UNIVERSITY OF WATERLOO

The CENTRE for EDUCATION in MATHEMATICS and COMPUTING

2024 Beaver Computing Challenge (Grades 7 & 8)

Questions, Answers, Explanations, and Connections

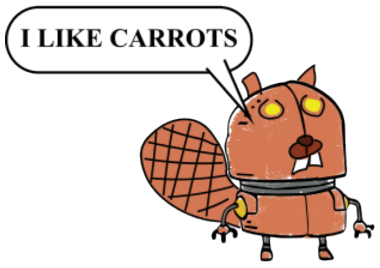
Part A

Beaver Robot

Story

Beaver Robot can say sentences containing exactly 3 words.

- The first word must be chosen from Box 1.
- The second word must be chosen from Box 2.
- The third word must be chosen from Box 3.



Box 1	Box 2	Box 3
I, YOU	LOVE, LIKE	HELP, SWIM
BEAVERS	CAN, ME	CARROTS
CAN, HELP	HELP, NEED	LOVE, YOU
PLEASE	YOU, US	WATER

Question

Which sentence below **cannot** be said by Beaver Robot?

(A) CAN YOU HELP

(B) BEAVERS CAN SWIM

(C) I LOVE YOU

(D) YOU NEED ME

Answer

(D) YOU NEED ME

Explanation of Answer

The word “ME” is the third word of the sentence in Option D but it is not in Box 3, so Beaver Robot cannot say this sentence.

Looking at the sentences in Options A, B, and C, the first words are “CAN”, “BEAVERS”, and “I”, which are all in Box 1. The second words are “YOU”, “CAN”, and “LOVE”, which are all in Box 2. The third words are “HELP”, “SWIM”, and “YOU”, which are all in Box 3. Thus, the sentences in Options A, B, and C can all be said by Beaver Robot.

Connections to Computer Science

Creating computer programs that communicate with humans has been studied for over 60 years. In the mid-to-late 1960s, a program called *ELIZA* was developed to explore how simple communication between a human and computer could occur. ELIZA would ask questions, and repeat back parts of the response of the user to convey “understanding” or “listening” to the human.

Underlying these communication programs is a *sentence generating system*. The system used in this task is very simple. For more complex modern systems, such as *ChatGPT*, much more complicated systems are needed. One current popular approach used in *artificial intelligence* involves *large language models (LLMs)* which use a very large collection of data to “learn” attributes of correct sentences.

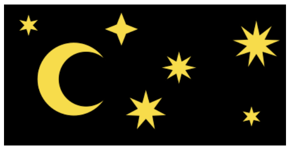
Country of Original Author

Vietnam

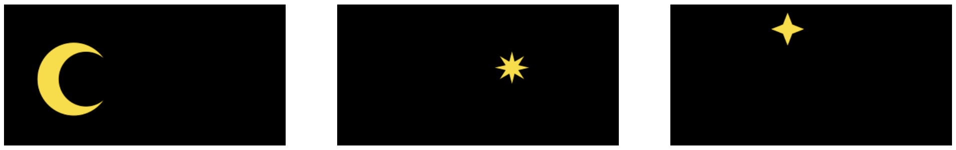
Digital Image

Story

Ana created the following digital image.



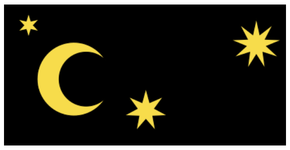
She did her work in stages. For the last three stages, she added the following parts of the image as shown in order from left to right.

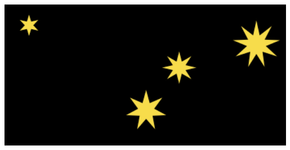


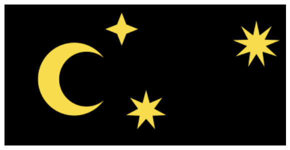
Now Ana has decided to undo some of her work, removing the parts added in the last three stages.

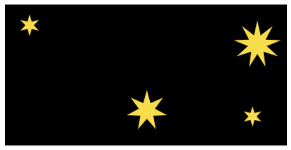
Question

Which of the following options shows the resulting picture?

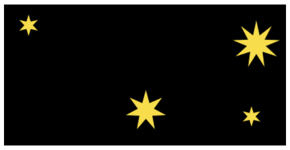
(A) 

(B) 

(C) 

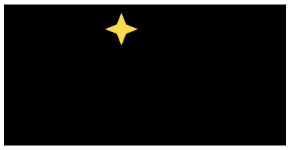
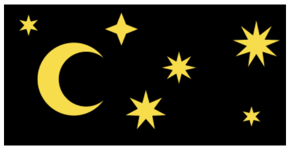
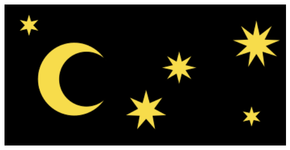
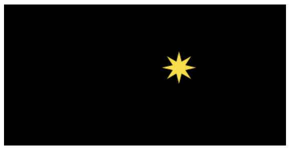
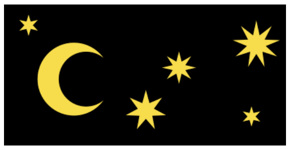
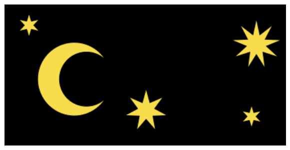
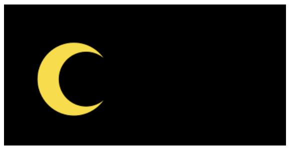
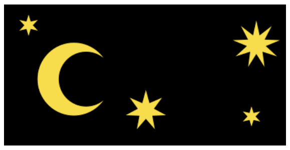
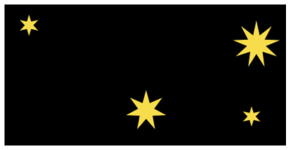
(D) 

Answer

(D) 

Explanation of Answer

Undoing the last three stages of Ana's work in reverse order results in the following sequence of pictures.

Part Undone	**Picture Before**	**Picture After**
		
		
		

The final picture above matches Option D.

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 added to the digital image by Ana can be thought of as elements put, or *pushed*, into a stack. When parts of the digital image are removed from the picture, these elements are removed, or *popped* from the stack.

Stacks are used frequently in computer science. One such usage is keeping track of operations that can be "undone", such as edits to a document, browsing history in a browser, or searching through a maze by way of *backtracking* if a dead end is reached.

