



UNIVERSITY OF  
**WATERLOO**



The CENTRE for EDUCATION in  
MATHEMATICS and COMPUTING



2024  
*Beaver  
Computing  
Challenge  
(Grades 5 & 6)*





*Questions,  
Answers,  
Explanations,  
and  
Connections*

# Part A

# Pizza Party

## Story

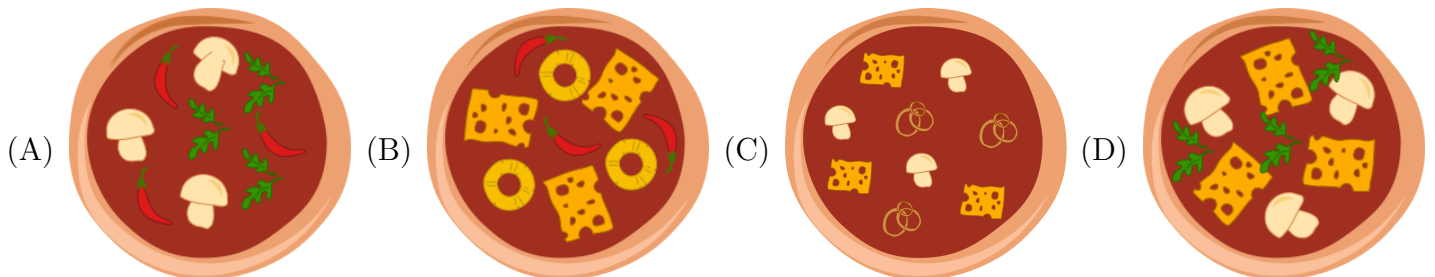
Jun is ordering a large pizza for four of his friends. Each of his friends has three favourite pizza toppings as shown below.

Friend	Favourite toppings
Eslam	
Meral	
Salma	
Wout	

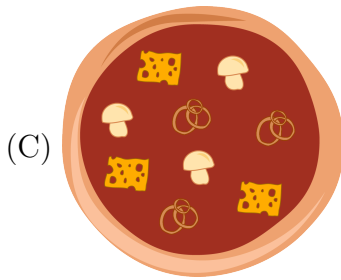
Jun wants to choose the three most popular toppings for the pizza he is ordering. (A topping is more popular than another topping if it is the favourite of more friends than the other topping.)

## Question

Which of the following pizzas should Jun choose?



## Answer



## Explanation of Answer

Looking at the favourite pizza toppings of each friend, we see that

- four people have mushrooms as a favourite topping,
- three people have cheese as a favourite topping, and
- two people have onions as a favourite topping.

The other three toppings (hot pepper, pineapple and spinach) are each the favourite of only one friend. This means that mushrooms, cheese, and onions are the three most popular toppings among Jun's friends. These are the three toppings in Option C.

## Connections to Computer Science

This problem highlights how to organize data in a way that makes retrieval easier. Specifically, if there was a way to ask a question such as “How many people like this topping?”, we could quickly determine what the most popular toppings were.

When there is a large amount of data to be stored and retrieved, often it is stored in a *database system*. A database stores some *key* value along with data associated with that key. In this particular problem, the name of the friend is the key, and the data associated with the key is their favourite toppings. Each entry can be thought of as a *row* in a *table* in a database.

To analyze the data, most database systems have the ability to *SELECT* various rows of data *WHERE* certain conditions are met, and even *COUNT* the number of such rows. For example, we could count the number of rows in our pizza table where cheese is one of the favourite toppings, and determine that there are three such rows.

Being able to organize data in a database, and perform *queries* to analyze the data contained in the database, are important skills for *database administrators*.



Country of Original Author




Germany

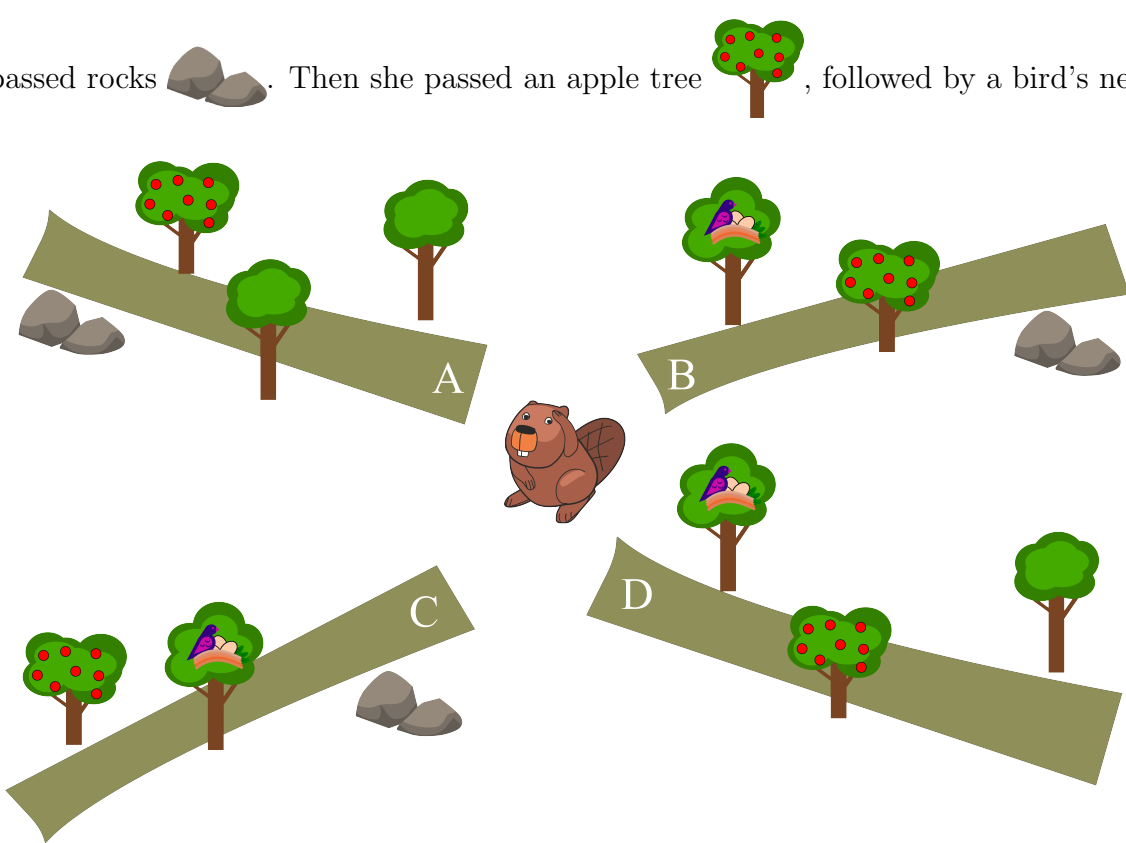


# Nature Hike

## Story

Beaver Alia hiked along a path from her home to the middle of the forest, enjoying nature along the way.

She first passed rocks . Then she passed an apple tree , followed by a bird's nest .



## Question

Which of the four paths leads back to Alia's home?

- (A) Path A
- (B) Path B
- (C) Path C
- (D) Path D

### Answer

(B) Path B

### Explanation of Answer

On the way into the forest, Alia first saw rocks, then an apple tree, and finally a bird's nest. If she takes the same path to get home, she will see the same objects, but in the reverse order. So, on the way home, she will first see a bird's nest, then an apple tree, and finally the rocks. Hence, Option B is the correct answer.

