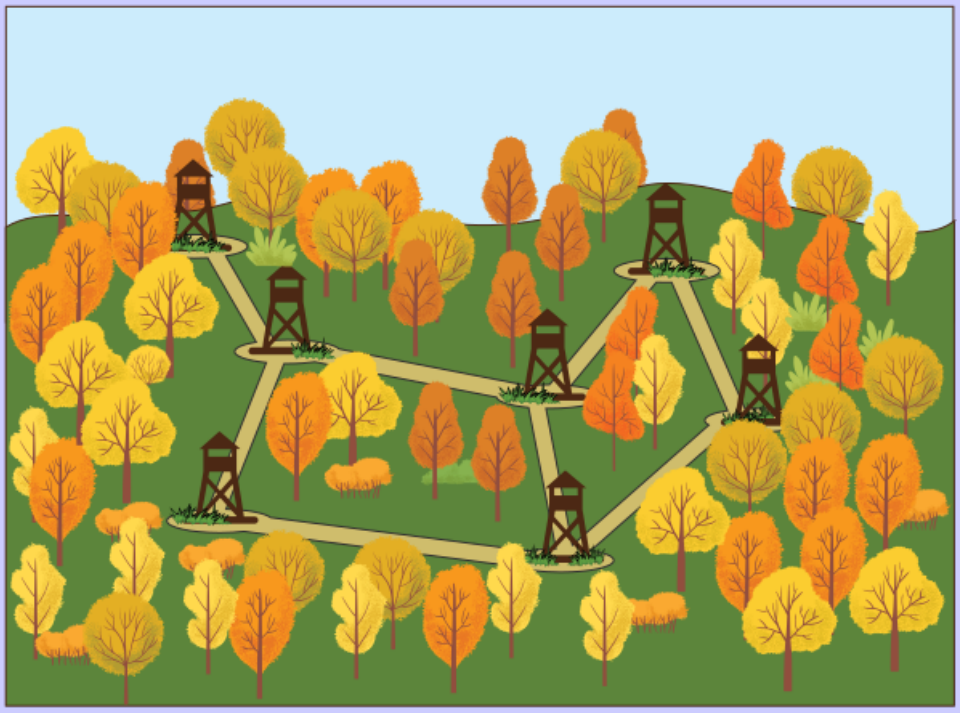
Romania

Part B

Forest Towers

Story

In a forest, there are seven towers and eight paths. Each path connects two towers as shown.



A forest ranger in a tower is able to see all paths that touch that tower, but cannot see any of the other paths. For example, a ranger in the top left tower can only see one path.

Question

What is the smallest possible number of forest rangers that need to be assigned to towers so that each path can be seen by at least one forest ranger?

(A) 2

(B) 3

(C) 4

(D) 5

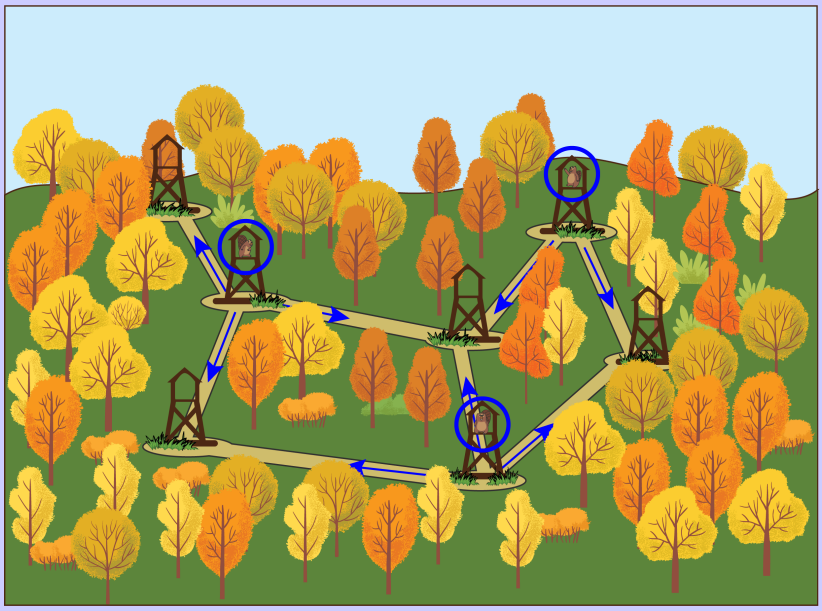
Answer

(B) 3

Explanation of Answer

Notice that from any tower, at most three paths can be seen. This means that if forest rangers have only been assigned to two towers, then at most $3 + 3 = 6$ of the paths will be seen by a forest ranger. Since there are eight paths, at least $8 - 6 = 2$ of the paths will not be seen if there are only two forest rangers. Thus, it is not possible for two forest rangers to see all the paths.

There are several ways to assign forest rangers to three of the towers so that all eight paths can be seen. One such way is to place the forest rangers in the circled towers shown below.



With this placement, all eight paths can be seen by a forest ranger.

Connections to Computer Science

Many problems can be represented with *graphs*. A graph consists of a set of *vertices* (or *nodes*) and a set of *edges*. In this problem, each tower is a vertex and each path is an edge. The problem underlying this task is to determine which set of vertices (i.e., towers) must be chosen so that every edge (i.e., path) is connected to one of the selected towers. This problem is known as the *minimal vertex cover*. One application of this problem is where to place closed circuit cameras in a building to ensure that every hallway can be observed.