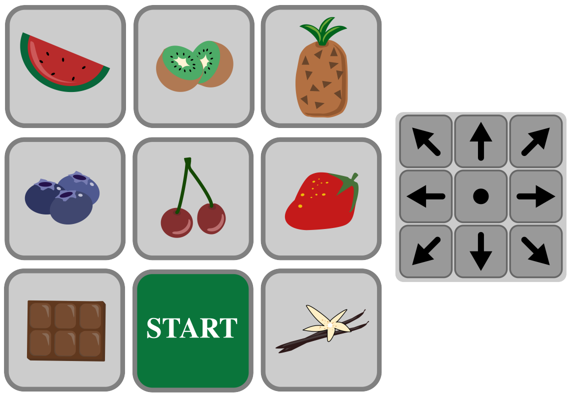
Country of Original Author

North Macedonia

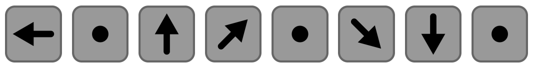
Ice Cream Shop

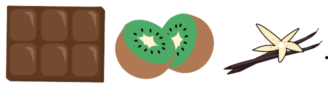
Story

A cool new ice cream shop with a self-service machine has opened! To place an order, you push the arrow buttons to move the robot to the square with the flavour you want and then press ● to add a scoop to the cone. After you have chosen three scoops, your order is prepared.



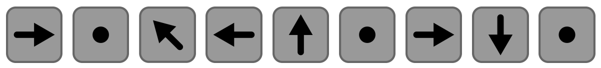
The robot always starts an order from the START square. For example, from left to right the sequence

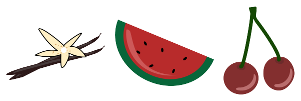


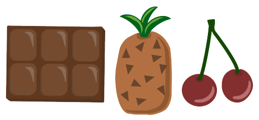
makes a cone where the flavours from bottom to top are .

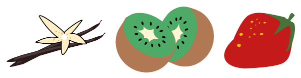
Question

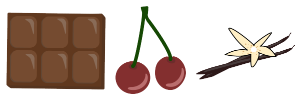
What flavours will be on the ice cream cone, from bottom to top, if the order uses the following sequence, from left to right?



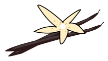
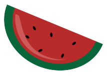
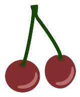
(A) 

(B) 

(C) 

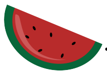
(D) 

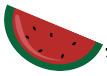
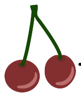
Answer

(A)   

Explanation of Answer

From the START square, the robot first moves → to .

From , it moves ↖ ← ↑ to .

From , it moves → ↓ to .

Connections to Computer Science

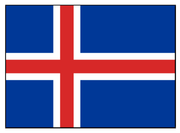
When an arrow button is pushed to move the robot, that is analogous to giving an *instruction* to a computer. A sequence of instructions is a *program*, which is the most common way of describing an *algorithm* to accomplish some goal in a language the computer understands.

When a program contains an incorrect instruction, or the sequence of instructions is not correct, this error is often referred to as a *bug* in the program. For example, in this task, the ice cream robot might attempt to move up when it is on the kiwi (which might have undefined behaviour) or dispense the incorrect flavour due to moving left when it should have moved right.

In order to minimize the number of bugs in a program, it is important to *trace* the instructions either on paper or "in your head", and also to create *test cases* that ensure that the program produces the expected output for particular input.

Country of Original Author

Iceland

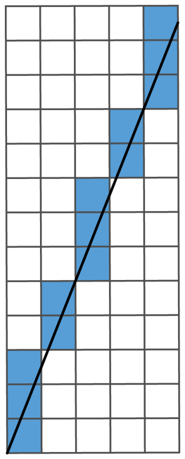


Line Drawing

Story

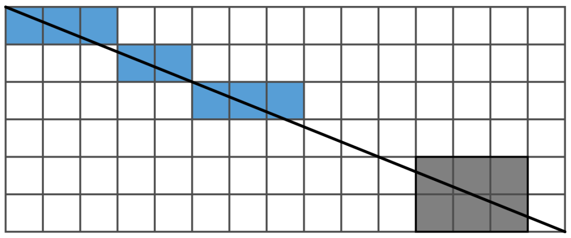
When Pixel tries to draw lines on graph paper, she does this by colouring in blocks of tiny squares. Each of these squares intersects the line she is trying to draw. This means she cannot always draw perfectly straight lines. An example is shown where the blocks of squares Pixel coloured do not look exactly the same as the straight diagonal line that she is trying to draw.

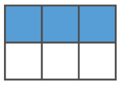
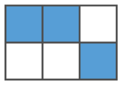
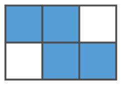
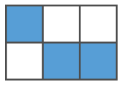
A pattern is formed by the shapes and sizes of blocks of squares that Pixel colours. This pattern repeats every time the line she is trying to draw passes through a corner of a square. In the example, it is always 3 vertical blocks followed by 2 vertical blocks.



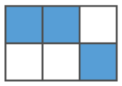
Question

Pixel has started to draw the diagonal line shown and coloured in the first 8 squares as shown. If she continues drawing the line, then how will she colour the 2×3 block of squares shown in grey?



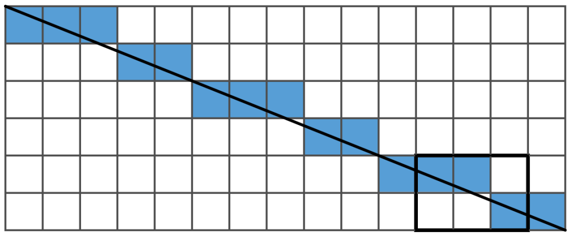
(A)  (B)  (C)  (D) 

Answer

(B) 

Explanation of Answer

By inspecting where this line passes through the corner of a square, we can see that the pattern for this line is 3 horizontal blocks followed by 2 horizontal blocks. Knowing this, we can extend the pattern to determine how Pixel will colour the graph paper as she tries to draw the diagonal line shown.

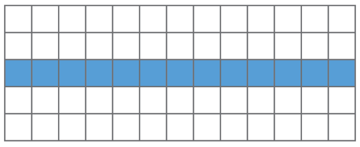
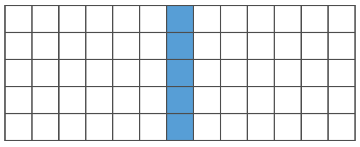


The 2×3 block of squares shown in grey in the question and shown in a black box here is coloured as shown in Option B.

