



UNIVERSITY OF  
**WATERLOO**



The CENTRE for EDUCATION in  
MATHEMATICS and COMPUTING



2020  
*Beaver  
Computing  
Challenge  
(Grade 5 & 6)*

*Questions,  
Answers,  
Explanations,  
and  
Connections*

# Part A

## Bear Selection

### Story

Ren is allowed to bring one of his four teddy bears to school.



Ren brings the bear that has a star on one of its feet, and is wearing a scarf or a bow, but not glasses.

### Question

Which bear did Ren bring to school?



## Answer

(D)



## Explanation of Answer

Ren did not choose the bear in Option A or the bear in Option C because those bears have glasses.

Ren did not choose the bear in Option B because that bear does not have a star on one of its feet.

Ren chose the bear in Option D. The bear in Option D has a star on one of its feet, is wearing a scarf, and does not have glasses.

## Connections to Computer Science

Ren decides that the bear he will take along to school must meet a specific set of requirements. This set of requirements is:

- the bear has a star on one of its feet
- the bear is wearing a scarf or a bow
- the bear does not have glasses

This task requires the use of *boolean logic*. Each of Ren's requirements is expressed as a *boolean statement*. A boolean statement is either *true* or *false*. The three individual statements are then combined together using the *boolean operators* AND, OR, and NOT. AND is true if all statements it connects are true. OR is true if at least one of the statements it connects is true. NOT is true if the statement it is referring to is false. We can think of Ren's set of requirements as the following *boolean expression* which combines the three statements above:

(star on one of its feet) AND (wearing a scarf OR wearing a bow) AND (NOT has glasses)

Boolean logic is used extensively in programming languages, in extracting information from databases or spreadsheets, and in the CPU itself.

## Country of Original Author

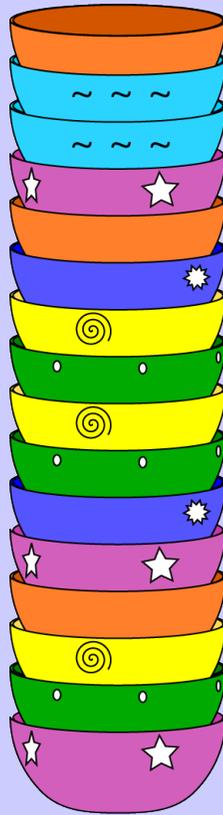
Canada



## Bowls

### Story

Whenever a customer orders soup, a bowl is taken from the top of the stack shown.



### Question

What is the fewest number of soup orders that need to be filled so that three identical bowls are used?

- (A) 13
- (B) 14
- (C) 15
- (D) 16

### Answer

(A) 13

### Explanation of Answer

Starting at the top of the stack and moving down, the first time three identical bowls will be used is when we take the third orange bowl. This bowl is the 13th bowl from the top of the stack, so that means at least 13 soup orders must be filled so that three identical bowls are used.

### Connections to Computer Science

In this problem, you are asked to keep track of which bowls are removed until the required number of identical bowls are removed. The key property is that the top bowl must be the first one that is removed. We say that this follows the *last-in first-out* or *LIFO* principle.

In computer science, a collection of items that follows this principle is called a *stack*. Stacks have two main operations: *push* places an element at the top of the stack, and *pop* removes the top element from the stack. In this problem, we are only using the pop operation.

A stack is one fundamental way in which a collection can be stored. In particular, a *computer program* will often be broken into smaller pieces called *subroutines*. The order in which these subroutines are executed is often determined using a stack. An example of an application of stacks that you might be more familiar with is an “undo button” in an editor or word processor. Can you see how this follows the LIFO principle?

### Country of Original Author

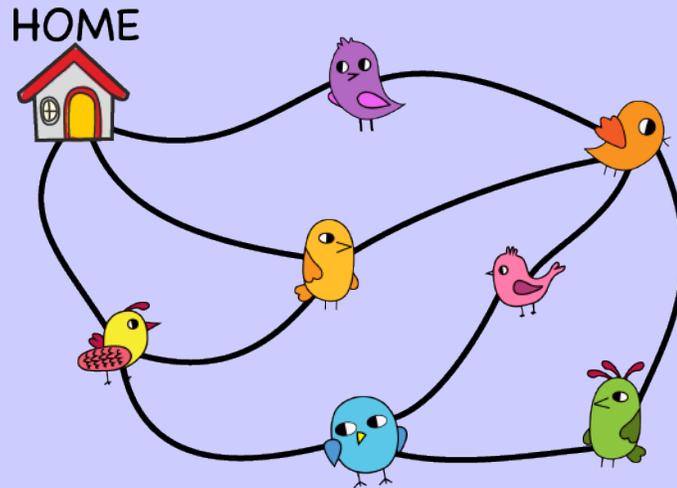
Ireland



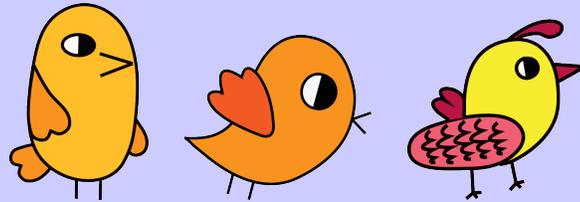
## Bird Watching

### Story

A family went for a walk. They started from their home and walked along some paths, eventually returning home. They did not walk on any path more than once.

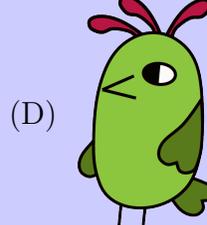
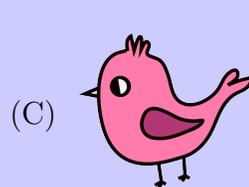
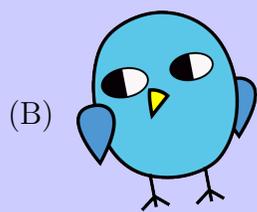
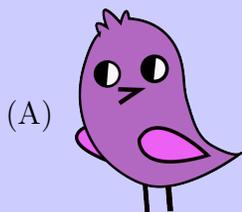


During their walk they saw *exactly* four birds. Three of the four birds they saw are shown below:

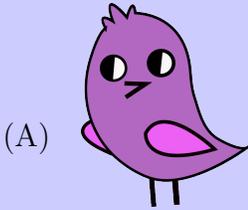


### Question

Which other bird must they have seen?