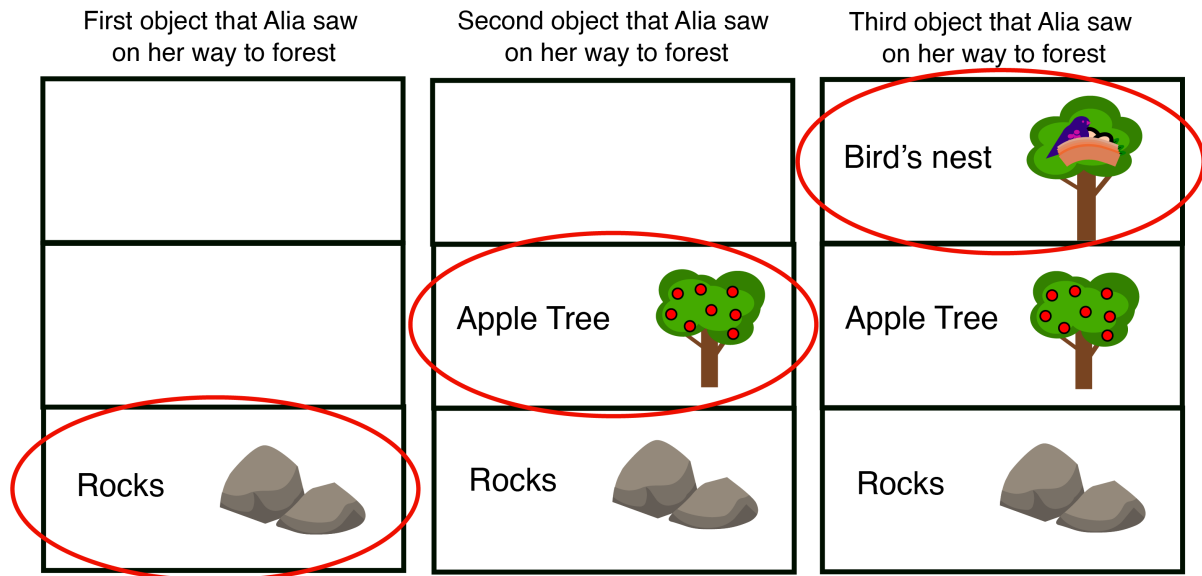
Option A is incorrect, because there is no bird's nest.

Option C is incorrect because the rocks should be seen last on the way home, not first.

Option D is incorrect because there are no rocks.

## Connections to Computer Science

This problem simulates a *stack data structure* that uses the *LIFO* (Last In First Out) rule: the item that is placed in last is the first to be removed. The objects seen by Alia can be thought of as objects put into a stack. The first object seen (rocks) is put into a stack, followed by the second object (apple tree) and the third object (bird's nest) so that the stack can be described as follows:



When Alia walks in the opposite direction, each object in the stack needs to be removed. The first thing Alia sees on her way home is a bird's nest. Then the bird's nest is removed from the stack so that in the top position is the apple tree. Then the apple tree is removed from the stack as well. The last item remaining in the stack is the rocks.

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.

## Country of Original Author

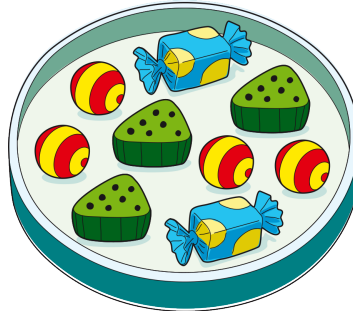
Indonesia





# Candies

## Story

Gabija offers to share the nine candies shown below with her friends.















They take some candy as follows:

- First, Benas takes one candy of each type.
- Then, Marija takes two green dotted candies .
- Finally, Andrius takes one round striped candy .

## Question

After this, how many candies of each type remain?

- (A) two , one , zero 
- (B) one , one , zero 
- (C) two , two , zero 
- (D) one , one , one 

### Answer

(A) two , one , zero 

### Explanation of Answer

After Benas takes one candy of each type, the number of candies is as follows:

three , one , two 

Then, after Marija takes two green dotted candies, the number of candies is as follows:

three , one , zero 

Finally, after Andrius takes one round striped candy, the number of candies is as listed in Option A.

### Connections to Computer Science

This task is focused on keeping track of a set of *variables*. We can think of the number of each type of candy as being stored in a variable that describes the candy being stored. For example, the variable **rs** could store the number of round striped candies, the variable **gd** could store the number of green dotted candies, and the variable **sb** could store the number of square blue candies.

Initially, we know that  $rs=4$ ,  $gd=3$  and  $sb=2$ . After each person takes their selection of candies, the values of some of the variables change. For example, if Benas takes one candy of each type, the variables would have values  $rs=3$ ,  $gd=2$  and  $sb=1$ .

Variables are used extensively in many *programming languages* to store information that computers can read, process, and display.

### Country of Original Author

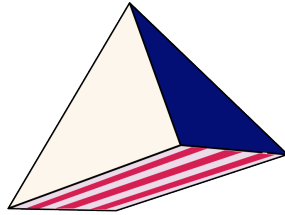
Lithuania



# Gift Box

## Story

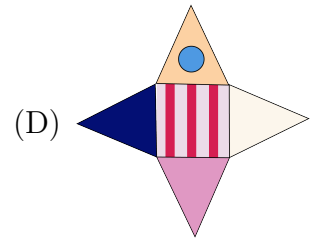
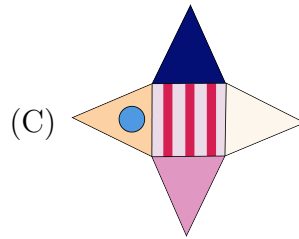
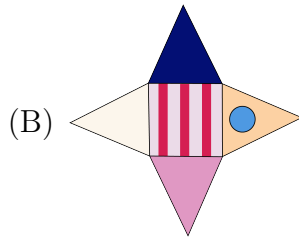
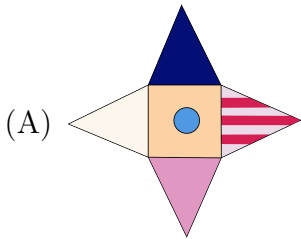
When folded, a gift box is a pyramid with a square base as shown.



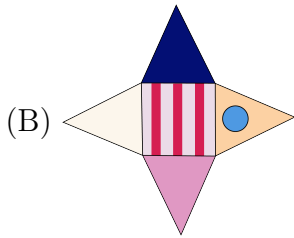
The sides and bottom of the box are different colours and patterns, but only some of these are shown.

## Question

Which of the following could be the outside of the gift box when it is unfolded?



## Answer



## Explanation of Answer

All four options have a square in the centre with a triangle on each of its four sides, which would fold up to make a pyramid with a square base. Since they are all the correct shape, we need to use the colours and patterns to determine which option is correct.

The square base of the folded gift box is striped, so when it is unfolded, the square in the centre will be striped. Two of the triangular sides of the folded gift box are shown. One of the triangular sides appears to be white and the other appears to be dark blue. Looking from the base, the dark blue side is beside the white side in the clockwise direction. The other two sides are not shown on the folded box so they can be any colour and/or pattern.

Option A is incorrect because its square is not striped.

Option C is incorrect because the dark blue triangle and white triangle are in the wrong order. The dark blue triangle should be the next triangle clockwise from the white triangle, but in Option C the dark blue triangle is the next triangle counter-clockwise from the white triangle.