Connections to Computer Science

Creating images on a computer screen that are as realistic as possible has been one driving force in computer graphics. The limitation of resolution in the early days of computers, when the standard in the 1970s was 320 x 200 pixels, caused many graphics to seem very choppy and unrealistic. In order to help smooth out the edges of images techniques such as *anti-aliasing* were invented.

Notice that if the image is a horizontal or vertical line, the line will appear perfectly straight, as shown below.

In this task, one simple *line drawing* algorithm is demonstrated to highlight how a line which is non-vertical and non-horizontal can be approximated by a series of pixels. This algorithm uses the slope of the line to determine which pixels should be turned on or off. The underlying algorithm which was followed in this task is known as *Bresenham's algorithm*. More advanced algorithms, such as the *Gupta-Sproull algorithm*, give a much closer approximation and more realistic image to a given line.

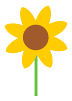
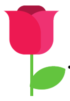
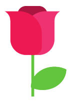
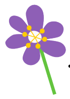
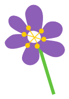
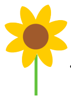
Country of Original Author

Canada

Magic Garden

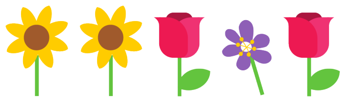
Story

In a magic garden, flowers can change during the night. If there is a group of at least two flowers of the same type directly beside each other, then that group of flowers transforms into a group of different flowers as follows:

- Groups of  transform to .
- Groups of  transform to .
- Groups of  transform to .

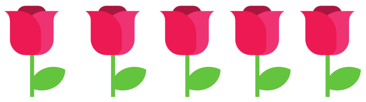
A flower never changes more than once per night.

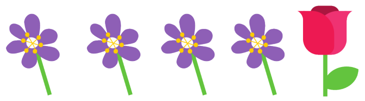
This is what the magic garden looks like one morning.

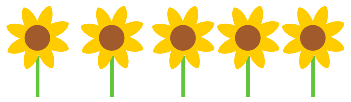


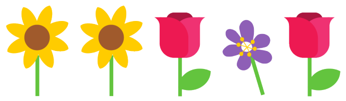
Question

What will the magic garden look like after four nights have passed?

(A) 

(B) 

(C) 

(D) 

Answer

(A) 🌷 🌷 🌷 🌷 🌷

Explanation of Answer

At the beginning the magic garden looks like this: 🌻 🌻 🌷 🌸 🌷.

The 🌻 are the only flowers of the same type directly beside each other, so they will transform to 🌷.

Thus, after the first night, the magic garden looks like this: 🌷 🌷 🌷 🌸 🌷.

Now, there are three 🌷 beside each other, so they will transform to 🌸.

Thus, after the second night, the magic garden looks like this: 🌸 🌸 🌸 🌸 🌷.

Now there are four 🌸 beside each other, so they will transform to 🌻.

Thus, after the third night, the magic garden looks like this: 🌻 🌻 🌻 🌻 🌷.

Now there are four 🌻 beside each other, so they will transform to 🌷.

Thus, after the fourth night, the magic garden looks like this: 🌷 🌷 🌷 🌷 🌷.

Connections to Computer Science

This task illustrates three computational thinking concepts: maintaining *state*, *sequential processing* and *conditional evaluation*.

The represention of the flowers of the garden after each day can be thought of as the *state* of the system, which is updated every night. This operation of updating a state is one of the fundamental operations of the *central processing unit (CPU)*: given the current state and some input, what new state should the computer be in?

Since there are a total of four nights passing in this task, we can think of each night as a one *sequential step* in the algorithm. That is, each step needs to know what the result of the previous step was, and will compute a new state from that previous step. Most computer *algorithms* are described as a sequential set of steps.

The transformation of the flowers can be thought of as a *conditional* instruction: an instruction happens only if certain conditions are met. For this task, *if* there is a group of flowers of the same type, *then* that group changes from one specified type to another. Conditional statements are often represented in programming languages using *if-then-else* statements, which allow certain instructions to happen only if a given decision/question has a *true* value.

Country of Original Author

Kosovo

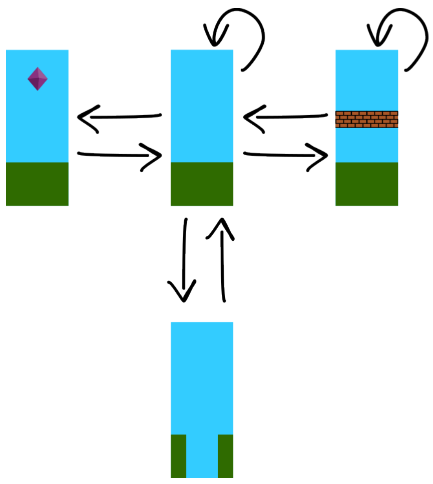
Part B

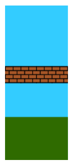
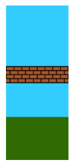
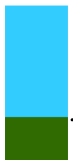
Superbebras

Story

In the computer game Superbebras, the background and the illusion of motion is created using a sequence of tiles.

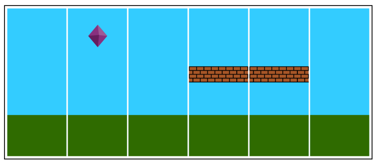
Tiles added to the right end of the sequence are chosen according to the rules in the diagram below. Arrows point directly from each tile to the only tile(s) that can be added immediately to the right of it.

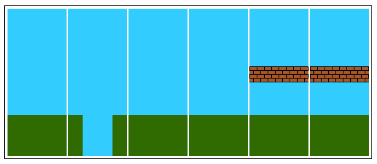


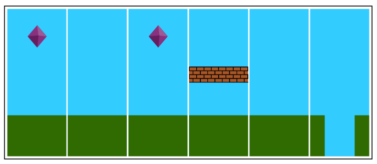
For example, a tile immediately to the right of the tile  can only be tile  or .

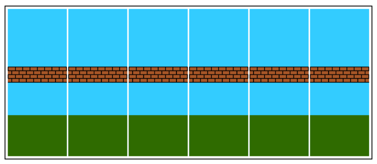
Question

Which of the following images is **not** a possible Superbebras background?