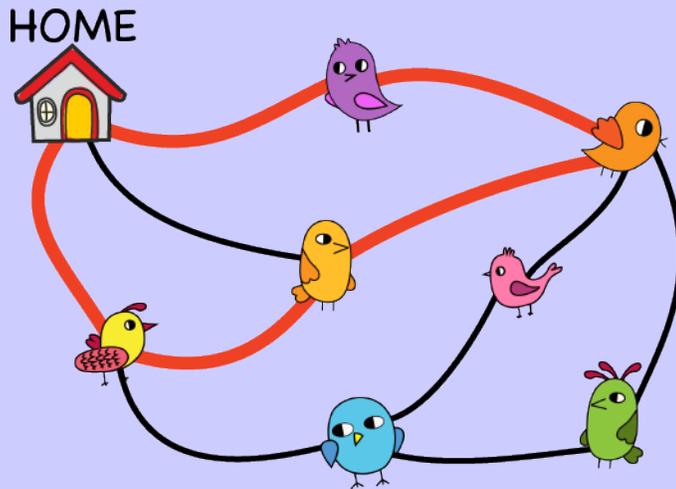


Answer

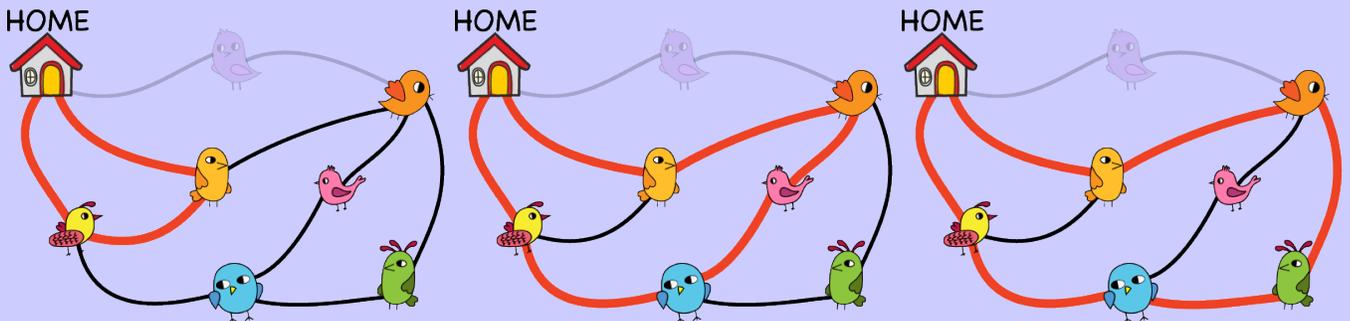


Explanation of Answer

The map below shows a route the family could have taken that passes exactly four birds and includes the three birds that they saw. The fourth bird seen along this route is the bird in Option A.



To be completely sure that the family could not have seen any of the other birds consider what happens if they do **not** see the bird in Option A. In order to avoid seeing the bird in Option A the family must not take either of the two paths that are adjacent to the bird in Option A. Of the remaining paths, there are three possible routes the family could take.



In the first possible route the family would see exactly two birds. In the second and third possible routes the family would see exactly five birds. There is no route that allows the family to see exactly four birds.

Therefore, the family must have seen the bird in Option A.

### Connections to Computer Science

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

Different computer algorithms can be used to find *cycles* in a graph. A cycle is a sequence of edges that can be followed starting from a vertex and ending at this same vertex. In our task, we wish to find a cycle starting from the family's home. The cycle must include exactly four birds, where three of the birds are specified.

One of the best known algorithms for finding cycles in a graph is *Floyd's cycle-finding algorithm*.

### Country of Original Author

Iceland



## Rare Mushrooms

### Story

Colby wants to take a picture of a rare mushroom. To determine whether or not a mushroom is rare, Colby assigns points to the stem and cap according to the following table:

Points for Stem		Points for Cap		
				
Plain	Layered	Dotted	Horned	Striped
0 points	2 points	1 point	3 points	4 points

A mushroom that scores 5 points or more is rare and a mushroom that scores less than 5 points is not rare.

### Question

Which one of the following four mushrooms is rare?



**Answer**

(B)

**Explanation of Answer**

The following table computes the score for each of the four mushrooms:

Mushroom	Stem Score	Cap Score	Total Score	Result
(A) 	Plain (0 points)	Dotted (1 point)	1 point	Not rare
(B) 	Layered (2 points)	Horned (3 points)	5 points	Rare
(C) 	Layered (2 points)	Dotted (1 point)	3 points	Not rare
(D) 	Plain (0 points)	Striped (4 points)	4 points	Not rare

Therefore, the only rare mushroom is the mushroom in Option B.

## Connections to Computer Science

This is an example of a *classification task*, a very important problem in computer science. In a classification task, we try to classify each item with a certain label based on its characteristics (usually called *features*).

For this task, given a set of mushrooms, we classify each of them as rare or not rare, given its stem and cap features.

Common real-life examples of classification tasks include classifying medical images (for example, whether a tumour is present), classifying documents (for example, automatically labelling news articles in the category of sports, politics, or education), facial recognition, and email spam detection.

The table describing the scores for each feature of the mushroom is known as a *scoring function*. Typically, various features of an item are scored and the item is classified into a category, based on whether the sum of the scores is above or below some threshold.

A typical example of this kind of scoring function would be a *spam filter* for email messages. Based on keywords in the message and transmission details (such as who the sender is), scores are assigned to the message and if the total score is more than a certain threshold, the message will be classified as spam.

This classification task is one of the main problems in the area of *data science*.