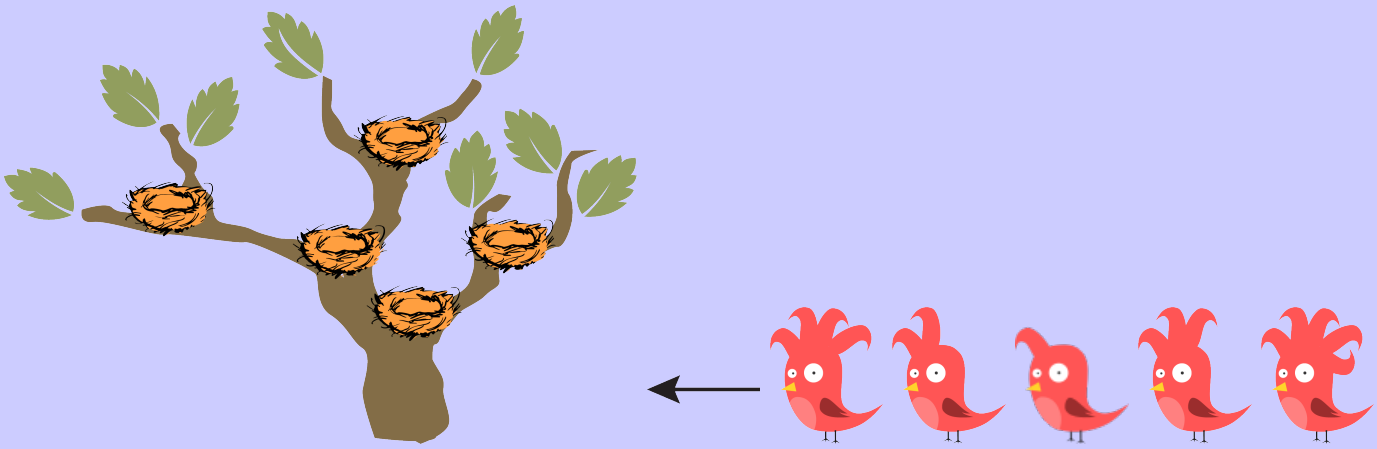
Country of Original Author

Austria

Cuckoo Birds

Story

Cuckoo birds don't build nests. Instead, they move into empty nests. Below is a tree with five empty nests, and a flock of five cuckoo birds.



The birds, *in order from left to right*, each move into an empty nest in the tree. Each bird does this by first inspecting the lowest nest. Then it repeats the following two steps until it finds a nest to move into:

1. If the inspected nest is empty, the bird moves in!
2. If the inspected nest is full, the bird compares its head feathers to those of the bird nesting there.
   (a) If it has *fewer* head feathers than the nesting bird, the bird inspects the first nest found by travelling along the branch extending to the left in the diagram.
   (b) If it has *more* head feathers than the nesting bird, the bird inspects the first nest found by travelling along the branch extending to the right in the diagram.

Question

Which bird moves into the highest nest?

(A) 

(B) 

(C) 

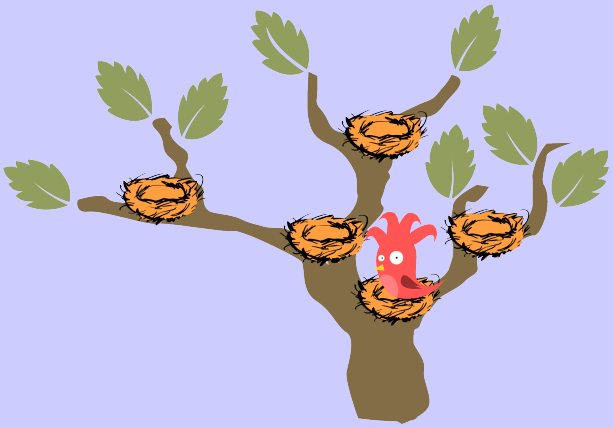
(D) 

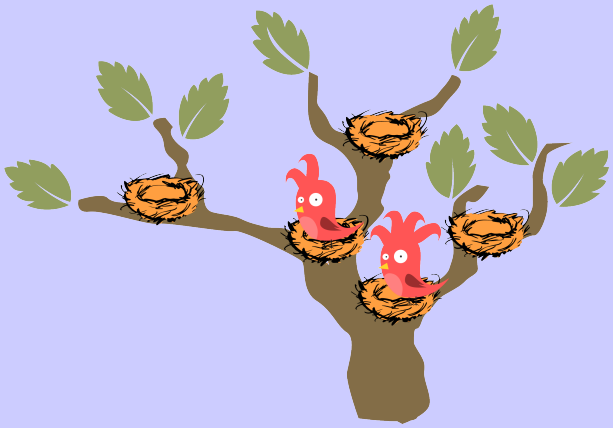
Answer

(A) 