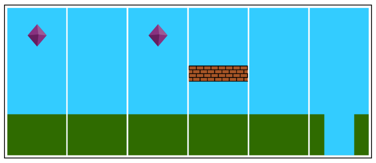
(A) 

(C) 

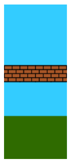
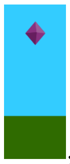
(B) 

(D) 

Answer

(B) 

Explanation of Answer

In Option B, the tile  is to the right of the tile , but there is not an arrow pointing between them in the diagram describing the rules (in either direction). Therefore, Option B is not a possible Superbebras background.

Options A, C, and D are all possible Superbebras backgrounds because they do follow the rules described by the diagram.

Connections to Computer Science

Some computer games, such as *endless runner games*, have a background that scrolls horizontally, creating the illusion that the player moves through a fantasy game world. Sometimes, the background is not a constant image, but is automatically created by the computer. This automatic generation is called *procedural generation*: usually smaller elements are randomly combined to create a huge variety of backgrounds. However, the combination of elements cannot be completely random: the elements must obey certain rules to avoid situations the player cannot cope with.

In this task, such rules are represented by a diagram made of tiles and arrows. This diagram is an example of a *directed graph*, which is a collection of *nodes*, together with *directed edges* that indicate which nodes can follow each other node. Searching through a directed graph graph can be done by performing a *breadth-first search (BFS)*, which is one way to determine which sequence of backgrounds is not possible.

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

Germany

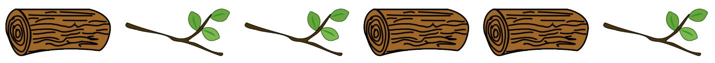
Logs and Branches

Story

Sequences of logs and branches are represented by codes, using the following two steps.

1. Moving from left to right in a sequence, each log or branch is represented by the number of branches to the right of it in the sequence. This gives a sequence of numbers.
2. Then each even number in this sequence is replaced with a 0 and each odd number with a 1, to obtain its code.

Let's look at the following sequence as an example.

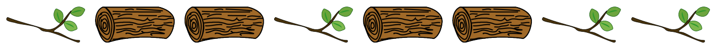


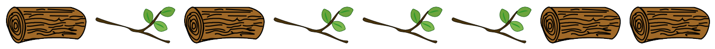
- The leftmost log has 3 branches to the right of it, so will be represented with a 3.
- The next branch has 2 branches to the right of it, so will be represented with a 2.
- The next branch has 1 branch to the right of it, so will be represented with a 1.

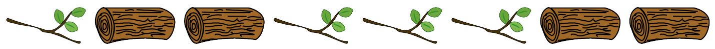
Continuing this process gives the sequence 3, 2, 1, 1, 1, 0. Its code is **101110**.

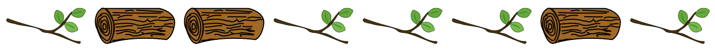
Question

Which of the following sequences of logs and branches is represented by the code **11101000**?

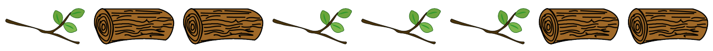
(A) 

(B) 

(C) 

(D) 

Answer

(C) 

Explanation of Answer

In the codes, a 0 means there is an even number of branches to the right of that position in the sequence, and a 1 means there is an odd number of branches to the right of that position in the sequence. We can go through the given code digit by digit.

The first digit is a 1, which means there should be an odd number of branches to the right of the first position. The sequences in Options A and C both have 3 branches to the right of the first position, so the first digits of their codes would be 1. However the sequences in Options B and D have 4 branches to the right of the first position, so the first digits of their codes would be 0. Thus, the given code cannot represent the sequences in Options B or D. Now we can look only at the sequences in Options A and C.

The sequences in Options A and C are the same until the fifth position. The sequence in Option A has 2 branches to the right of the fifth position, so its code would have a 0 in the fifth position. The sequence in Option C has 1 branch to the right of the fifth position, so its code would have a 1 in the fifth position. The given code has a 1 in the fifth position, so it cannot represent the sequence in Option A.

If we create the code for the sequence in Option C, we obtain the sequence 3, 3, 3, 2, 1, 0, 0, 0, which has a code of **11101000**, as desired.

Connections to Computer Science

To represent information in a computer, it must be *encoded* in a way that the computer can understand. In this task, the proposed encoding does not store the exact number of logs and branches, but rather a sequence of 0s and 1s that is in a more *compressed* representation. Even with this compressed representation, the original information (the order of the logs and branches) is able to be recovered. Therefore, for each encoding process, both the *encoding algorithm* and *decoding algorithm* need to be known.

This task is an example of a *compression algorithm*. Compression algorithms reduce the size of a file, such as an image or a large text document, by reducing the number of *bits* (binary digits) used to represent the file information, without losing information. For example, the *PNG image format* uses a combination of compression algorithms in order to reduce the size of the file, which reduces the time to load or store the image.

Country of Original Author

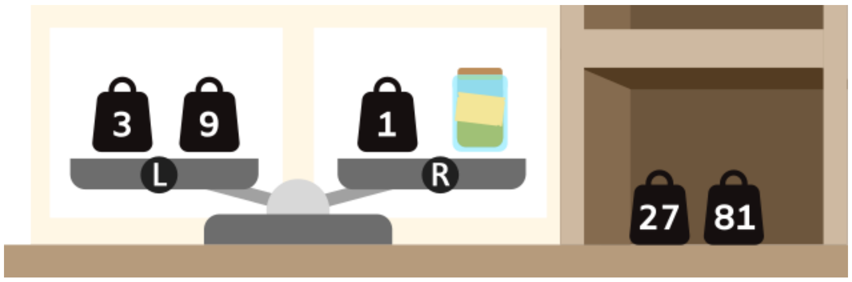
Bulgaria

Spices

Story

Genaro uses his balance scale to weigh the spices he sells. He uses only the following 5 weights on his scale: 1 gram, 3 grams, 9 grams, 27 grams, and 81 grams.

Genaro always puts the spices on the right side of the scale. To measure 11 grams of spices, he places his weights as shown.



To help train new employees, Genaro creates codes for different amounts of spices. In his code, L means the weight is placed on the left side, R means the weight is placed on the right side, and O means the weight is placed off the scale. Genaro writes each code in order from the smallest weight to the largest weight. For example, the code for 11 grams of spices is $RLLOO$, as shown.

Weight (grams)	1	3	9	27	81
Position	R	L	L	O	O

Question

Among the following four codes, which measures the heaviest amount of spices?

(A) $LORLO$

(B) $RRRRL$

(C) $LRLLO$

(D) $OLRLO$

Answer

(B) *RRRRL*

Explanation of Answer

We look at each of the codes, to determine what they are used to measure.