## Country of Original Author

South Korea



## Part B

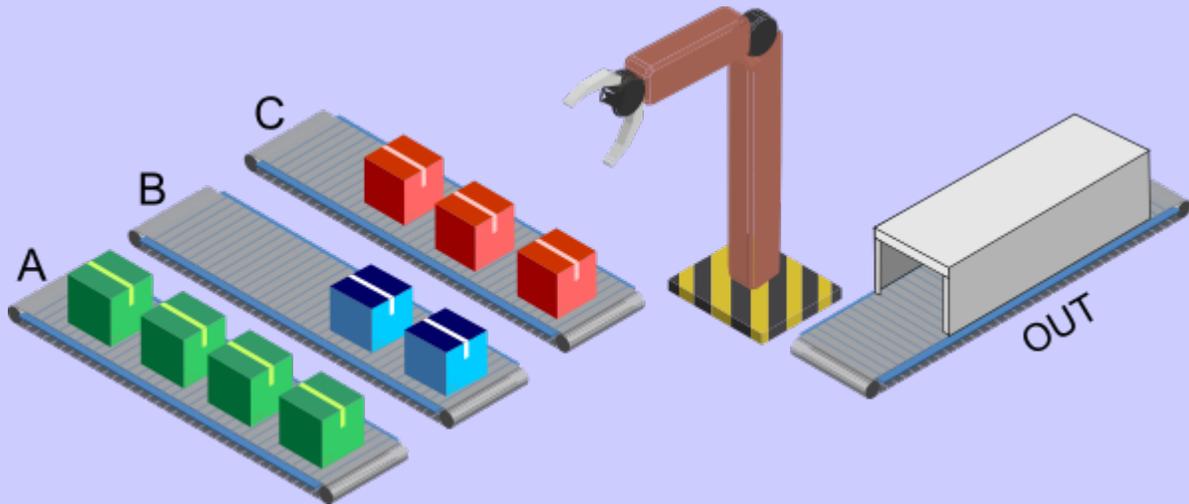
## Moving Packages

### Story

A robotic arm takes packages from three conveyor belts (labelled A, B, and C) and moves them to the conveyor belt labelled OUT. The rules for the robotic arm are as follows:

- If there is a package on belt A, take one and move it to belt OUT. Then,
- if there is a package on belt B, take one and move it to belt OUT. Then,
- if there is a package on belt C, take one and move it to belt OUT. Then,
- move to belt A and start again.

If the robotic arm is ready to take a package from a particular belt, but no package is available there, the robotic arm will shut down.



### Question

Given the arrangement of packages on the three belts as shown, how many packages will the robotic arm move before shutting down?

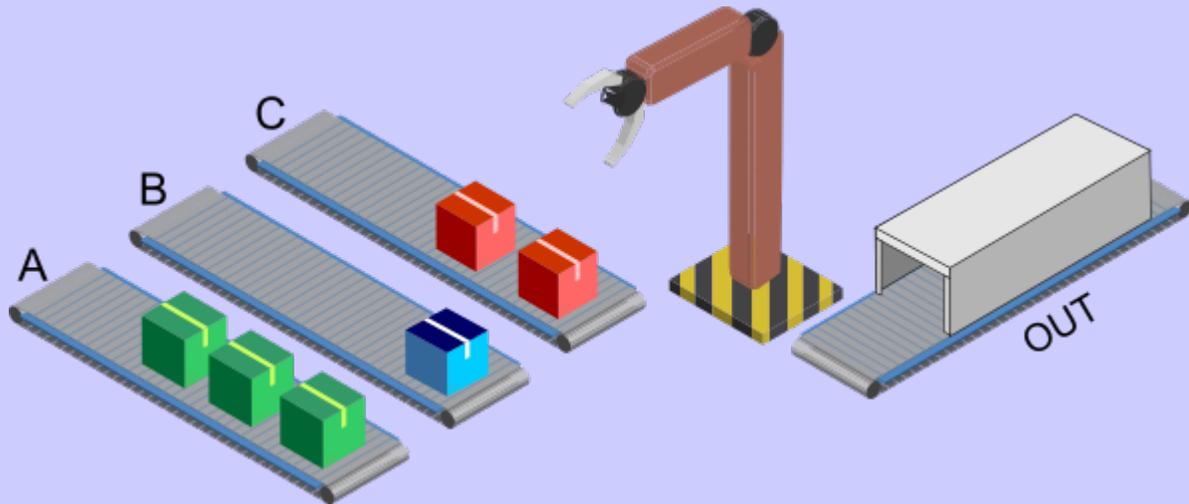
- (A) 9
- (B) 8
- (C) 7
- (D) 6

**Answer**

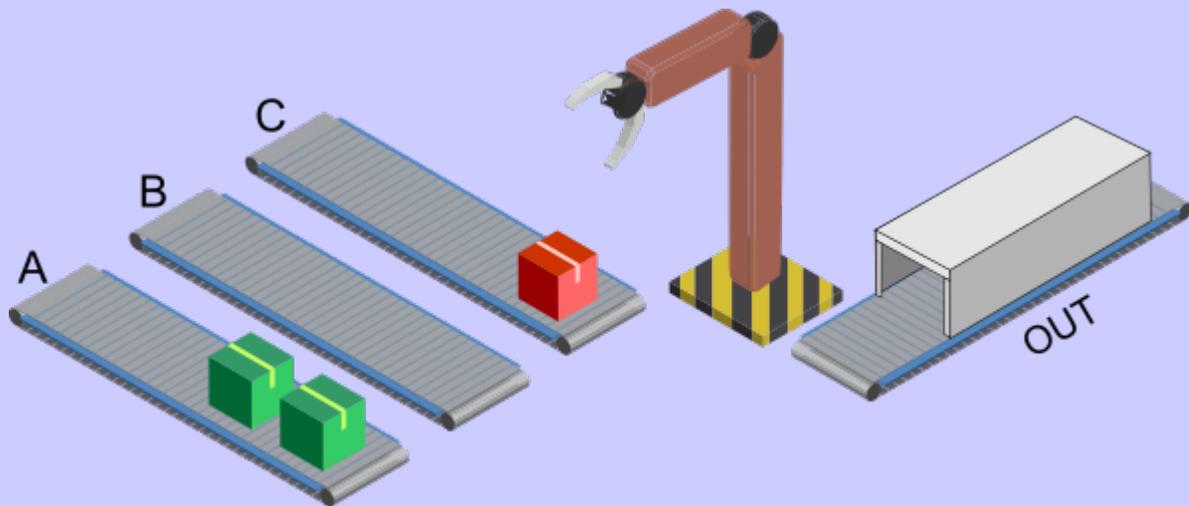
(C) 7

**Explanation of Answer**

The robotic arm will first move one package from belt A, then one from belt B, and then one from belt C, leaving packages on the belts as shown:



The robotic arm will then move to belt A and start again. The arm will move one package from belt A, then one from belt B, and then one from belt C, leaving packages on the belts as shown:



The robotic arm will then move to belt A and start again. The arm will move one package from belt A, and then shut down since belt B is empty.

The robotic arm will move 3 packages from belt A, 2 packages from belt B, and 2 packages from belt C, for a total of 7 packages.

## Connections to Computer Science

When solving a problem it is often helpful to keep track of the current position or *state* of an object. In this problem, the arm position and the number of packages on each belt determine the state.