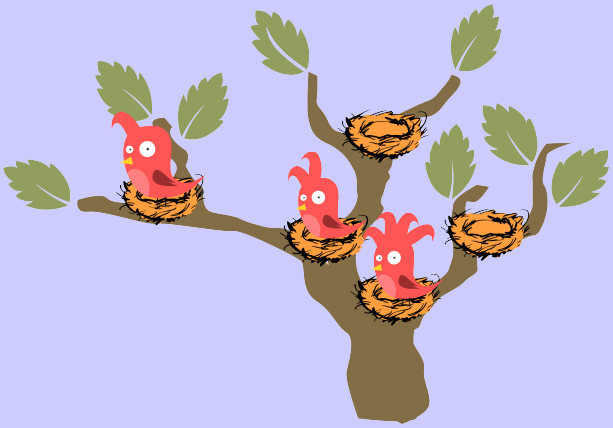
Explanation of Answer

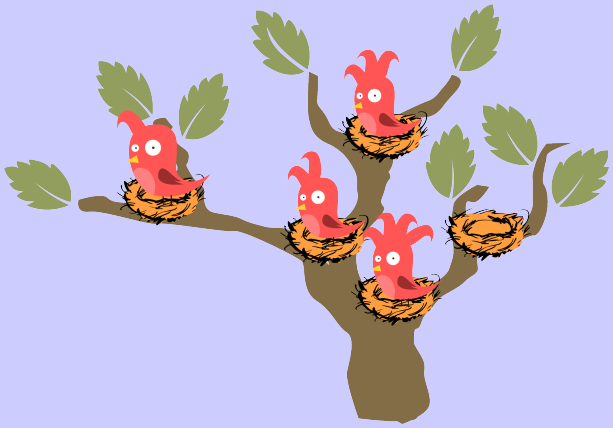
The first bird, with 4 head feathers, moves into the lowest nest. The second bird has 2 head feathers. Since 2 is less than 4, this bird moves into the first nest found by travelling along the branch extending to the left.





The third bird has 1 head feather. Since 1 is less than 4 and 1 is also less than 2, this bird travels left past the two nesting birds and moves into the leftmost empty nest. The fourth bird has 3 head feathers. Since 3 is less than 4, this bird travels left and meets the bird with 2 head feathers. Since 3 is greater than 2, this bird now travels right, and moves into the highest nest.





Connections to Computer Science

In computer science, storing information in useful ways is a crucial step in understanding how to solve problems using a computer. Designing, analyzing, and implementing these various storage mechanisms falls under the area of *data structures* in computer science.

This task relies on one such data structure called a *binary search tree*. A binary search tree stores elements at *nodes* with the following property: all of the nodes to the left of a node are "smaller" than the current node, and all of the nodes to the right of a node are "larger" than the current node. The tree is *binary* in that every node has at most two nodes as it's *children* in the tree. There is one special node, which is the *root* of the tree, which in this task is where the first bird will nest.

Binary search trees allow very *efficient* operations for a *dictionary* of items. Specifically, the three key operations of *insertion*, *deletion*, and *retrieval* (or *lookup*) are very efficient when a binary search tree is used to store the information, even for very large sets of data.

Country of Original Author

Canada

Line of Fish

Story

Fish swim in a line as shown:

Positions are numbered starting from 1 on the left. Occasionally, someone says the positions of two fish. If these positions are A and B where $A < B$, then

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

Positions are renumbered after any fish swim away.

For example, after someone says positions 4 and 9, there would be 6 fish remaining in the line (now in positions 1, 2, ... 6) as shown:

Starting with the original line of 12 fish, suppose that

- someone says positions 2 and 10, then
- someone says positions 3 and 8, and then
- someone says positions 2 and 4.

Question

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

(A) 

(B) 

(C) 

(D) 

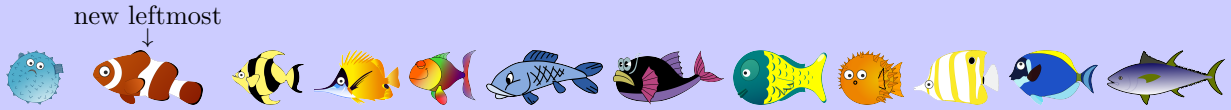
Answer

(D) 

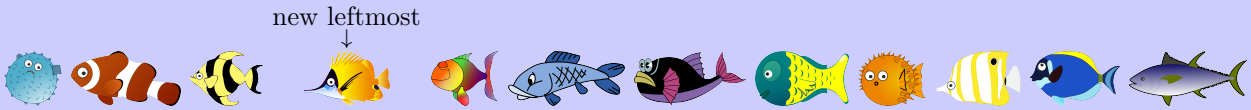
Explanation of Answer

One way to determine the new line of fish is to write down the entire remaining line of fish after each time someone says two positions. However, we can be a bit more clever. The first thing we do is keep track of *leftmost* remaining fish.

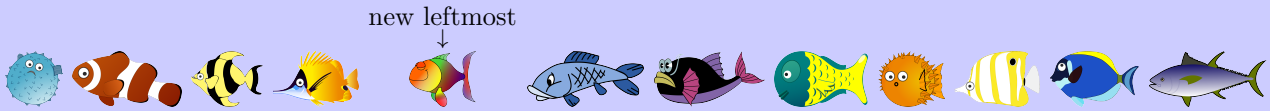
After positions 2 and 10 are called, the leftmost fish will swim away.



Then, after positions 3 and 8 are called, the two leftmost fish will swim away.



Finally, after positions 2 and 4 are called, the leftmost fish will swim away.



Next, we determine how many fish remain in the final new line of fish. Since the last positions to be called are 2 and 4, the only fish in this final new line of fish are the ones at the (renumbered) positions 2, 3 and 4. That is, three fish will remain in the line.

Therefore, the three fish beginning with  form the final new line of fish.

Connections to Computer Science

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

Data types are usually associated with common operations performed on the data. For example, operations associated with sequences include *concatenation* (joining two sequences), *membership* (determining if a certain value is in the sequence), *indexing* (finding the item at a certain position), and *slicing* (creating a subsequence of a given sequence). The key operation in this task is the selection of fish between two given positions, which is an example of slicing in a sequence.

One common application of slicing is in the area of *text processing*: given a collection of formatted data, such as in a *database*, find important subsequences of information to help make decisions.