For code *LORLO*, the total mass on the left side is $1 + 27 = 28$ grams. The total mass on the right side is 9 grams. Since $28 - 9 = 19$, the code from Option A is used to measure 19 grams.

For code *RRRRL*, the total mass on the left side is 81 grams. The total mass on the right side is $1 + 3 + 9 + 27 = 40$ grams. Since $81 - 40 = 41$, the code from Option B is used to measure 41 grams.

For code *LRLLO*, the total mass on the left side is $1 + 9 + 27 = 37$ grams. The total mass on the right side is 3 grams. Since $37 - 3 = 34$, the code from Option C is used to measure 34 grams.

For code *OLRLO*, the total mass on the left side is $3 + 27 = 30$ grams. The total mass on the right side is 9 grams. Since $30 - 9 = 21$, the code from Option D is used to measure 21 grams.

Among these, 41 is the heaviest amount of spices, which is measured using the code from Option B.

Connections to Computer Science

The code created by Genaro is a variation of representing numbers in base-3, called *ternary*. Although base-3 is not very common in computer science, understanding how it works will help in understanding numbers in base-2, called *binary numbers*, which is how information is stored on a computer.

This task uses a variant of ternary numbers, called *balanced ternary*. Balanced ternary representation is used in applications where elements can be in one of three states: typically positive, negative, and zero/null. Some examples of these applications include electric devices, mathematical structures, and even ranking of sports teams, when the results of the game are wins (+1), draws (0) and losses (-1).

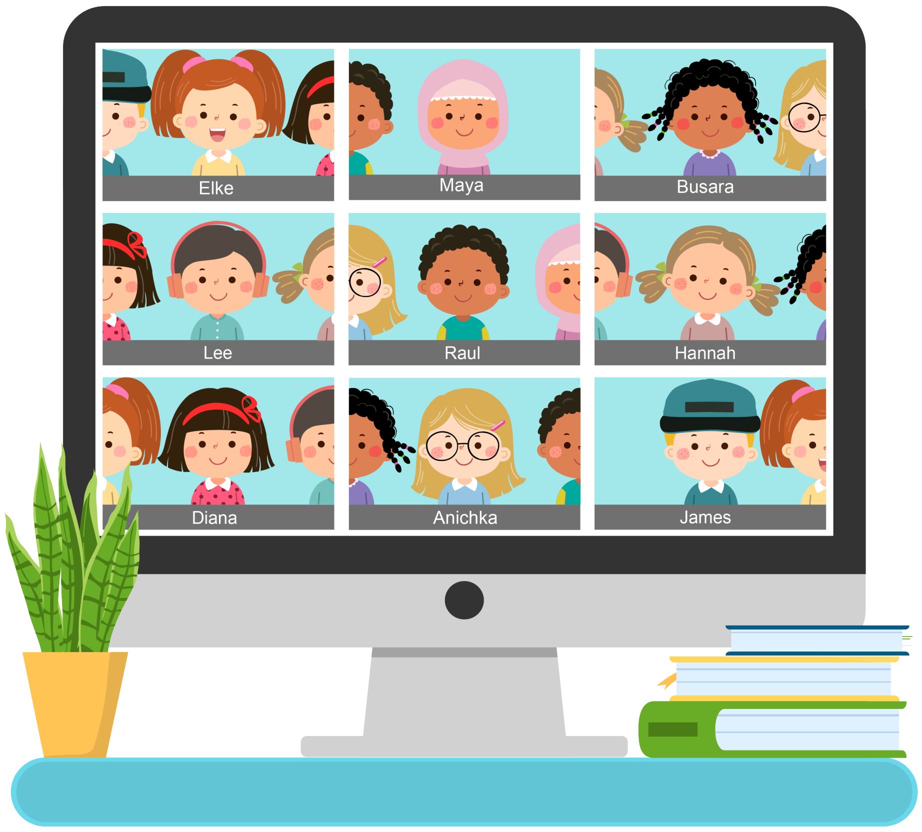
Country of Original Author

Brazil

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

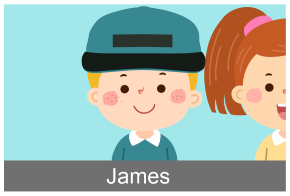
Answer

(D) Hannah

Explanation of Answer

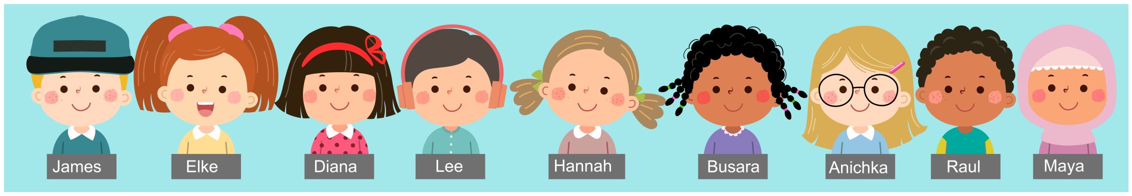
Based on the what can be seen on the teacher's screen, we can determine who each student sits next to.

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.