Performing the same action on an object may not always have the same effect. Actions done on an object can have different effects based on its current state. For example, if the arm moves to a belt, even though it may have removed a package from that same belt earlier in the process, it may be unable to remove a package now, since the state has changed.

This also applies to computers. For example, when you are drawing a picture, the parts that you have already drawn determine the state of your picture. Adding new lines or deleting some lines change the state of the picture and a drawing program needs to keep track of this. Using the fill tool can change the colour of bigger or smaller parts of the picture depending on which lines have been drawn.

There are many other examples where state is important. When following a map on a phone, the important state will include your current location.

When writing a computer program, a programmer has to decide what the state of the system they are working with is and write the program to correctly keep and update that state.

## Country of Original Author

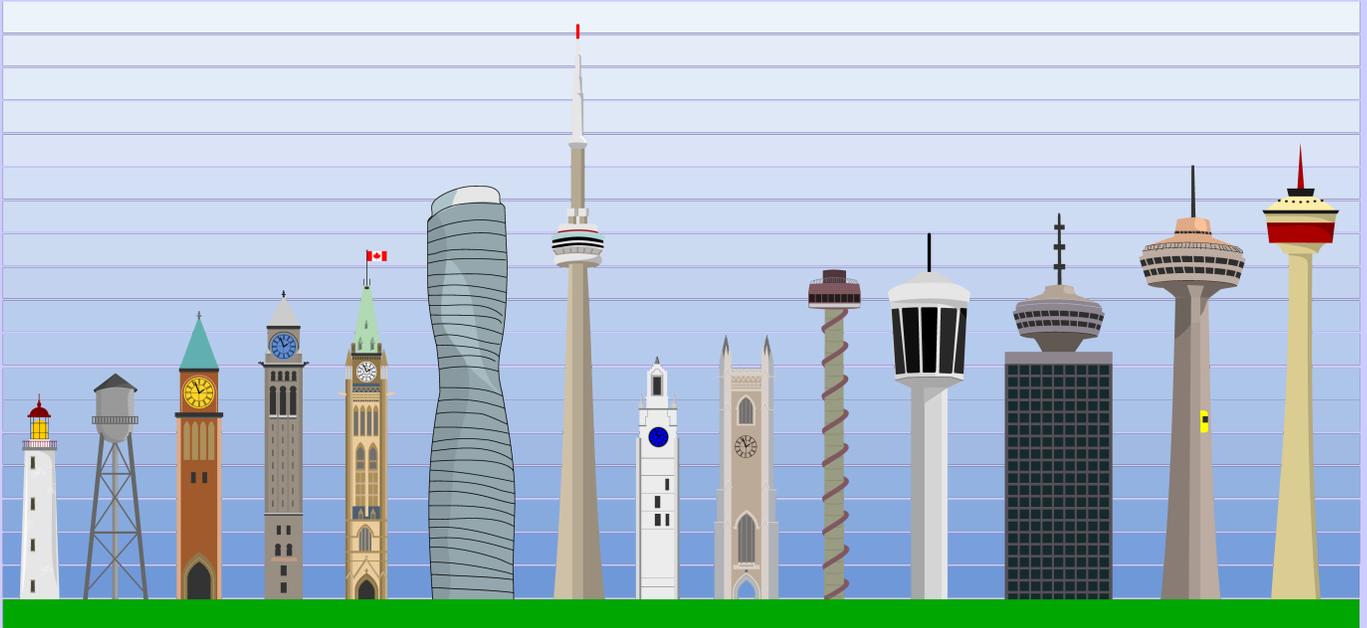
Belgium



# Skyline

## Story

A skyline consists of 14 towers as shown.



The height of a tower is measured from the bottom of its base to its highest point, including any flagpoles or antennas.

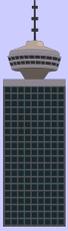
## Question

If the towers are listed from shortest to tallest, which tower would be 10th in the list?

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

### Answer

(A)



### Explanation of Answer

Notice that the skyline consists of one sequence of towers that gets taller from left to right followed by a second sequence of towers that gets taller from left to right. Also notice that the last two towers in each of these sequences make up the four tallest towers overall. (This did not have to be true. It could have been the case, for example, that the four tallest towers were all at the end of one of the two sequences.)

Since there are 14 towers in total, if we ignore the four tallest towers, the tallest remaining tower will be 10th in the list. The tallest remaining tower is the one in Option A.

### Connections to Computer Science

In this task, you are asked to determine which tower would be the 10th in a list of towers if all the towers were ordered by height. Arranging items in order is called *sorting*.

There are many well-known sorting algorithms and this task is related to the *mergesort algorithm*. The key idea behind this technique is to separately sort the two halves of the items. Then you must *merge* these two sorted lists to produce one completely sorted list. If you chose to solve this problem by sorting all 14 towers, then you could have been merging the sorted towers in the left half with the sorted towers in the right half.

### Country of Original Author

Canada



# Market Exchange

## Story

A beaver goes to a market to trade items. It has one carrot  but needs one fir tree .

Each stall of the market allows a different trade as shown:

Stall	Give	Get
P		
Q		
R		
S		
T		
U		
V		
W		

## Question

Which of the following sequences of stalls should the beaver visit in order to trade its carrot  for one fir tree .

- (A)  $P, Q, T$
- (B)  $W, T, U$
- (C)  $S, V, U$
- (D)  $S, R, U$

## Answer

(C)  $S, V, U$

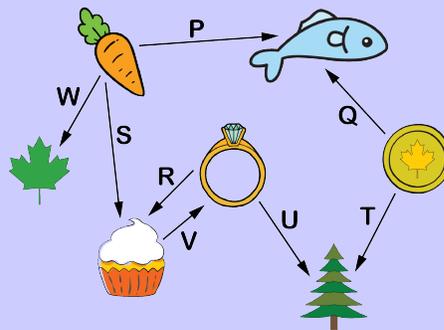
## Explanation of Answer

At stall  $S$  the beaver can trade the carrot 🥕 for a cupcake 🍰.

At stall  $V$  the beaver can trade the cupcake 🍰 for a ring 💍.

At stall  $U$  the beaver can trade the ring 💍 for a fir tree 🌲.

To consider the other options, the following diagram can help us understand the possible trades at various stalls. An arrow from item  $X$  to item  $Y$ , labelled  $Z$ , means that at stall  $Z$  the item  $X$  can be traded for item  $Y$ .



Notice how you can now trace the connection from a carrot to a fir tree. Also, notice that when starting at the carrot, the only way to reach the fir tree is by following the arrows labelled  $S$ ,  $V$ , and  $U$ , in that order.