Country of Original Author

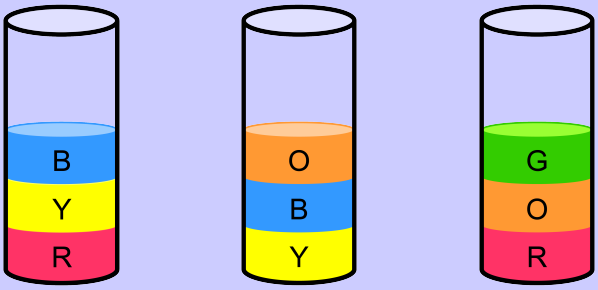
Canada

Colourful Tubes

Story

In science class, Beaver Currie learns that different liquids have different densities. If you pour a liquid into a tube and then carefully pour a different liquid into the same tube, the lower density liquid will stay separate and on top of the higher density liquid.

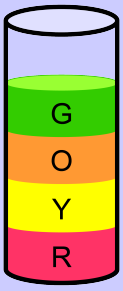
The results of Beaver Currie's three experiments to demonstrate this property are shown.

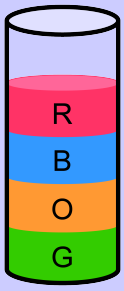


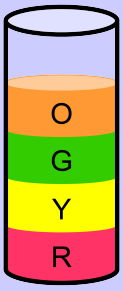
Each different liquid is a different colour and marked with a different letter.

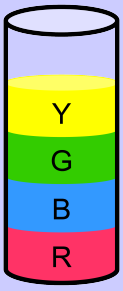
Question

If Beaver Currie used the same liquids in a fourth experiment, which of the following might be the result?

(A) 

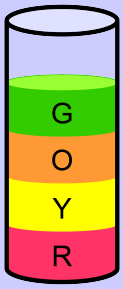
(B) 

(C) 

(D) 

Answer

(A)



Explanation of Answer

We can determine how the density of the liquids compare by inspecting the results of Beaver Currie's first three experiments.

From the first and third experiments, we know that liquid R has a higher density than all other liquids. This rules out Option B.

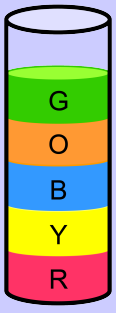
From the second experiment, we know that liquid Y has a higher density than liquids B and O. This rules out Option D.

From the third experiment, we know that liquid O has a higher density than liquid G. This rules out Option C. Only Option A remains, but to give a complete answer, we still need to check if Option A is consistent with the results of the first three experiments.

From the previous two observations, liquid Y must have a higher density than liquid G. Therefore, since it has a higher density than the other three liquids, liquid Y must have the second highest density.

From the second experiment, we know that liquid B has a higher density than liquid O. From the third experiment, we know that liquid O has a higher density than liquid G. These last two observations tell us that the order of these three liquids from highest to lowest density is: B, O, G.

In summary, the order of all liquids according to their density is as follows:



Only in Option A are the layers of the liquids consistent with this ordering.

Connections to Computer Science

This task is fundamentally concerned with *comparisons* and *logical inference*, both of which are key ideas in computer science.

The concept of *comparisons* is used in this task in terms of comparing the relative density of two liquids: the comparison will determine if liquid A is less dense or more dense than liquid B. This is a *binary* comparison: there are only two possible answers (less dense or more dense). Comparisons are the key operation in many operations involving data, such as *searching* or *sorting*.

The concept of *logical inference* is used extensively in *artificial intelligence* and *smart devices*. For example, a temperature measurement on a home thermostat may trigger the heating to turn on. This simple inference model is called *if-then*: if some event occurs, then some other event is triggered. In this task, the if-then model captures the relative densities of various liquids: for example, **if** liquid R is more dense than liquid B, **then** liquid R must be below liquid B.

Country of Original Author

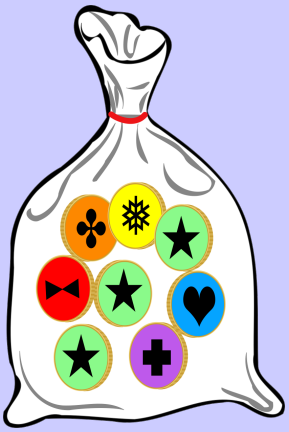
Indonesia

Coin Bag

Story

In Saoirse's country there are four different types of coins. Some coins are the same on both sides, and some are not. The images below show both sides of each type of coin.

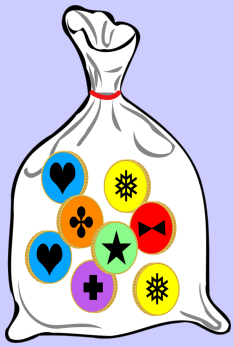
Saoirse has the following bag of coins:

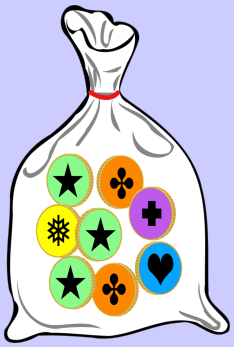


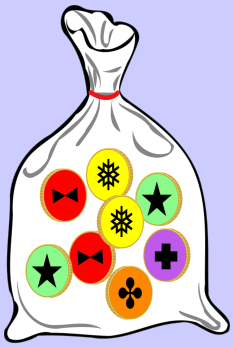
Then the bag is shaken and the coins in the bag move around.

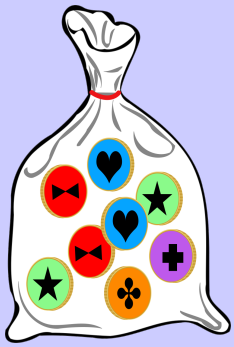
Question

Which of the following could be Saoirse's bag of coins after it was shaken?