Option D is incorrect because when it is folded, the dark blue triangle and white triangle would be opposite each other instead of beside each other.

Option B is correct because the square in the centre is striped like the base on the folded gift box. Also the dark blue triangle is the next triangle clockwise from the white triangle.



### Connections to Computer Science

To solve this task, we must identify the properties of the parts of the gift package that remain the same when the gift package is unfolded and possibly rotated. Because of these various arrangements, it is crucial to consider the positions of the parts of the gift package relative to each other. For example, the blue part's left-hand neighbour is the light part when looking from below the gift package.

This use of *visual pattern matching* is also used in *computer vision*, to detect the location and position of objects based on characteristics of those objects. One example of visual pattern matching is in the design of robots to locate and move physical objects in a factory. Another application is assisting medical doctors in the detection of tumours from CT scans of patients. Both of these applications use partial information to make *inferences* about the most likely location/orientation/size of objects in visual images.

### Country of Original Author

Estonia

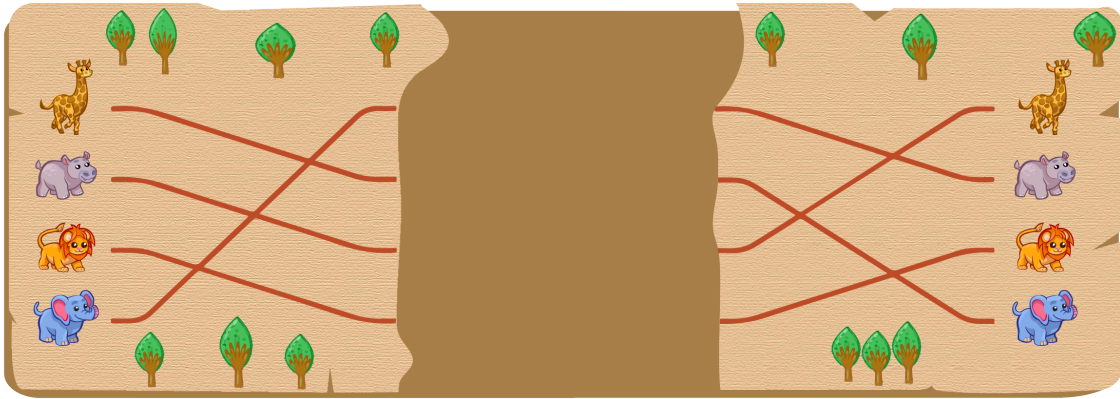


## Part B

# Map It Out

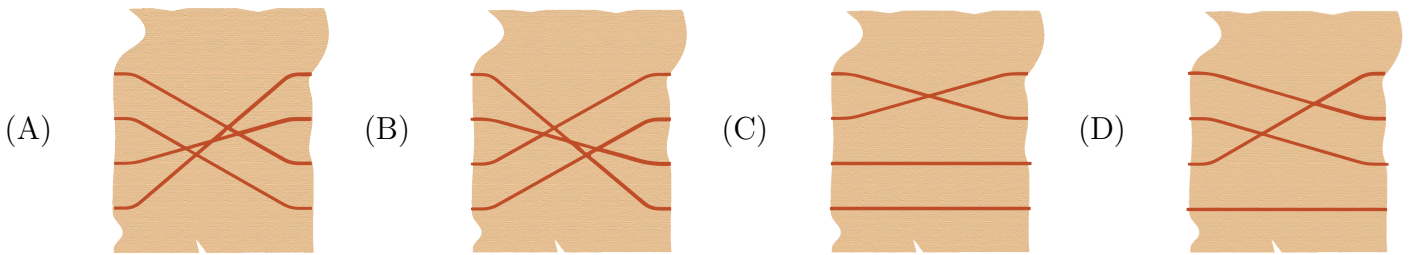
## Story

Four animals travelled across a plain. The following map shows their routes with a path from where each animal started to where they finished. However, the middle section of the map is missing.



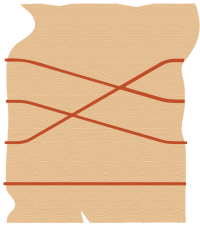
## Question

Which of these pieces could be the missing middle section of the map?



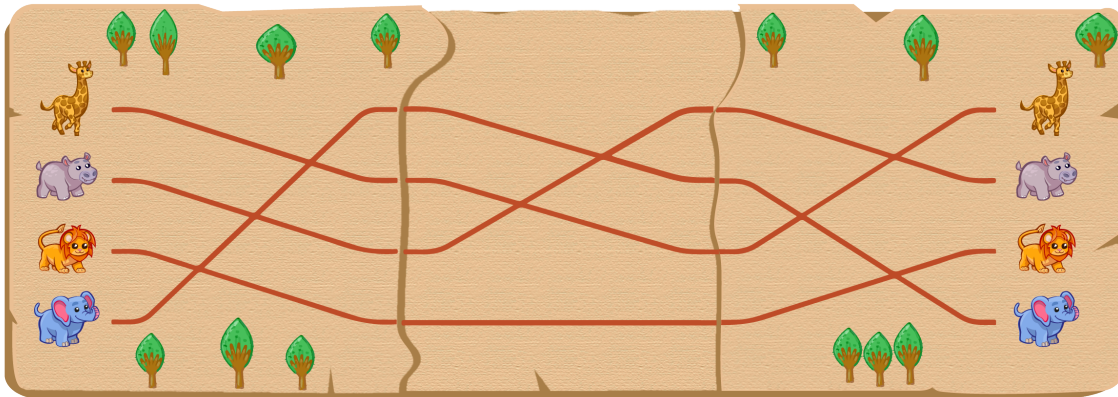
Answer

(D)

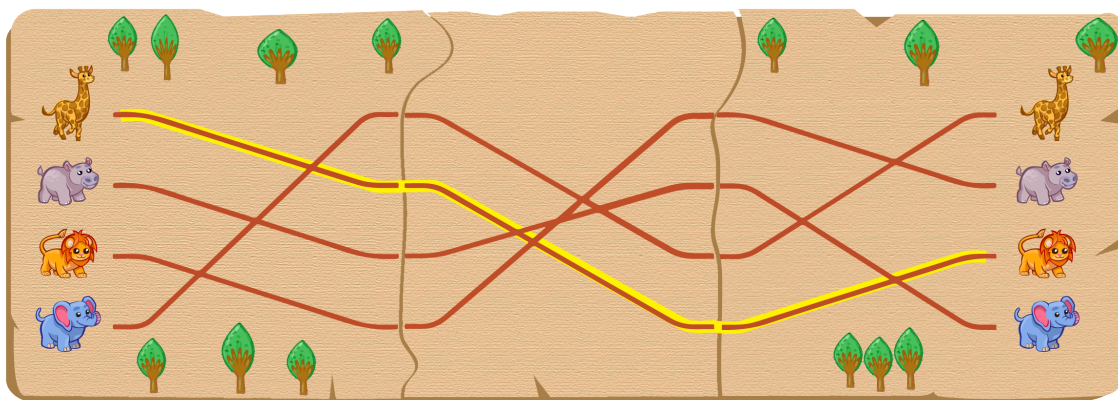


Explanation of Answer

When we fill in the missing middle section with Option D, each path connects an animal's starting point with its finishing point, as shown. Thus, Option D is correct.