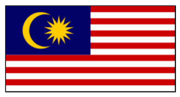
Connections to Computer Science

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

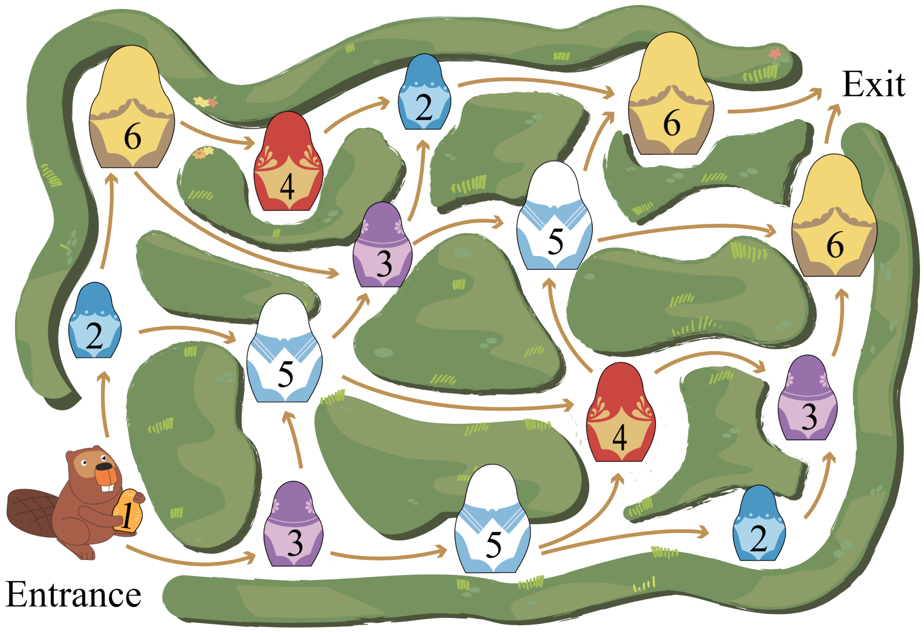
Malaysia



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

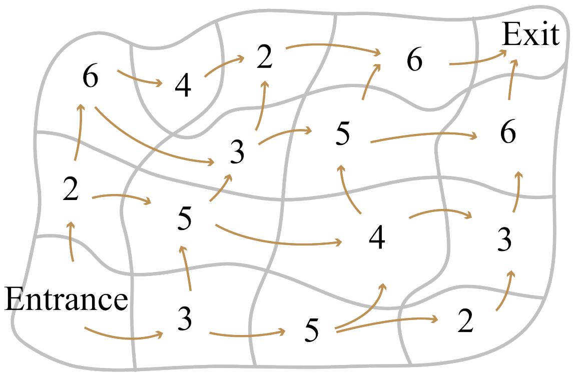
(D) 6

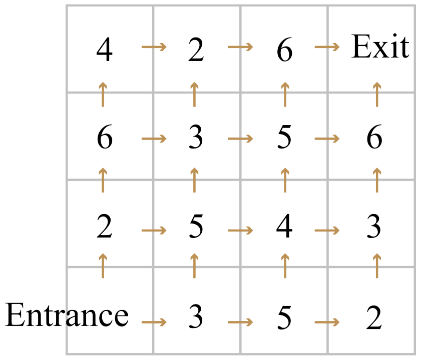
Answer

(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.

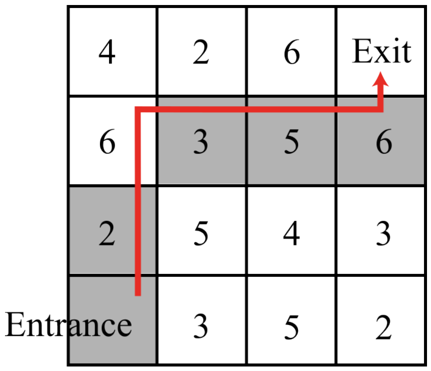


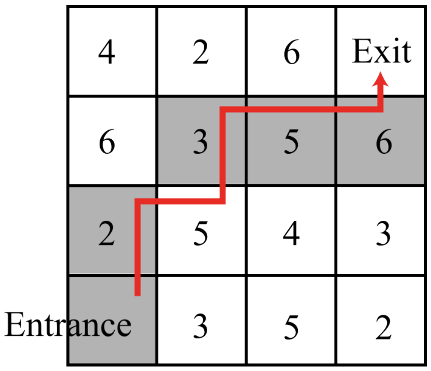


All paths from the Entrance to the Exit are 6 steps long so the maximum number of dolls Deana can carry is 6. By the rule and because the biggest doll is of size 6, if she collects 6 dolls, they must be of sizes 1, 2, 3, 4, 5, and 6, and picked up in that sequence.

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

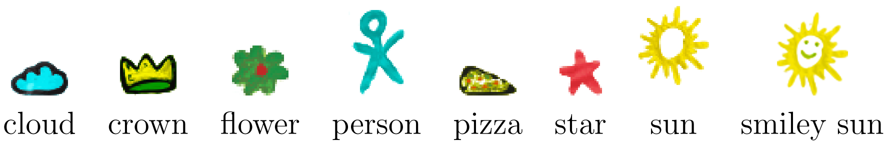
Taiwan

Part C

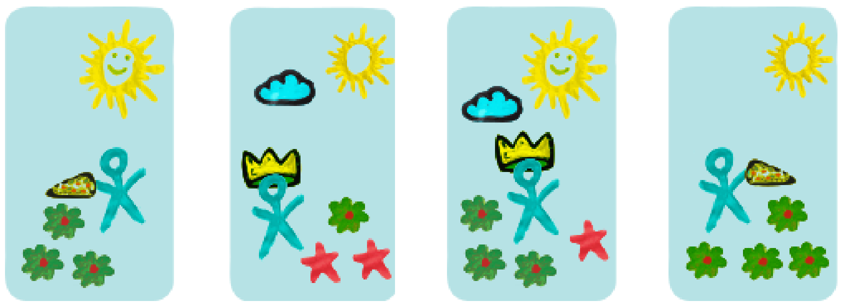
Card Art

Story

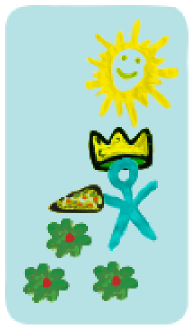
Manas creates drawings by combining some of the following images:



Manas draws on cards following one, single, secret rule. He has created four cards that obey this rule:



Anil observes the cards and creates a new card, but it does not obey the rule.



Question

Which of the following could be the secret rule?

(A) If there is a cloud on the card, then there are no flowers on the card.

(B) There must either be a star on the card, or there is a person holding pizza on the card.

(C) If there is a person holding pizza on the card, then there is no crown on the card.

(D) There must either be a smiley sun on the card, or a cloud on the card.

Answer

(C) If there is a person holding pizza on the card, then there is no crown on the card.

Explanation of Answer

Option A is incorrect because there is a cloud on Manas' second (and third) card but also flowers on Manas' second (and third) card.

Option B is incorrect because Anil's card has a person holding pizza on it which means it does obey the secret rule.

Option D is incorrect because Manas' fourth card has neither a smiley sun nor a cloud on it.

Option C is correct because all four of Manas' cards obey this rule and Anil's card does not. Specifically, Manas' first and fourth cards have a person holding pizza and not a crown, and their second and third cards do not have a person holding pizza. Anil's card has a person holding a pizza on it, but also a crown.

Connections to Computer Science

This task involves *logical reasoning*. Specifically, the key component of the task is reasoning about the cards and the possible rules that involve logical *OR* expressions, logical *NOT* expressions, as well as *conditional logical inference*.

Some of the possible secret rules involve the *logical connective* OR. For example, rule (B) states that there must be a star on the card OR there is a person holding a pizza on the card. In order for an OR expression to be *true*, at least one of the two component expressions must be true.