(A) 

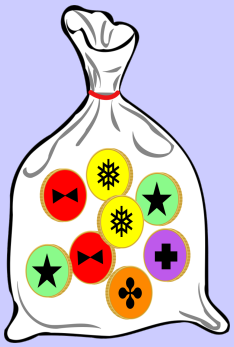
(B) 

(C) 

(D) 

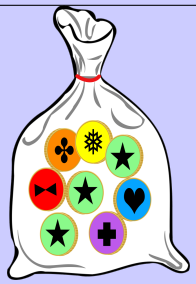
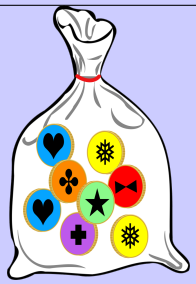
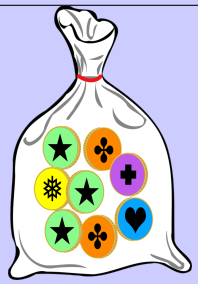
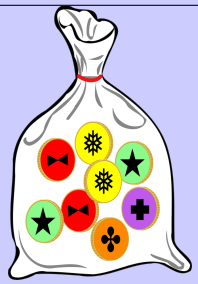
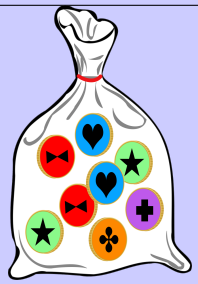
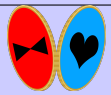
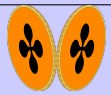
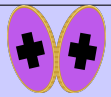
Answer

(C)



Explanation of Answer

Notice that we need to see only one side of each coin to determine which type of coin it is. One way to find the correct answer is to compare the number of coins of each type in Saoirse's bag to the number of coins of the same type in each of the four other bags. This is summarized in the table below.

	 Saorise's Bag	 Bag (A)	 Bag (B)	 Bag (C)	 Bag (D)
	4	3	4	4	2
	2	3	1	2	4
	1	1	2	1	1
	1	1	1	1	1

From the table we can see that Bag (C) is the only bag that has the same number of each type of coin as Saorise's bag. It follows that only Bag (C) could be Saoirse's bag after it was shaken.

Connections to Computer Science

This task is concerned with finding a *bijection* between Saoirse's bag and one of the bags A, B, C, or D. A bijection is a way to pair every element from one set to every element in the other set, so that there are no unpaired elements in either set.

There are $n! = n \times (n-1) \times \cdots \times 2 \times 1$ ("n factorial") different bijections between two sets of size n. In this problem, since there are 8 coins in Saoirse's bag, there are $8! = 40320$ different possible bijections. In order to reduce that number, *constraints* can be applied.

The constraints that can be applied are the number of particular coins of each type in each bag. Specifically, a coin of a certain type must be mapped to exactly the same type of coin in the other bag. This greatly reduces the number of possibilities to examine.

Looking for patterns and eliminating possibilities is a fundamental concept in *constraint programming*, which is a technique used in *artificial intelligence*.

Country of Original Author

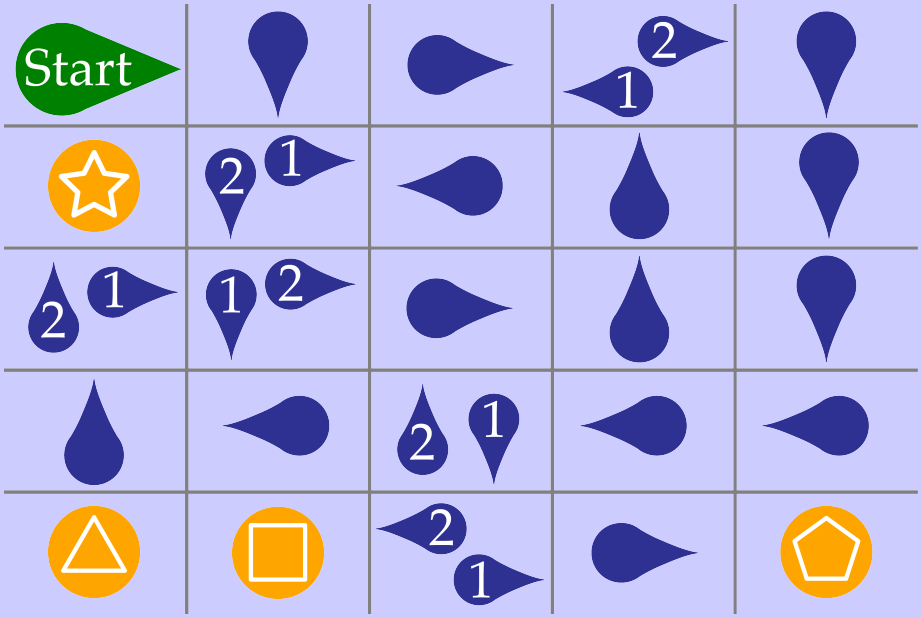
Ireland

Part C

Treasure Hunt

Story

You find the following 5-by-5 treasure map created by pirates. Treasure is hidden at exactly one of the four locations marked by an orange circle.



To find the treasure, you begin at the top left. Then you continually move either up, down, left or right from location to location. When you reach a raindrop symbol, the pointed end of the symbol indicates which direction to move next. For example,  indicates your next move should be to the left.

Some locations contain two raindrop symbols. When this happens, the first time you reach such a location, follow the raindrop labelled with the number 1, and the second time you reach that location, follow the raindrop labelled with the number 2.

The first location you reach marked by an orange circle is where the treasure is hidden.

Question

Which symbol marks where the treasure is hidden?