## Connections to Computer Science

The ways in which items are traded at this market can be modelled by a *directed graph*. We can represent the items as the *vertices* of the directed graph and a possible trade of one item for another as a (*directed*) *edge*. In the diagram, the edges are the arrows labelled with the particular stall that can trade one item for another.

The reason we say “directed” is because the trade is only in one direction. For example, if item A can be traded for item B at stall X, then we have a directed edge from item A to item B labelled with X. In this case, we call B a *neighbour* of A.

In this problem, we want to find a *directed path* from the carrot to the fir tree. One way to do this is to perform a *breadth-first search*, where all of the neighbours of the carrot 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

Switzerland



# Beaver Homes

## Story

Beaver homes are identified using symbols rather than digits according to the table shown:

	-	=	≡	▷	▷
□	0	1	2	3	4
◁	5	6	7	8	9

The symbol assigned to a row and the symbol assigned to a column are combined to form a new single symbol. This symbol represents the digit where that row and column meet.

For example, the symbol  represents the digit 5, since it is a combination of its row symbol  and its column symbol .

	-	=	≡	▷	▷
□	0	1	2	3	4
◁	5	6	7	8	9

Here is a picture of one beaver's home:



## Question

What four-digit number is represented by the symbols on this beaver's home?

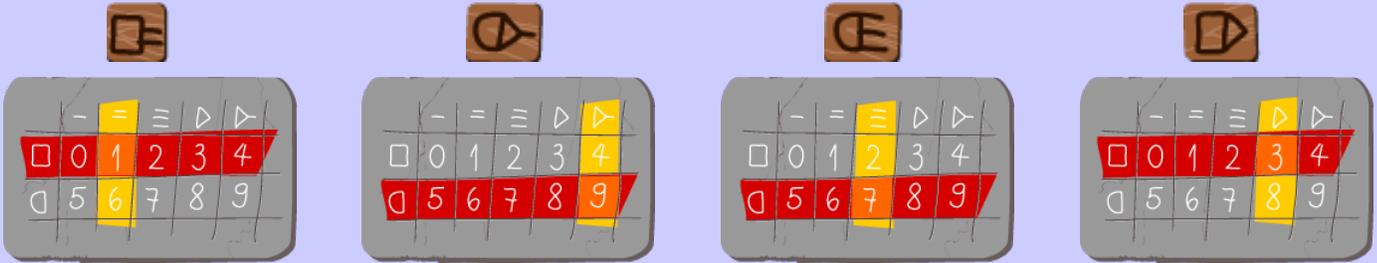
- (A) 1874
- (B) 1923
- (C) 1824
- (D) 1973

### Answer

(D) 1973

### Explanation of Answer

To find the four-digit number, we translate each of the four symbols using the given table.



Therefore, the four-digit number represented by the symbols on the beaver's home is 1973.

### Connections to Computer Science

This task focusses on *encoding*. Encoding is the process of applying a code to data in order to represent the data in a different but equivalent way. The encoding in this task is represented by the table, which defines which combinations of symbols represent a particular digit from 0 through 9.

Billions of times a day, some form of digital message is sent using computers or smartphones. We typically write messages in a *natural language* such as English but some code is needed to store the message on a computer. Two common encodings of information are *ASCII*, which deals with simple text characters, and *Unicode*, which deals with text, accented letters, and emojis.

How to encode and decode information falls under a field of research called *information theory* which includes the study of *data compression* (how to store data using less memory) and *cryptography* (how to store or send information securely).

### Country of Original Author

Switzerland



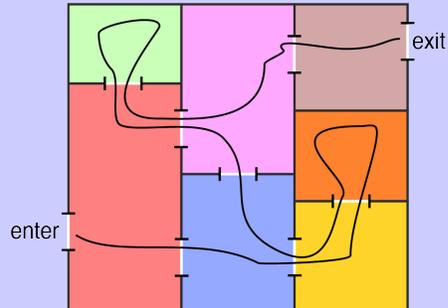
## Part C

# Museum Tour

## Story

A new museum with seven rooms has been constructed. The builders are now trying to decide where to place the doors between rooms, so that visitors can enter, walk through the rooms, and exit.

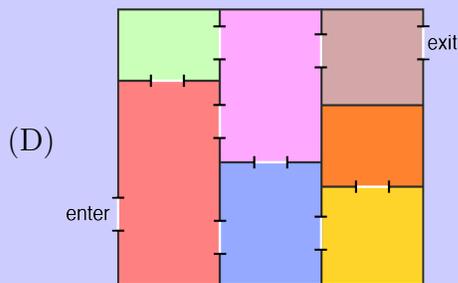
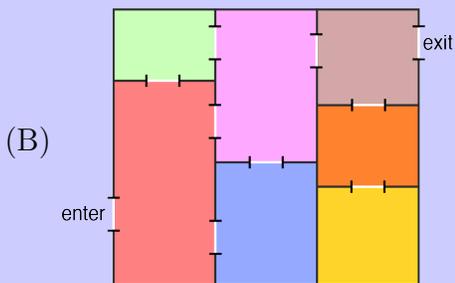
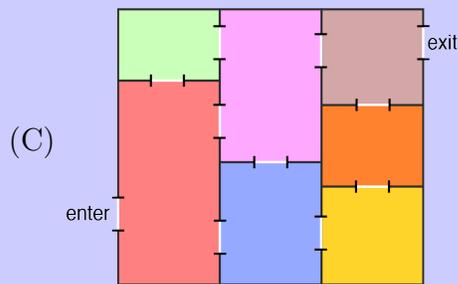
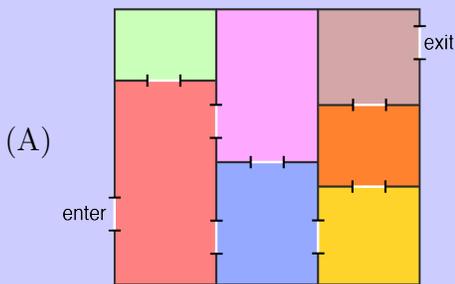
The following possible layout of doors shows how guests might walk through the museum. Notice that some rooms are visited multiple times.



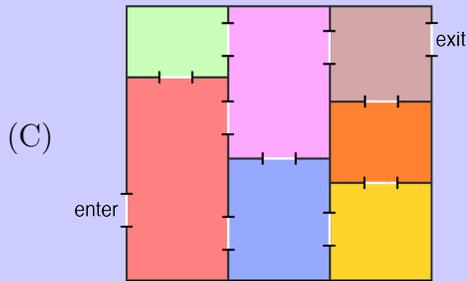
In an ideal layout, guests should be able to visit each room without having to walk through any room more than once.