Other rules, such as rule (A), state that in some case, there must *NOT* be any flowers on the card. In order for the NOT (or *negation*) of an expression to be true, that expression must be *false*.

Finally, conditional logical inference appears, for example, in rule (C), which states that *IF* there is a person holding pizza on the card, *THEN* there is no crown on the card. Conditional logical inference is used frequently in many *programming languages* which contain *if-then-else* statements.

Logic plays a key role in computer science in a huge variety of areas: databases, programming languages, artificial intelligence, hardware and software design and verification, etc. Logic is one of the fundamental concepts that underlies *computational thinking*: being able to take information, model it logically, and make *logical conclusions* from that model.

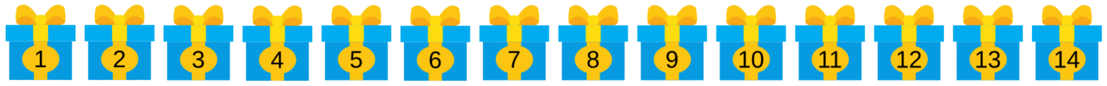
Country of Original Author

India

Gifts

Story

Bernard has wrapped gifts for the 14 students in his class, and numbered the gifts from 1 to 14.



Bernard knows the weight of each gift. After wrapping each gift, Bernard realized that he accidentally dropped his phone in one of the boxes. In order to avoid unwrapping many gifts, he plans to do the following:

1. Divide the 14 gifts into 2 piles, with the smallest 7 gift numbers in the first pile and the largest 7 gift numbers in the second pile.
2. Weigh the first pile. If it is heavier than expected, retain the first pile. Otherwise retain the second pile.
3. Divide the retained pile into 2 piles containing as close to the same number of gifts as possible. If there is an odd number of gifts, the first pile will have one fewer gift than the second pile. The smallest gift numbers are in the first pile.
4. Repeat Steps 2 and 3 until the retained pile has a single gift that contains the phone. Open that gift to retrieve the phone.

Question

In which of the following situations would Bernard weigh the fewest number of piles?

(A) The phone is in gift number 3.

(B) The phone is in gift number 13.

(C) The phone is in gift number 8.

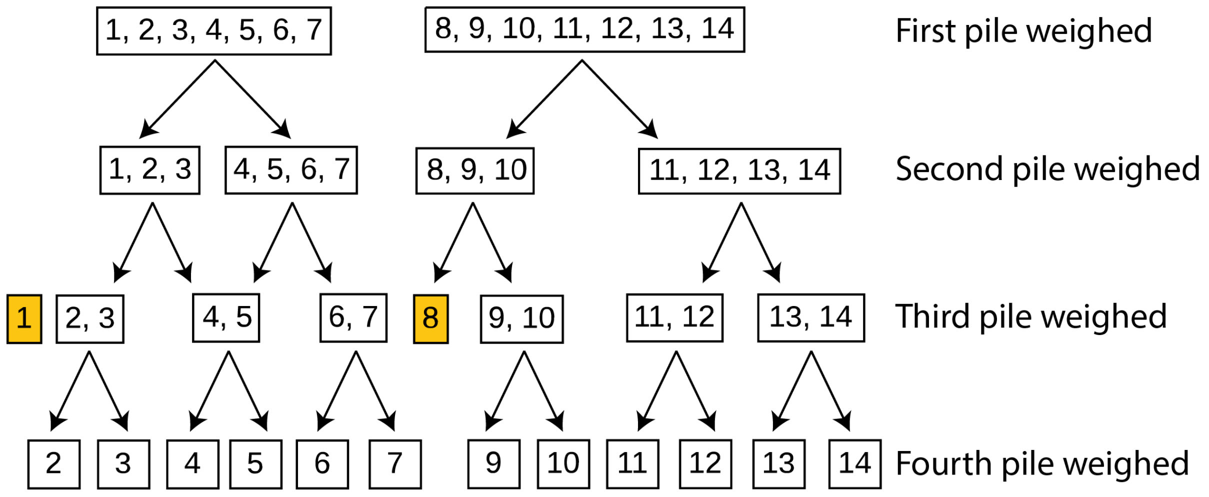
(D) The phone is in gift number 6.

Answer

(C) The phone is in gift number 8.

Explanation of Answer

We look at all the possibilities for the retained piles. These are illustrated in the following diagram, where each pile is indicated by a box.



Notice that if gifts 1 or 8 contained the phone, it could be found after weighing exactly three piles. In all other cases we must weigh four piles. Thus, of the options given, the situation that requires the fewest number of piles to be weighed is if the phone is in gift number 8.

Connections to Computer Science

This task highlights a problem solving technique called *divide and conquer*. In computer science, a divide and conquer approach to solving a problem is to repeatedly divide a problem into several *non-overlapping subproblems*, solve the subproblems, and combine the subproblem solutions.

In this task, notice that every time the problem is divided, we can remove one of the subproblems: that is, approximately one half of the remaining boxes are no longer under consideration. This approach to solving this task is a variant of *binary search*, which searches for a value in a *sorted list of values* and is exponentially faster than searching linearly through that list of values.

Country of Original Author

Romania

Beaver Travels

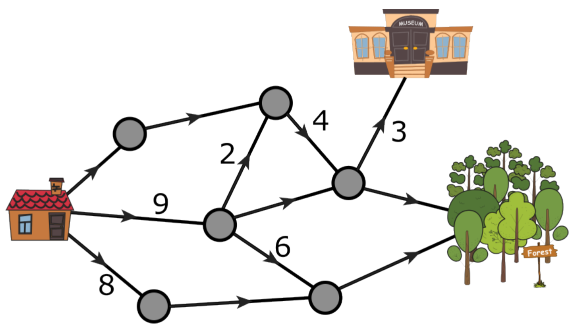
Story

A group of beavers left their house .

Some of them went to the museum , and the rest of them went to the forest .

Beavers documented their various travels in the map below. Each circle is a place where beavers could pass through, and the lines between them are the paths they could take. Each path is labelled with an arrow indicating the direction the beavers took on that path, and also a number indicating the number of beavers that took that path. For example, three beavers took the path to the museum.

Unfortunately, some documentation is missing, so some paths are missing their label.



Question

How many beavers went to the forest?