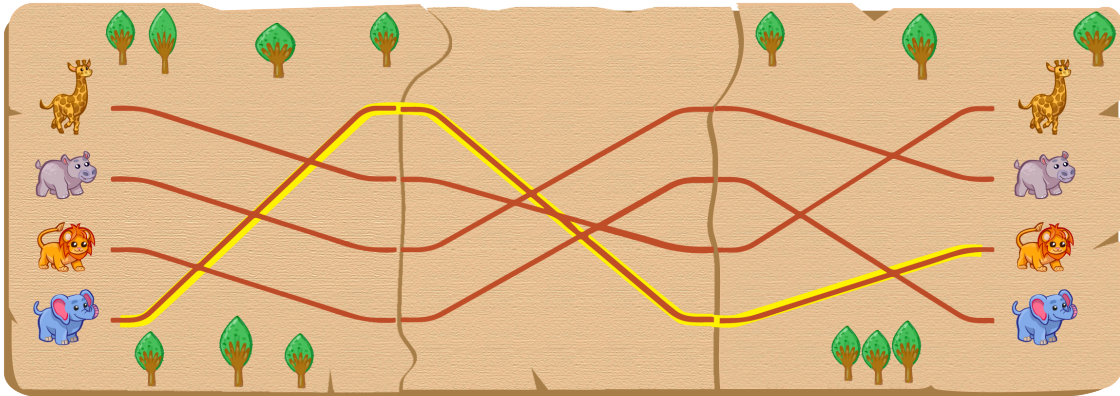
When we fill in the missing middle section with Option A, the path starting at the giraffe finishes at the lion, as shown. Thus, Option A is incorrect.



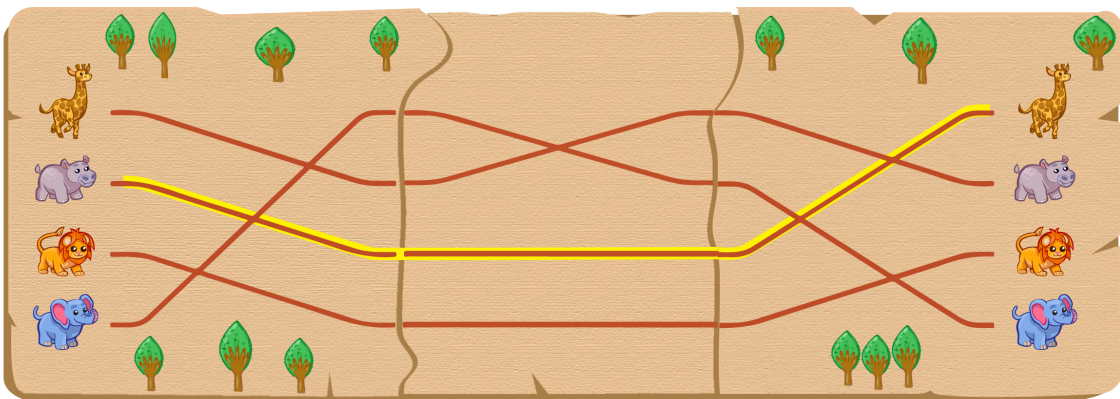


### Explanation of Answer Continued

When we fill in the missing middle section with Option B, the path starting at the elephant finishes at the lion, as shown. Thus, Option B is incorrect.



When we fill in the missing middle section with Option C, the path starting at the hippo finishes at the giraffe, as shown. Thus, Option C is incorrect.



## Connections to Computer Science

This task can be thought of as modelling a *function* as a *black box*, with determining the “contents” of the black box as the main goal of the task.

A function can be thought of as a box that consumes some *input* and produces some *output*. For example, we can think of “add” as a function that takes two numbers as input and produces a number, which is the sum of those two numbers, as the output. Computer scientists think of many functions as “black boxes”: *what* the function does is more important than *how* the function operates. Ignoring “how does it work?” and focussing on “what does it do?” is the fundamental computational concept of *abstraction*: the details of the function can be ignored to make the understanding or reasoning about the function easier.

For example, when a person drives a car, they *abstract* away the *low-level* details of the axel turning, spark plugs firing to move a piston, etc. Instead, the driver focusses on the gas pedal, brake pedal, and steering wheel. The complex details of how an engine works are ignored in order to make driving easier to understand.

Determining the right level of abstraction, where there is just enough but not too much detail, is a very important aspect of modelling problems to be solved on a computer.

To make this task easier to approach, the details of where the animals are at the right-most side of the left section, and the left-most side of the right section can be used to determine the function. That is, knowing that the left section ends with the animals in the order elephant, giraffe, hippo, and lion (from top to bottom), and the right section starts with the animals in the order hippo, elephant, giraffe, and lion (from top to bottom) helps focus on the “what” needs to happen in the middle piece, and then testing each middle for the “how” it happens becomes the goal of the task.

## Country of Original Author

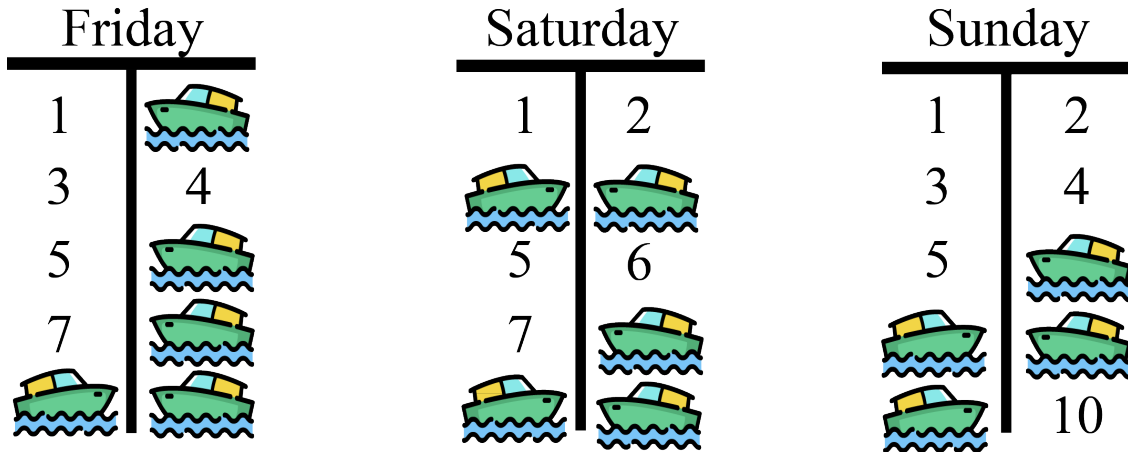
India



## Boat Parking

### Story

The boat slips at a marina are either reserved or available as shown. For example, boat slip 2 is reserved on Friday but available on Saturday.



Tom will be parking his sailboat at the marina for two consecutive days. He will choose to arrive on either Friday or Saturday and he needs to book one boat slip that is available for both the day he arrives and the next day. He has several options available. For example, one option is for Tom to arrive on Saturday and book boat slip 1.

### Question

How many options does Tom have in total?