(A) 

(B) 

(C) 

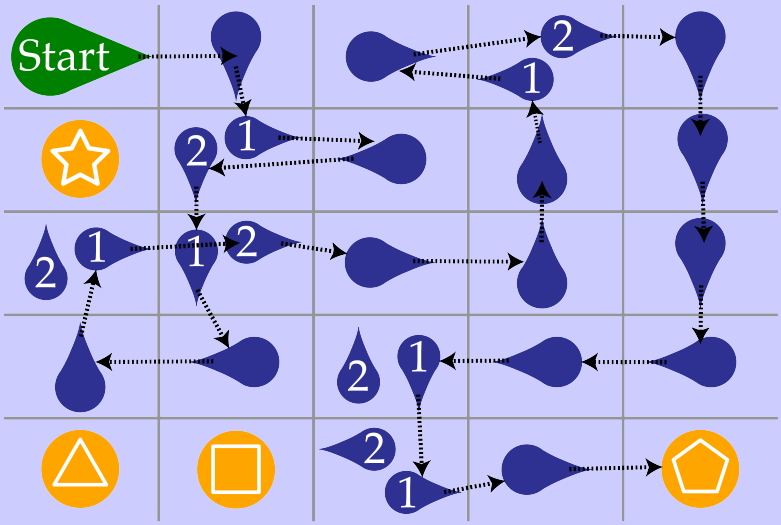
(D) 

Answer

(D) 

Explanation of Answer

The path from the top left location determined by the raindrop symbols is shown below. We are told that the treasure is hidden at the first location reached which is marked by a circle. The first circle reached is the one containing a pentagon.



Connections to Computer Science

This task is focussed on *tracing* and keeping track of *state*, which are important skills in computer programming.

When a computer programmer is *tracing* a program, they are usually carefully following each *statement* in their program, including statements that may cause *branching*. In this task, each location with one raindrop symbol is a *statement* of the form "go here next". The locations with two raindrop symbols are a representation of *branching*: "if this is the first time at this location, go in this direction; otherwise, go in this other direction".

In order to assist with tracing, the *state* of the program must be maintained. In this task, the *current state* is a combination of which location we are in and whether this is the first or second time visiting this location. The *next state* is determined by the direction that the raindrop symbol is pointing in. The task of tracing is to repeatedly determine what state we are currently in and what the next state is.

Country of Original Author

Germany 

Unlock the Crown

Story

A crown is locked in one of 15 drawers as shown.

There is a keyhole at the top of each drawer. To open the drawer, you must insert an object with the same shape as the keyhole. For example, for the keyhole on the top left drawer, you must insert an object shaped like a diamond.

Each drawer contains one object as indicated on the front of the drawer below the keyhole. For example, the top left drawer contains an object shaped like a heart.

Question

Bella has an object shaped like a circle. What is the minimum number of drawers that Bella needs to open in order to retrieve the crown?

(A) 3

(B) 4

(C) 5

(D) 6

Answer

(C) 5

Explanation of Answer

Working backwards from the drawer with the crown, we can determine which drawers Bella must open to retrieve the crown.

1. To retrieve the crown, Bella must open the middle drawer. This means she needs to retrieve a moon.
2. From #1, we know Bella must open the drawer containing a moon. This means she needs to retrieve a star.
3. From #2, we know Bella must open one of the two drawers containing a star. This means she needs to retrieve either a square or a triangle. Since there are no drawers containing a square, she needs to retrieve a triangle.
4. From #3, we know Bella must open the drawer containing a triangle. This means she needs to retrieve a pentagon.
5. From #4, we know Bella must open the drawer containing a pentagon. She can do this using the circle she began with.

This work tells us that Bella must open at least 5 drawers in order to retrieve the crown. It also tells us that the 5 drawers highlighted in the following diagram can be used to retrieve the crown in the order indicated. Therefore, 5 is the minimum number of drawers that Bella needs to open.

Connections to Computer Science

The ways in which each drawer can be opened can be modelled by a *directed graph*. We can represent each drawer as a *vertex* of the directed graph. Each *(directed) edge* will go from a drawer with a certain symbol to a drawer with the keyhole matching that symbol. In the diagram, the edges are *implicit*, since they are not actually part of the picture, but are rather inferred from the information available on each drawer.

The reason we say the graph is "directed" is because the opening of a drawer is only valid in one direction. For example, if drawer A contains a symbol that opens the keyhole of drawer B, we would have a directed edge from drawer A to drawer B, and it is not necessarily the case that the symbol in drawer B opens the keyhole for drawer A. In this case, we call B a *neighbour* of A.

In this problem, we want to find a *directed path* to the crown. One way to solve this is to perform a *breadth-first search*, where all of the neighbours of the crown are determined, and then repeatedly determine the neighbours of each of those neighbours, and so on.

The idea of searching for a path in a directed 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.

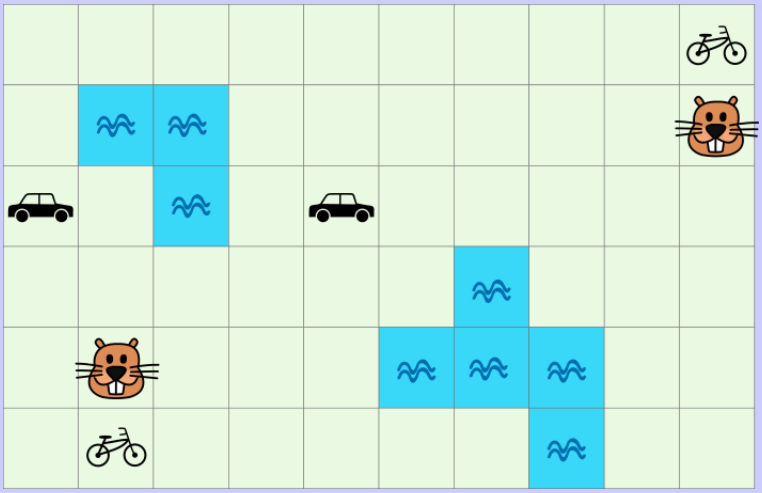
Country of Original Author

Iceland

Meet in the City

Story

Two beavers are planning on meeting somewhere in their city. Their current locations are shown on the following map of the city along with the locations of water, two bikes, and two cars.