## Question

Which one of the following layouts makes it possible for guests to tour the museum by visiting each room exactly once?

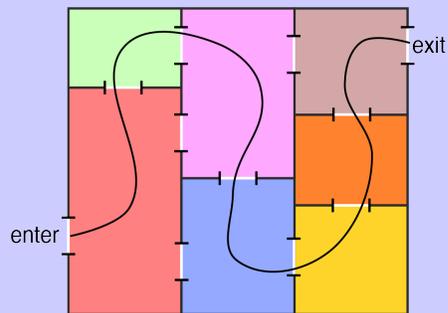


## Answer



## Explanation of Answer

The following tour shows how the layout in Option C allows guests to visit each room in the museum exactly once:



Now we must explain why the other layouts do *not* allow us to draw such a tour. If a room has only one door, guests will need to exit into the same room that they entered from. So they will have visited that room twice.

Any layout that includes a room with only one door will make it impossible for guests to tour the museum by visiting each room exactly once.

In Option A, the room in the top-left corner has only one door.

In Option B, the room in the bottom-right corner has only one door.

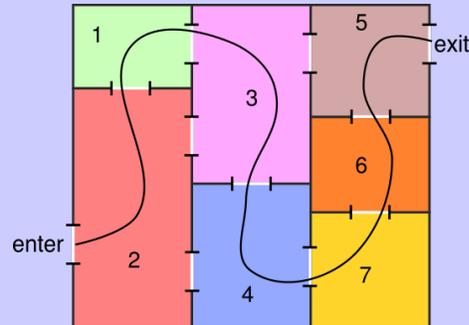
In Option D, the middle room on the right-hand side has only one door.

Therefore, of the possible layouts, only the layout in Option C allows guests to tour the museum by visiting each room exactly once.

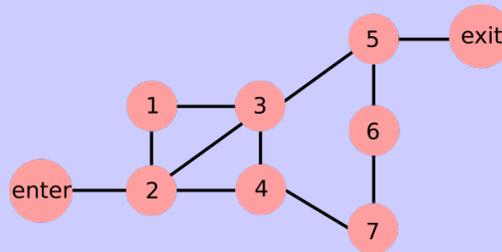
## Connections to Computer Science

To solve a problem using a computer, we must first model the problem in a way that is better understood by computers.

The most important information about the possible museum layouts is the location of the doors. Consider the layout in Option C. If we add room numbers as follows:



then we can model this layout as follows:



A circle indicates a room (plus the “outside rooms” before entering and after exiting), and a line between circles indicates that there is a door between rooms. This model is called a *graph*, the circles are called *vertices*, and the lines are called *edges*.

If a layout allows guests to visit each room without having to walk through any room more than once, then the layout’s corresponding graph will include a path along edges from the entrance to the exit that visits each vertex exactly once. Such a path is called a *Hamiltonian path*.

Determining whether or not a graph has a Hamiltonian path is an inherently difficult task. Since the number of rooms and doors is small in this problem, we can solve it without the use of computers. However, if the number of rooms and doors get large, even the fastest computers may fail to solve this problem in a reasonable amount of time.

## Country of Original Author

Switzerland

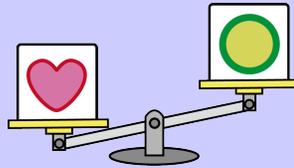


## Weighing Boxes

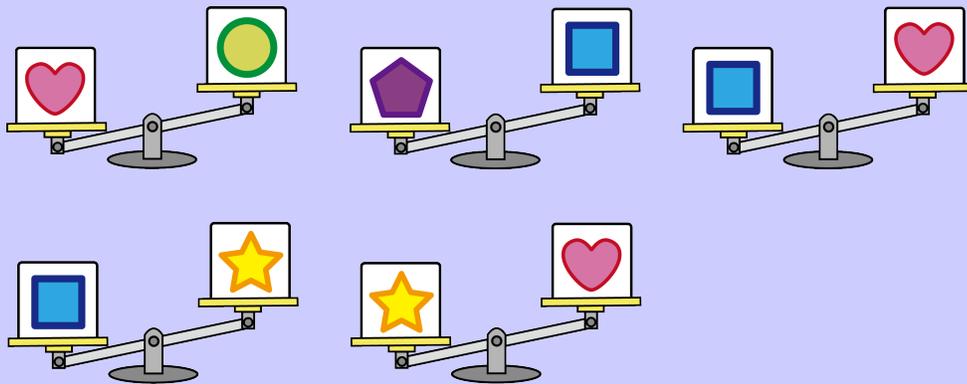
### Story

There are five boxes, each featuring a different shape, and each having a different mass. Using a scale we can compare the masses of two boxes.

For example, the following scale shows that  is heavier than .



Five comparisons were made, and the results are shown on the following scales:



### Question

If we arrange the boxes in order from heaviest to lightest, which box would be in the middle?

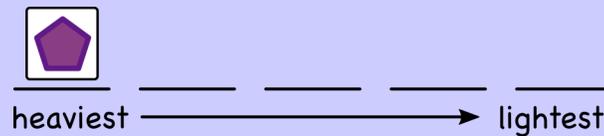
- (A)       (B)       (C)       (D) 

## Answer

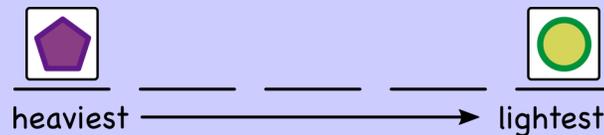
(B) 

## Explanation of Answer

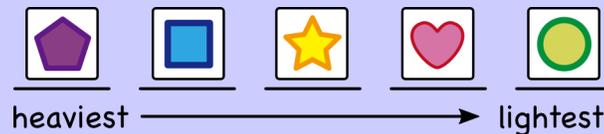
We can start by arranging the boxes in order from heaviest to lightest. The heaviest box will always appear on the lower part of the scale. Looking at all five boxes and all five comparisons, the pentagon box is the only box that meets this criteria. All the other boxes are on the upper part of the scale in at least one comparison. So the pentagon box is the heaviest box.



Similarly, the lightest box will always appear on the upper part of the scale. Looking at all five boxes and all five comparisons, the circle box is the only box that meets this criteria. All the other boxes are on the lower part of the scale in at least one comparison. So the circle box is the lightest box.