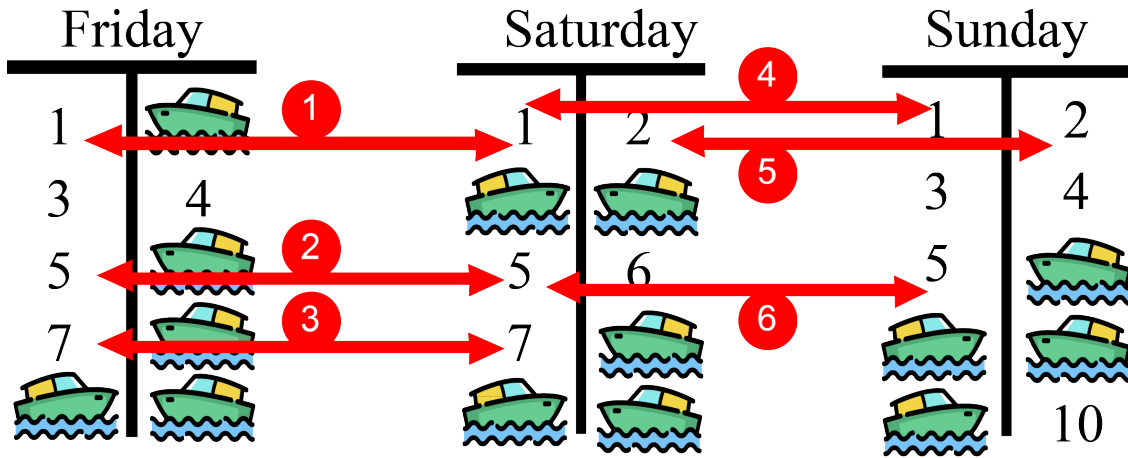
- (A) 5
- (B) 6
- (C) 7
- (D) 8

Answer

(B) 6

Explanation of Answer

If Tom arrives on Friday, he can only book boat slip 1, 5, or 7. If Tom arrives on Saturday, he can only book boat slip 1, 2, or 5. Thus, there are six options in total. These are illustrated below.





## Connections to Computer Science

All data that a computer uses or stores is actually a sequence of zeros and ones. Each zero or one is called a *bit*, which is short for *Binary digIT*. A sequence of bits is called a *binary code*, *binary representation*, or *binary number*.

In this task, if we denote an empty space as 0 and a reserved space as 1, then each boat slip corresponds to one bit. We can get a sequence of bits by viewing the boat slips on each day. In total, each day will correspond to a binary code of length 10 since there are 10 boat slips available. For example, Friday corresponds to 0100010111, Saturday corresponds to 0011000111 and Sunday corresponds to 0000011110.

This task asks us to take the sequence for two days in a row, and looking for bits (boat slips) that are both 0. This corresponds to the *binary logical operator* called *NOR*, which is short for *NOT OR*. The NOR operator takes two bits, and produces the bit 1 if both bits are 0, and otherwise produces a 0. That is, the NOR operator will produce 1 for each boat slip that is empty on both days. Specifically:

$$\begin{array}{r} \phantom{\text{NOR}} \phantom{0011000111} \phantom{(\text{SAT})} \\ \text{NOR} \phantom{0011000111} \phantom{(\text{SAT})} \\ \hline 1000101000 \end{array} \qquad \begin{array}{r} \phantom{\text{NOR}} \phantom{0000011110} \phantom{(\text{SUN})} \\ \text{NOR} \phantom{0000011110} \phantom{(\text{SUN})} \\ \hline 1100100000 \end{array}$$

Then, notice that the total number of 1s between these two results is the correct answer to the task.

The NOR operator is used in *computer hardware*, such as the *central processing unit (CPU)*, along with other *logic gates* such as AND, NOT, and OR to help process binary data.


## Country of Original Author

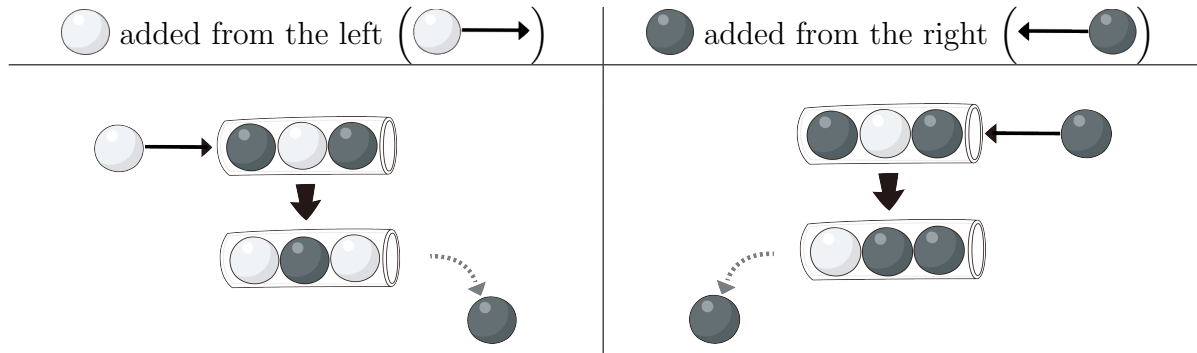
Malta



# Pushed Out

## Story

Minji has a tube  that is open at both ends and can hold a maximum of three balls. Balls can be added to the tube from either end. If the tube is already filled with three balls and Minji adds a fourth ball from one end, the ball closest to the other end will be pushed out, and the remaining two will shift, as shown below.







## Question

The tube is initially filled with the following three balls.



What does the tube look like after Minji performs the four actions shown below in order from left to right?




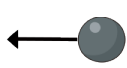

- (A) 
- (B) 
- (C) 
- (D) 




### Answer




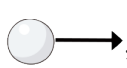

### Explanation of Answer

To solve this problem, we can simulate putting each of the four balls in the tube.

At first our tube looks like . After performing , it becomes .

From , we perform , and it becomes .

From , we perform , and it becomes .

From , we perform , and it becomes , which is Option D.

### Connections to Computer Science

In this task, the tube is similar to a *deque* (*double-ended queue*), a *data structure* that allows data to be placed or retrieved from both ends.

This task also demonstrates that storage, or *memory* in a computer system, is fixed. Specifically, the tube in the task holds at most three balls at a time. Similarly to how a computer works, if additional information is to be stored in a limited memory, some information needs to be moved elsewhere.

One common use of a deque is to store web browser history: there are “back” and “forward” buttons in most web browsers that behave in a similar way to the ends of the tube. Other applications of deques include scheduling print jobs and verifying the validity of math expressions such as  $((1+2) \times (3+4))$ .

Another application of deques is in *parallel computing*. Specifically, there is an *algorithm* called *work stealing* that allows multiple *processors* to work as efficiently as possible. The key idea of the algorithm is that processor B can remove the “last” element from a list of jobs for processor A, in order to not interfere with the “first” job that processor A may be executing.

### Country of Original Author

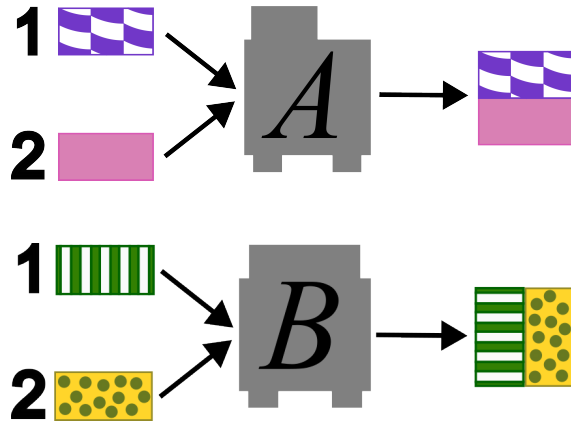
South Korea



# Sewing Machines

## Story

Two types of sewing machines can each be fed two pieces of fabric to make a larger piece of fabric. They work separately as shown.