The beavers can only move from one square on the map to another square that is horizontally or vertically adjacent to their square and does not contain water.

They can move 1 square in 1 minute when walking. However, if they reach a square with a bike or a car, then they can use it to travel faster. They can move 1 square in 30 seconds while on a bike, and they can move 1 square in 12 seconds while in a car.

Question

What is the least amount of time needed for the beavers to meet on the same square together?

(A) 3 minutes and 48 seconds

(B) 4 minutes

(C) 4 minutes and 12 seconds

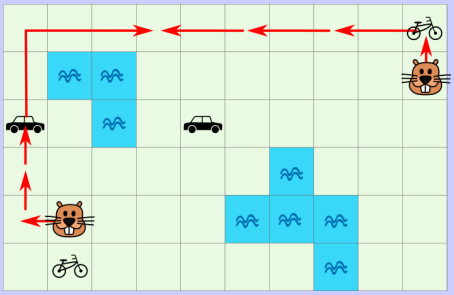
(D) 5 minutes

Answer

(B) 4 minutes

Explanation of Answer

If we convert all the time to minutes, we see that in 1 minute a beaver can move 1 square when walking, or 2 squares while on a bike, or 5 squares while in a car.



The beavers can arrive at the same square after 4 minutes if they take the routes as shown. Each individual arrow represents 1 minute of travel time.

To see why the beavers cannot arrive at the same square after fewer than 4 minutes, we determine all squares that one beaver could reach in 3 minutes and compare this to the squares that the other beaver could reach in 3 minutes.

The three diagrams given show the squares that the beavers can reach after 1 minute, 2 minutes, and 3 minutes.

After 1 minute, the only new squares that the beavers can reach are the squares adjacent to their starting locations. These squares are indicated in the first diagram. Notice that both beavers can reach a bike after 1 minute.

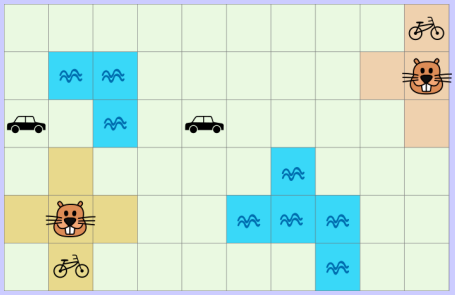
To find the new squares that can be reached after 2 minutes, we consider the squares that can be reached after 1 minute and determine what squares can be reached from these given 1 additional minute of travel. These new squares are indicated in the second diagram. If these squares can be reached while on a bike, then a bike symbol is indicated on the square.

This process is repeated to find the new squares that can be reached after 3 minutes. These squares are shown in the third diagram.

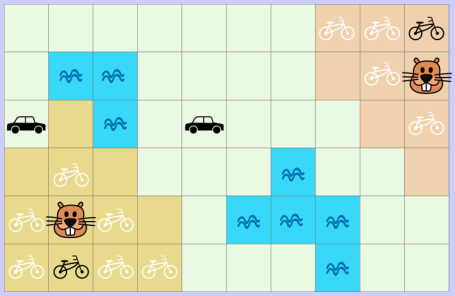
Notice that the beavers have no squares in common in this third and final diagram. This means they cannot possibly meet after only 3 minutes.

Note that this final diagram can also be used to find the 4 minute route shown earlier.

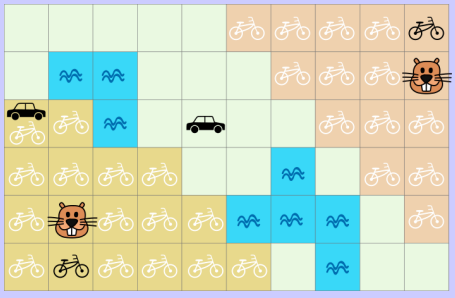
After 1 minute



After 2 minutes



After 3 minutes

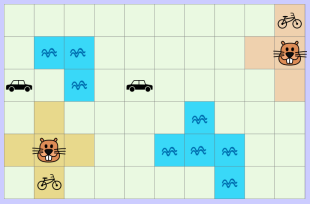


Connections to Computer Science

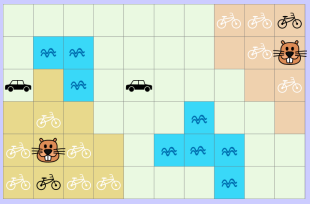
In order to solve this task, we can use a technique called *breadth first search*, and specifically, a variant called the *flood fill* algorithm.

Breadth first search is an *algorithm* or procedure that searches by looking at all immediate "next locations" based on the "current location". In this task, the next locations are the squares which are horizontally or vertically adjacent to the current location. This process repeats, with the most recently added "next locations" becoming the "current location". When observing the pictures below, the entire area is increasingly filled as if it were being flooded, and thus, the algorithm is known as *flood fill*.

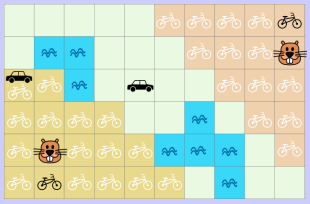
After one minute



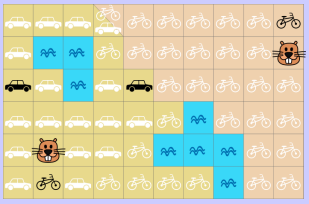
After two minutes



After three minutes



After four minutes



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.