Now we are left with the square, heart, and star boxes. Looking at the bottom two comparisons, we see that the star box is lighter than the square box, but heavier than the heart box. That means the star box must be in between the square box and the heart box.



Therefore, the star box is in the middle when we arrange the boxes in order from heaviest to lightest.

### Connections to Computer Science

This problem involves *sorting* elements. The particular *sorting algorithm* that was used in the explanation was *selection sort*. Selection sort finds the heaviest/largest value by comparing the currently known heaviest/largest value to all other values. In this problem, each different scale arrangement is a different *comparison*, and sorting algorithms try to minimize the total number of comparisons.

*Sorting algorithms* are algorithms that put elements of a list in a certain order. The most-used orders are ascending and descending. From the beginning of computing, the sorting problem has attracted a great deal of research. Many sorting algorithms were developed throughout the years e.g. bubble sort, insertion sort, and quicksort. Each algorithm has its own characteristics. Some algorithms are faster than others, some are more complicated, and some others use special techniques such as *recursion*.

### Country of Original Author

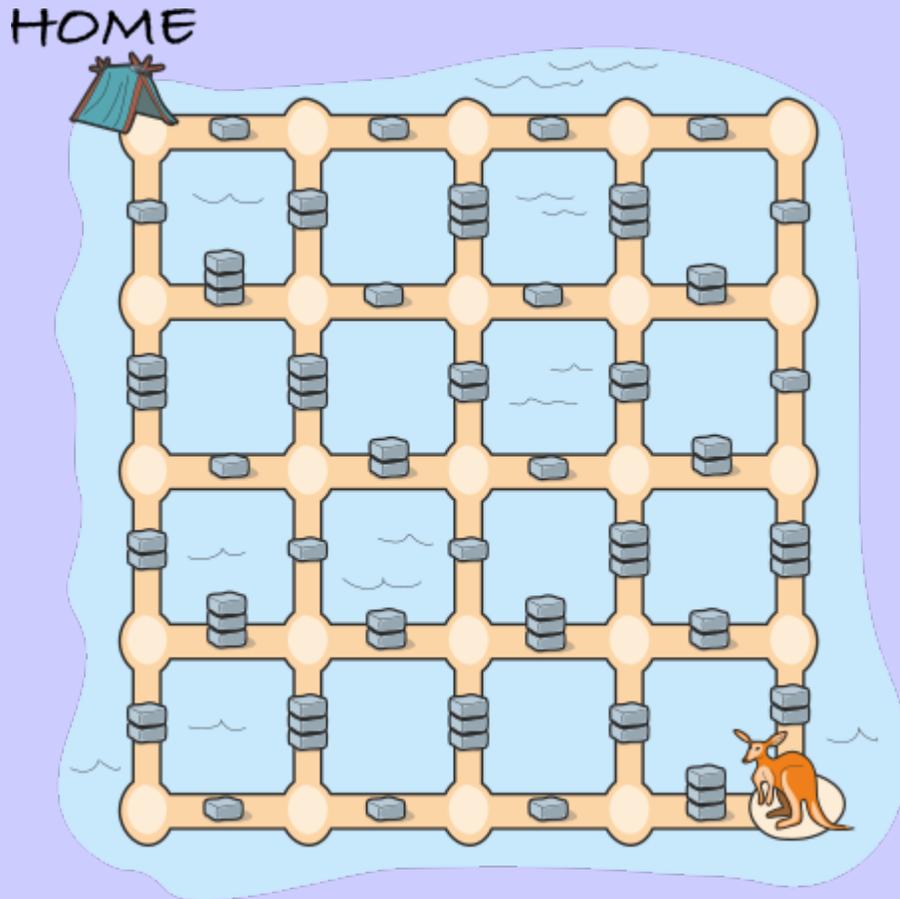
Japan



## Jumping Kangaroo

### Story

Kanga Roo is jumping home along the vertical and horizontal paths. Kanga jumps over exactly one pile of bricks with each jump. Kanga cannot jump over brick piles that have a height of 3 bricks.



### Question

If Kanga wants to jump home using the fewest jumps possible, how many jumps must Kanga make?