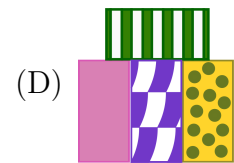
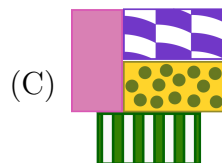
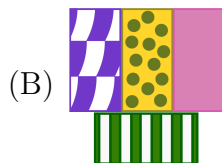
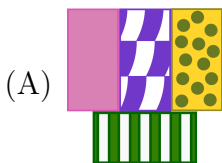
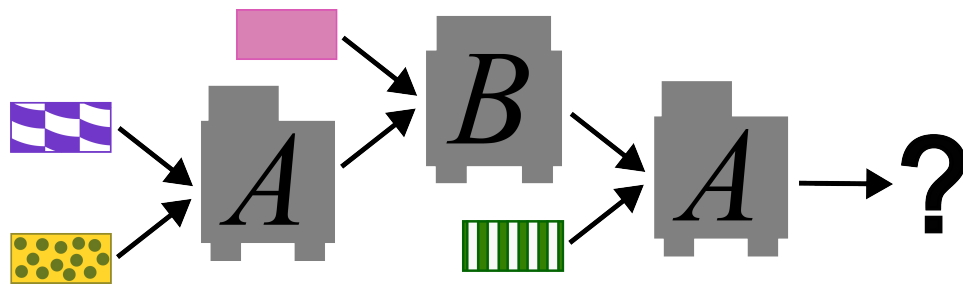
Machine *A* sews piece 1 centered on top of piece 2.

Machine *B* turns piece 1 and piece 2 counter-clockwise by 90 degrees and then sews piece 1 centered to the left of piece 2.

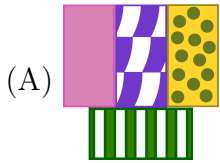
The two machines can be combined to form even larger pieces of fabric.

## Question

Pieces are fed through machines as shown by the arrows in the diagram below. What does the final large piece made by the three combined machines look like?

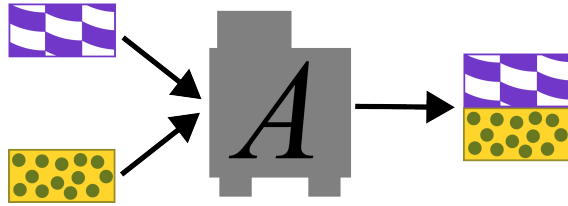


Answer

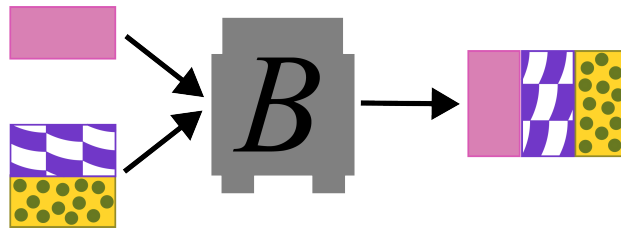


Explanation of Answer

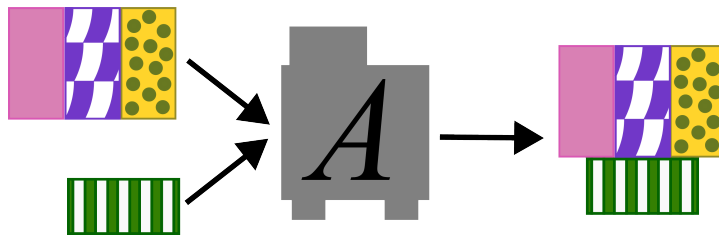
First, machine *A* makes a new larger piece as shown.



Then, that piece and another new piece is fed into machine *B* which makes an even larger piece of fabric as shown.



Finally, that piece is put into machine *A* with yet another new piece which makes the piece of fabric shown in Option A.



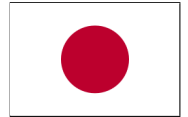
### Connections to Computer Science

*Functions* are often used in a *computer program* to divide up the program into smaller components, and also to use these components in possibly different ways, without having to repeat the description of those same steps.

A function requires some *input* values, and will produce some *output* value. As demonstrated in this task, the output of one function can be used as an input to another function. This is one way to *modularize* a larger program: for a larger project, each team member can work on one particular function, that will all combine to implement the larger project. For example, in a program, there may be a function that takes in a list of numbers as input, and produces the sum of those numbers. This function could be called many times in a program, with different lists of numbers, instead of repeating the code to sum a list of numbers.

### Country of Original Author

Japan



## Part C

## Online Class

### Story

Nine students are sitting side by side in one row in the library while their teacher conducts an online lesson from her home. The teacher sees the class from her laptop screen as shown.



Each student is using a different computer, but the teacher's screen shows who each student is sitting next to.

### Question

Which student is sitting in the middle (5th position) of the row in the library?

- (A) Raul
- (B) Lee
- (C) Busara
- (D) Hannah



### Answer

(D) Hannah

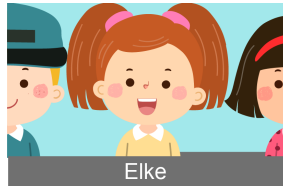
### Explanation of Answer

Based on the what can be seen on the teacher's screen, we can determine who each student sits next to.

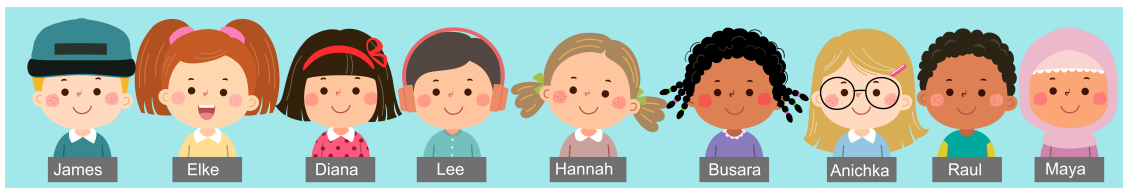
To begin, we can determine that James is sitting on an end of the row because only one person is right beside him:



Then, we can determine that Elke is sitting between James and Diana so she must be the student next to James:



At this point, we know the row begins with James, then Elke and then Diana. Using this approach with the rest of the views, we can determine the order of all nine students in the row:



The student sitting in the middle of the row is the one at the 5th position from left, or the 5th position from right. That person is Hannah.

### Connections to Computer Science

The actual student arrangement in a row can be modelled as a *data structure* called a *doubly linked list*. In a doubly linked list, each *node* of the list contains the element value, a reference to the previous node, and a reference to the next node. In this task, each screen (node) shows the child (element of the list) and references for the child on the right and on the left of the child in the center of the screen. This task includes an exercise in traversing a list, starting from either end.

Additionally, this task focusses on *visual pattern matching*, which is important for *computer vision*. In particular, the problem of *edge detection* is highlighted in this task: for every student that is on the edge of a screen, we need to determine where those same features continue on the edge of another screen. These are important problems to be solved in *autonomous driving*: it is crucial to determine if there are pedestrians or other objects on the road, since the edge of the road and the dividing lines on the road form the edges that need to be maintained.