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.

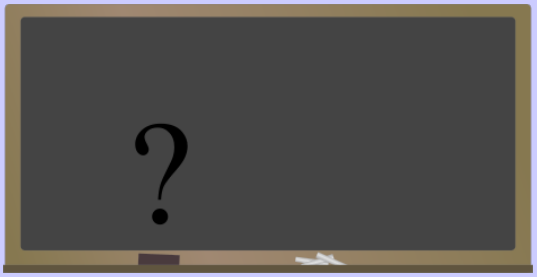
Country of Original Author

Lithuania

Missing Erasers

Story

Four students were helping their teacher clean up. While cleaning, one of the students hid the blackboard erasers. When the teacher realized that the erasers were missing, she asked the students, "Which one of you hid the erasers?" Each student answered as follows.



Amélie: "I didn't hide the erasers."

Benin: "Dahila didn't hide the erasers."

Cai: "Amélie hid the erasers."

Dahila: "Either Benin or Cai hid the erasers."

Only one of these answers was true.

Question

Which student hid the erasers?

(A) Amélie

(B) Benin

(C) Cai

(D) Dahila

Answer

(D) Dahila

Explanation of Answer

First note that Amélie's and Cai's answers can't both be false. This is because if they are, it would mean that Amélie both hid the erasers and didn't hide the erasers which is clearly impossible. Therefore, since only one of the students' answers is true and the other three are false, either Amélie or Cai told the truth.

From this we know that Benin's statement must be false which reveals that Dahila hid the erasers.

For completeness, we notice that this is consistent with Amélie's answer being true, Cai's answer being false, and Dahila's answer being false.

Connections to Computer Science

This is a *boolean logic* problem, since each statement is either *true* or *false*. In this task specifically, we are given a set of *statements* and asked to determine which one of four possible statements could be *true* while the other three statements are *false*.

To help solve this problem, we must use some *logical connectives* such as AND, OR, and NOT. Specifically, if a statement like "I didn't hide the erasers." is *negated* by writing NOT ("I didn't hide the erasers"), this can be rewritten as the equivalent statement "I did hide the erasers". Similarly, the statement "Either Benin or Cai hid the erasers." uses the logical connective OR, meaning that either one of Benin or Cai hid the erasers. We also use the boolean connective AND since one of the statements must be true AND each of the other three statements must be false.

Knowing how logical statements can be rewritten, combined, or simplified is a very useful skill in computer science in areas such as writing down the specification of a problem in a logical and thorough way or in writing computer programs that use logic in their decision making.

Country of Original Author

New Zealand

Shapes

Story

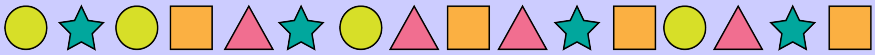
Here is a line of shapes.

The line has a run of stars of length 2. A *run* is an unbroken chain of identical shapes.

Ali likes to create long runs by changing shapes. For example, if Ali changes the middle square to a star in the line above, then he can create a longer run of length 4.

Question

Suppose Ali chooses and changes exactly 3 of the 16 shapes in the following line:



What is the length of the longest possible run that Ali can create?

(A) 4

(B) 5

(C) 6

(D) 7

Answer

(C) 6

Explanation of Answer

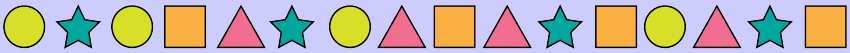
To show that Option C is correct, we need to prove two things:

(1) that a run of length 6 is possible, and

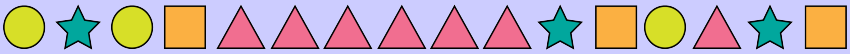
(2) that a run with length greater than 6 is not possible.

A run of length 6 can be made by changing the second star, third circle, and second square to triangles, as shown.

Before:

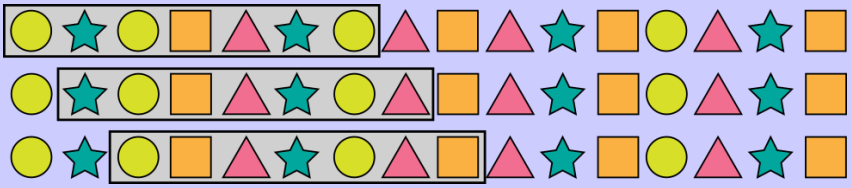


After:



To prove that a run with length greater than 6 is not possible, consider what it would mean if there was a longer run. It would mean there is a run of length 7. Remember that Ali changes only 3 shapes. Therefore, any run of length 7 must come from an unbroken chain of length 7 with 4 identical shapes in it.

There are ten unbroken chains of 7 shapes in the original sequence, the first few of which are shown below.



As we can see, none of these unbroken chains of length 7 have 4 identical shapes in them. Convince yourself that this is also true for the remaining unbroken chains of length 7. This means that it is not possible to have a run of length 7.

We have shown there are 6 shapes in the longest possible run.

Connections to Computer Science

This task is related to finding a *longest substring* that matches some given criteria. A *substring* is a sequence of adjacent characters, such as "ubs" within "substring".

One of the most studied applications of the longest substring in computer science is the *longest common substring* between two strings. Finding the longest common substring is used in *plagiarism detection* to determine if a document contains a significant amount of copied material. Another application of longest common substring is in *data deduplication*, where redundant copies of data (such as duplicate files on a disk drive) are found and removed.

Country of Original Author

Ukraine