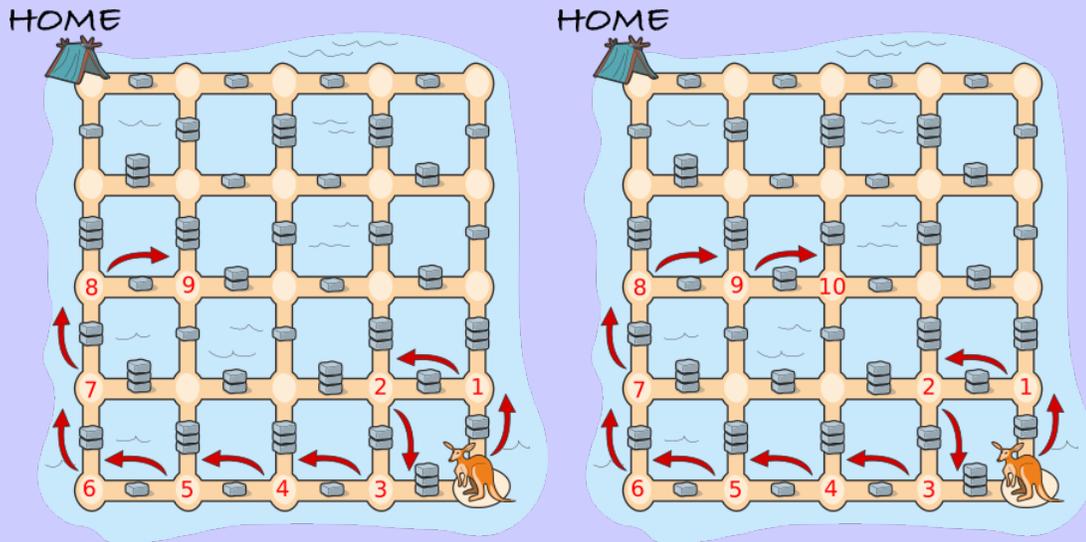
- (A) 8
- (B) 13
- (C) 14
- (D) 16

Answer

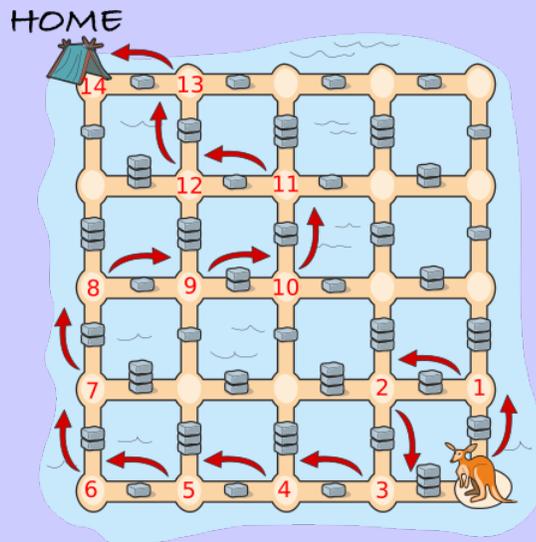
(C) 14

Explanation of Answer

Kanga does not have a choice for the first 9 jumps because of the positions of the brick piles that are 3 bricks high. After the 9th jump, Kanga can either jump right OR down-right-up. Both bring Kanga to the same place on the paths. Since one jump is fewer than three, Kanga jumps right.



Now notice that any jump either up or left brings Kanga closer to home, and any jump either right or down brings Kanga further from home. As long as Kanga jumps only up and left, the jumps will be minimal. As it turns out, from Kanga's current position this is possible to do all the way home.



Kanga can get home in as few as 14 jumps.

### Connections to Computer Science

One way to find a solution to this problem is to apply a *depth first search (DFS)* algorithm, which is also known as a *backtracking* algorithm. This technique tries to reach the destination by “moving forward” until either the solution is found, or there are no more possible “forward” movements, at which point we “backtrack” to an earlier position and try another path forward.

This backtracking concept is a technique used in many applications: solving puzzles like sudoku, determining possible moves in games like chess, or combinatorial optimization problems like how to ship packages to a variety of locations as cost-effectively as possible.

### Country of Original Author

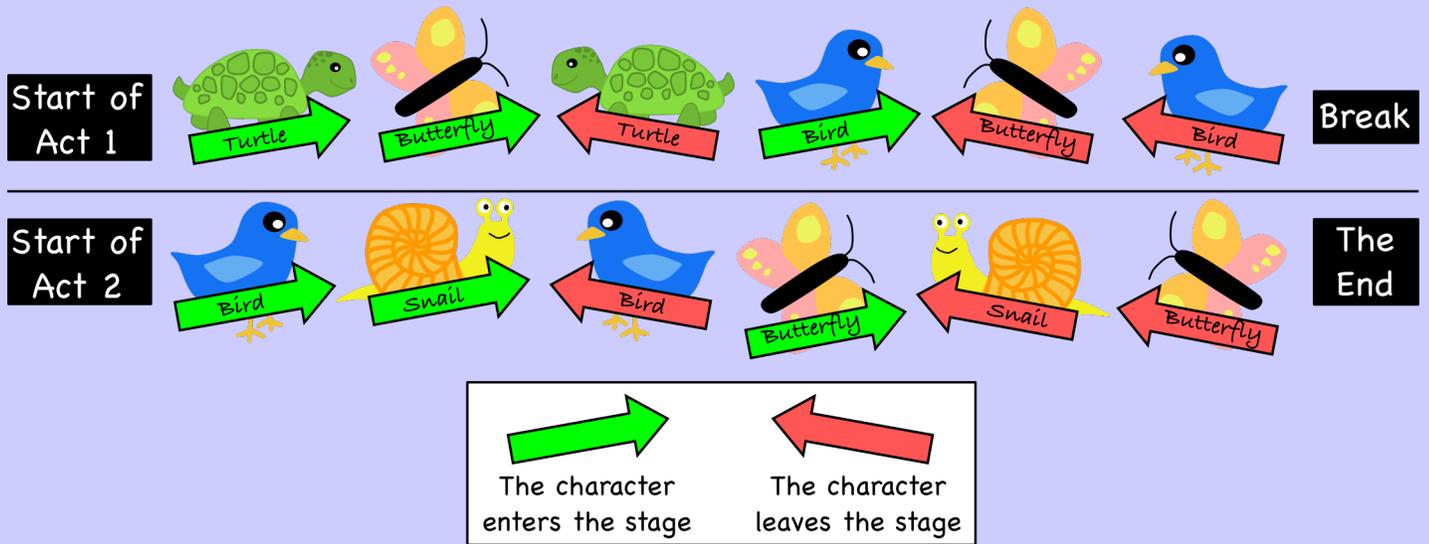
Lithuania



# Theatre Performance

## Story

Four characters are in a play. They enter and leave the stage according to the order shown, read from left to right. The play has two acts and one break between the acts.



## Question

Which statement is *not* true?

- (A) The snail and the butterfly were together on the stage.
- (B) The turtle and the bird were together on the stage.
- (C) The snail entered the stage after the break.
- (D) The snail and the bird were together on the stage.

### Answer

(B) The turtle and the bird were together on the stage.

### Explanation of Answer

The statement in Option A, “the snail and the butterfly were together on the stage”, is true. In Act 2, the snail entered the stage, the bird left the stage, and then the butterfly entered the stage. Therefore, the snail and the butterfly were on the stage at the same time.

The statement in Option B, “the turtle and the bird were together on the stage”, is **not** true. In Act 1, the turtle left the stage right before the bird entered the stage, and the turtle never appeared again in the play.

The statement in Option C, “the snail entered the stage after the break”, is true. The snail was the second character to enter the stage in Act 2, right after the bird.

The statement in Option D, “the snail and the bird were together on the stage”, is true. At the start of Act 2, the bird entered the stage and then the snail entered the stage. Therefore, the snail and the bird were on the stage at the same time.

### Connections to Computer Science

This problem focusses on keeping track of the *state* of a system. The state in this problem is the set of characters which are currently on the stage. Since a character is either on the stage or off the stage, we can keep track of each character using a *binary number*. A binary number is either the number 0 or 1, which we can think of as “off” or “on”.

We can then think about the state of the stage as being a list of four binary numbers. For example, if we list the characters in the order (bird, butterfly, snail, turtle), then the state (1, 0, 1, 0) would indicate that bird and the snail are on the stage, and the butterfly and the turtle are off the stage.

This problem then involves changing the state every time a character enters or exits the stage. This operation of updating a state is one of the fundamental operations of a *CPU*: given the current state and some input, what new state should the computer be in? Every mouse click or key press changes the state of the computer, and the state is being stored on the computer as a sequence of binary numbers.

### Country of Original Author

Slovakia