### Country of Original Author

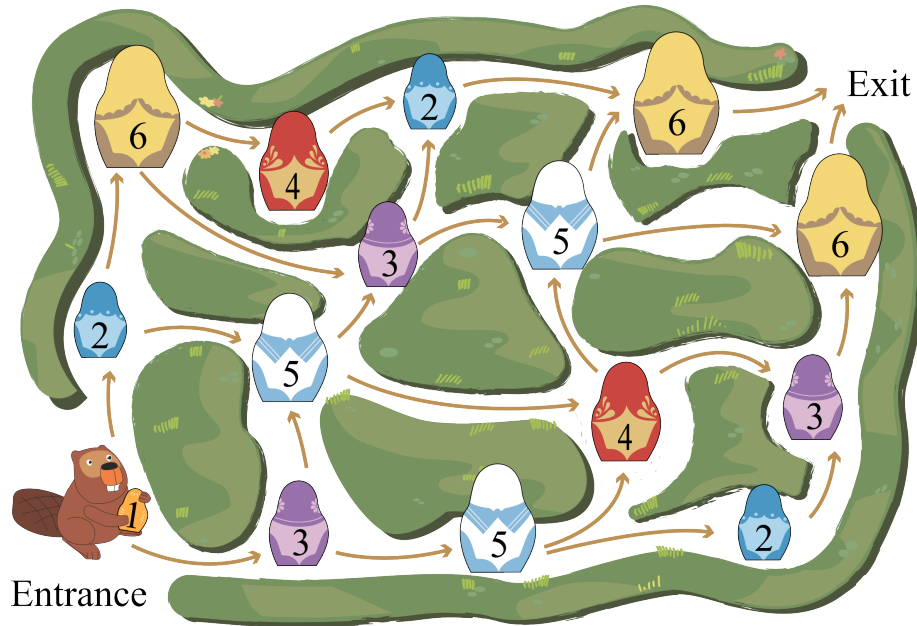
Malaysia



## Collecting Dolls

### Story

Beaver Deana enters the maze below carrying a doll of size 1. She then goes through the maze and collects dolls of different sizes, placing smaller dolls inside larger dolls.



Deana follows the arrows and obeys the following rule whenever she encounters a doll.

- If the doll she encounters is bigger than the biggest doll she already has, she can choose to either take the doll and put her dolls inside of it, or leave it behind.
- Otherwise, if the doll she encounters is the same size or smaller than the biggest doll she already has, she must leave the doll behind.

### Question

What is the maximum number of dolls that Deana can collect, including the size 1 doll, by the time she reaches the exit of the maze?

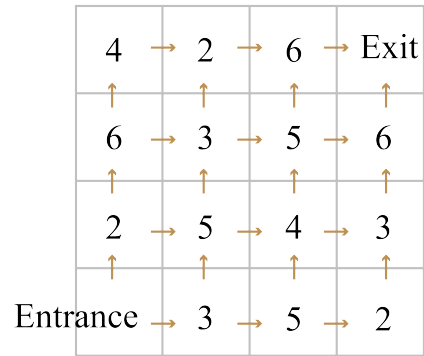
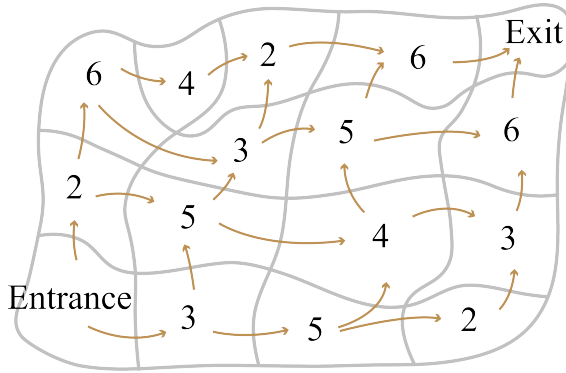
- (A) 3
- (B) 4
- (C) 5
- (D) 6

**Answer**

(C) 5

**Explanation of Answer**

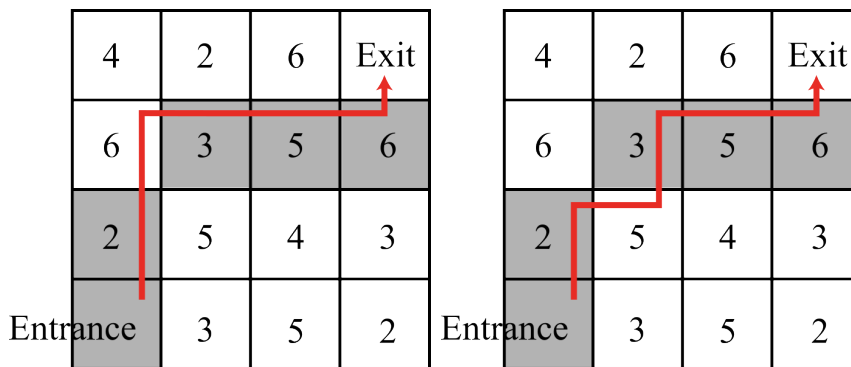
We can draw a simplified map of the maze and then an even simpler map upon noticing that all arrows move either rightwards or upwards.



All paths from the Entrance to the Exit are 6 steps long so the maximum number of dolls Deana can carry is 6. By the rule and because the biggest doll is of size 6, if she collects 6 dolls, they must be of sizes 1, 2, 3, 4, 5, and 6, and picked up in that sequence.

Notice that the only one way to pick up a doll of size 3 and also a doll of size 4 is to begin by moving right to pick up the doll of size 3, then moving either right again or up and choosing to leave the doll of size 5 behind, and then choosing to pick up the doll of size 4. In this case, Deana cannot pick up and carry a doll of size 2. This means the answer is at most 5.

Here are two ways Deana can carry five dolls out of the maze. The dolls she takes are indicated in grey.



## Connections to Computer Science

This task asks us to find the longest increasing sequence of doll sizes across all paths from the entry to the exit. For a single path, the *longest increasing subsequence problem* is a standard algorithmic problem. The problem statement is that given a list of numbers, pick numbers that form the longest sequence that is constantly increasing. For example, the sequence 4, 10, 3, 3, 20 would have the subsequence 4, 10, 20 as the longest increasing subsequence. There is an efficient algorithm that uses *dynamic programming* to remember partial solutions of shorter subsequences without having to recompute them.

Searching for the longest such sequence across multiple paths in a two-dimensional grid is a more complex problem. In the maze in this task, the paths have a simple structure (all of equal length, going only up and right) and our simple analysis was sufficient to find the answer.

For more complicated mazes, a more general approach is needed to systematically explore all possible paths. If the maze is small enough, the process of *backtracking* can be used to examine all possible paths. However, when playing a game like chess or Go, where there are a huge number of possible moves, it is not feasible to explore all paths. So we try to identify moves that will not be useful in reaching a good outcome and we stop exploring such paths as we go along, using various *heuristics*.




## Country of Original Author

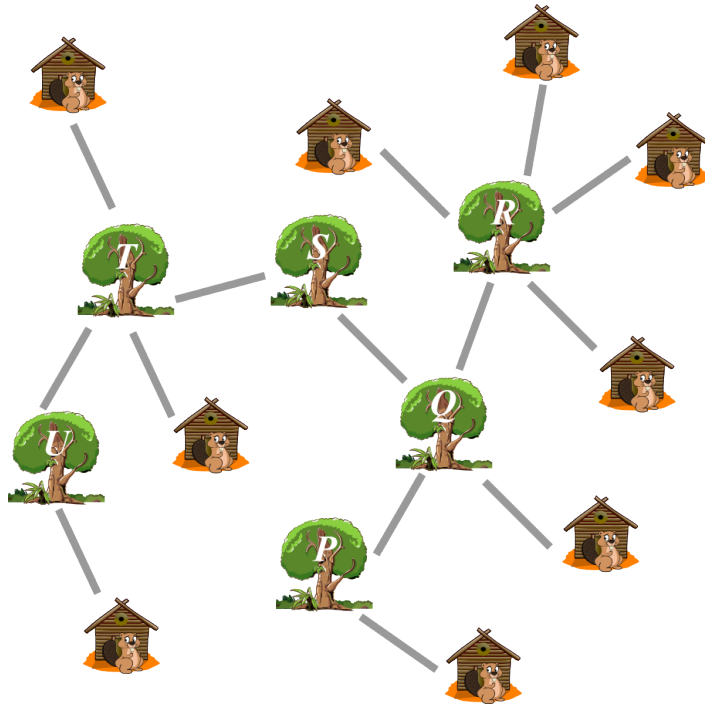
Taiwan



# Meeting Point

## Story

In an area containing trees , beavers have constructed lodges . The beavers have also constructed paths  that connect their lodges to trees, and some trees to one another.



The beavers now want to hold a group meeting at one of the trees, but they don't want to travel more than necessary. They want to choose a tree so that each beaver can travel to that tree using only one, two or three paths.

## Question

Where should the beavers hold their group meeting?

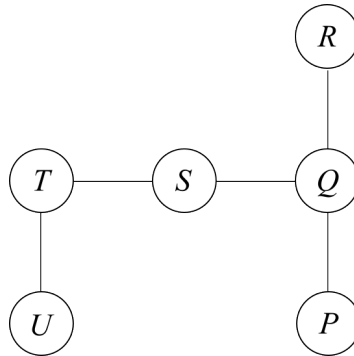
- (A) Tree *Q*
- (B) Tree *R*
- (C) Tree *S*
- (D) Tree *T*

**Answer**

(C) Tree  $S$

**Explanation of Answer**

Regardless of the meeting's location, each beaver must initially use one path to reach the tree connected to its lodge. This allows us to disregard the lodges and focus solely on the trees. We can create a simplified diagram as shown below, where each tree is depicted as a labeled circle.



Now we can see that from any tree, a beaver can get to Tree  $S$  using only one or two paths. This means they can get to Tree  $S$  from their lodge using only one, two, or three paths in total.

We can also see that choosing any other tree as the meeting point will require beavers to use more than three paths to get to the meeting point. One way to see this is to notice that either the beaver who travels from their lodge to get to  $U$  or the beaver who travels from their lodge to get to  $R$  (or  $P$ ), must use at least three additional paths to get to the meeting point.

## Connections to Computer Science

A *tree* is a data structure that represents many real-world situations. A tree is a special form of a *graph*: a graph is a collection of *nodes* along with *edges* that connect nodes together. A tree is a graph where there are no *cycles*: that is, there is no *path* that starts and ends at the same node.

In computer science, a tree is a widely used abstract data type that represents a *hierarchical structure* with a set of connected nodes. Each node in the tree can be connected to many children (depending on the type of tree), but must be connected to exactly one parent, except for the root node, which has no parent (i.e., the root node as the top-most node in the tree hierarchy).

This task focusses on finding the *center* of a tree. The center of a tree is a node for which the distance from the farthest node is as small as possible.

The algorithm of finding a center of a tree is an example of a *tree pruning* technique which can be applied to many situations involving trees and graphs in general. Pruning is a data compression technique in *machine learning* and *search algorithms* that reduces the size of trees by removing non-critical and redundant sections.

## Country of Original Author

Poland

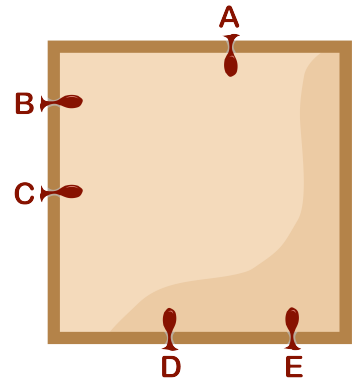




# Balloon Machine

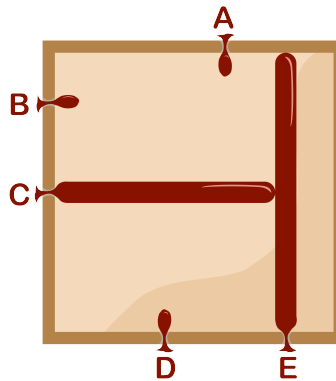
## Story

A machine creates different designs by inflating long, skinny balloons attached to the edges of a square frame. There are five balloons labelled  $A$ ,  $B$ ,  $C$ ,  $D$ , and  $E$ , arranged around the frame as shown.



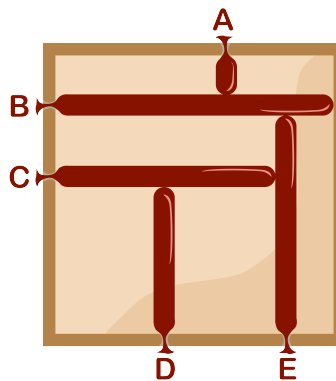
All balloons start deflated. To create a design, the machine reads a sequence of letters, from left to right, and inflates the corresponding balloons in that order. When a particular balloon is inflated, it will continue to extend until it reaches either another balloon or the opposite edge of the frame.

For example, starting with all balloons deflated, if the machine reads the sequence  $E C$ , then it will create the following design.



## Question

Starting with all balloons deflated, the machine reads a sequence consisting of the five letters  $A$ ,  $B$ ,  $C$ ,  $D$ , and  $E$  in some order and creates the design shown. How many of the balloons could have been the third balloon inflated?



(A) 1

(B) 2

(C) 3

(D) 4

### Answer

(C) 3

### Explanation of Answer

Since the design is created by inflating each balloon exactly once, in some order, we can determine the following about balloons  $B$  through  $D$  from the final design.

- Balloon  $B$  must have been inflated before balloon  $E$ , since balloon  $E$  stops at balloon  $B$ .
- Balloon  $E$  must have been inflated before balloon  $C$ , since balloon  $C$  stops at balloon  $E$ .
- Balloon  $C$  must have been inflated before balloon  $D$ , since balloon  $D$  stops at balloon  $C$ .

This means that balloons  $B$  through  $D$  must have been inflated in the following order:  $B E C D$ .

We can also see from the design that balloon  $A$  must have been inflated after balloon  $B$ , since balloon  $A$  stops at balloon  $B$ . Thus, balloon  $B$  must have been inflated first. After this, balloon  $A$  could have been inflated at any time in order to produce the desired final design. This means there are four different possibilities for the sequence, corresponding to the four different ways to place  $A$  in the sequence  $B E C D$  after  $B$ . These are as follows:

- $B A E C D$
- $B E A C D$
- $B E C A D$
- $B E C D A$

Thus, only balloons  $E$ ,  $A$ , or  $C$  could have been inflated third.

### Connections to Computer Science

The sequence of letters is an example of a *computer program* controlling a machine. Each letter is a *statement* that causes the balloon machine to inflate a balloon. As in every computer program, the order of statements is essential. For example, the sequence  $B E$  makes the machine create a different image than the sequence  $E B$ .

This task also relies on *logical inference*, which is an important skill in computational thinking. Specifically, reasoning about the order of operations, and which ones can or cannot come before others, is fundamental in designing programs that are both *efficient* and free of *logical errors*.

Country of Original Author

Germany