(A) 16

(B) 11

(C) 18

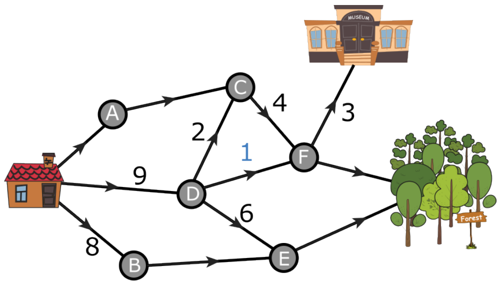
(D) 14

Answer

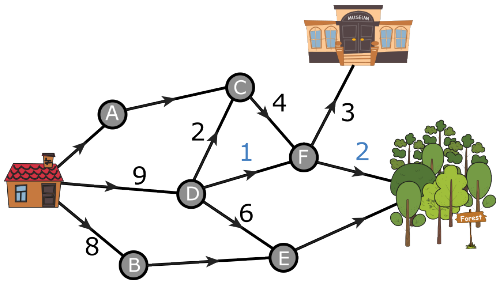
(A) 16

Explanation of Answer

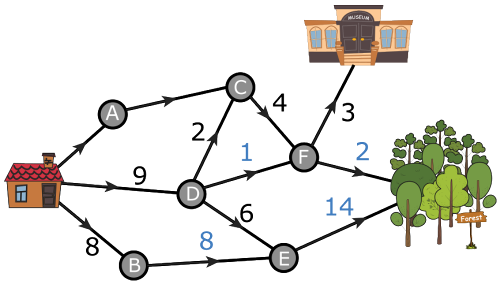
We begin by labelling the intersections as shown below. The beavers that went to the forest must have gone through F or E. The only way to F is from C or D. We know that 4 beavers arrived from C, but we do not know how many arrived from D. Therefore, we consider the paths connected to D. From the house, there are 9 beavers that went to D. This means there are also nine beavers that left D. This can happen either on the path to C (2 beavers), to E (6 beavers) or to F. Therefore, only 1 beaver went from D to F path because $9 - (2 + 6) = 1$. This is what we know now:



Next, we consider the path from F to the forest. There are $4 + 1 = 5$ beavers that went to F: 4 from C and 1 from D. Since 3 beavers went to the museum from F, this means that 2 beavers went to the forest because $5 - 3 = 2$. We can use this to update what we know at this point:



We still need to find out how many beavers went to the forest from E. There are 6 beavers that went from D to E. And, because 8 beavers went to B from the house and there is only one path leaving B, 8 beavers also went from B to E. In total, $6 + 8 = 14$ beavers went to E. There is also only one path out of E so all 14 of these beavers went from E to the forest. We now have all the information we need:



Summing up the numbers of beavers that went from F to the forest and from E to the forest, we determine that $2 + 14 = 16$ beavers went into to the forest.

Connections to Computer Science

This task involves a concept from *graph theory* called a *weighted directed graph*.

A *directed graph* is a collection of *nodes* (or *vertices*) connected by *directed edges*. For example, if u and v are two different places on a map, a directed edge may go from u to v to indicate that it is possible to travel from u to v. In a *weighted graph*, each edge has a numerical weight attached to it. This weight can represent a variety of quantities: the length of a road, the maximum capacity of a river system, the toll cost for a highway, or the frequency of traversal along that path. In this task, the numbers on the directed edges indicate how many times a beaver has travelled along that path.

Additionally, this task utilizes the concepts of *in-degree* and *out-degree* of a vertex. The in-degree refers to the sum of weights of inbound edges to a node, while out-degree refers to the sum of weights of outbound edges from a node.

There are many applications of weighted directed graphs: determining the least expensive flight path between two cities, the quickest path in the internet to send an email message, or how to transport electricity through an electrical grid with the least amount of wasted energy.

Country of Original Author

Slovakia

Seashell Game

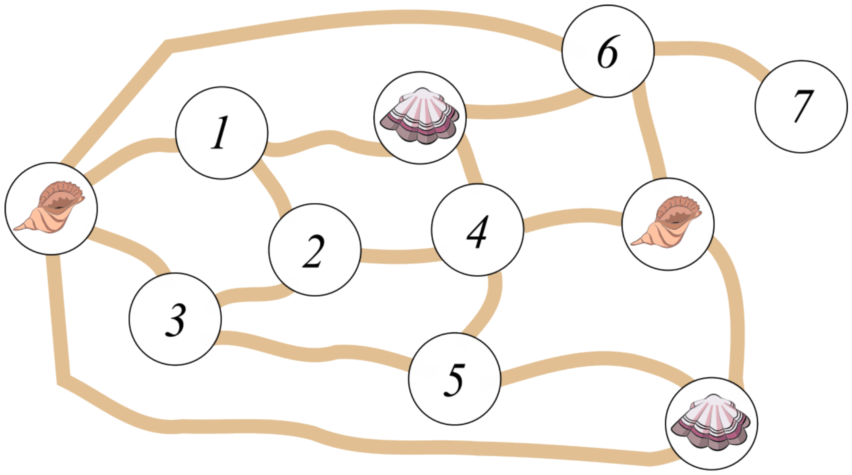
Story

Quinn and Evan are playing a game on the beach involving shells, holes, and lines in the sand.

To play the game, they take turns placing new shells in empty holes, some of which are connected by lines. The first person to have two of their shells placed in holes directly connected by a line loses the game.

Quinn plays with one type of shell: , and Evan plays with another type of shell: .

The game has started and each player has completed two turns. The placement of these four shells are as shown, and the remaining empty holes are numbered 1 through 7.



Question

It's now Quinn's turn. In which empty hole should Quinn place her shell if she wants to guarantee a win in the game?

(A) 1

(B) 2

(C) 5

(D) 7

Answer

(D) 7

Explanation of Answer

Quinn will lose immediately if she places a shell in holes 1, 3, 4, or 6. This means she is left to choose hole 2, 5, or 7 if she wants a chance to win.

If Quinn places her next shell in hole 2, then Evan might place a shell in hole 7. At that point, Quinn must place a shell in hole 5 to avoid losing immediately. Evan can follow by placing a shell in hole 3, and Quinn will immediately lose on her next turn.

Similarly, if Quinn places her next shell in hole 5, then Evan might place a shell in hole 7. At that point, Quinn must place a shell in hole 2 to avoid losing immediately. Evan can follow by placing a shell in hole 3 and Quinn will immediately lose on her next turn.

Instead, suppose Quinn places a shell in hole 7. Then Evan must place a shell in hole 1, 2, 3, 4, 5, or 6. He will lose immediately if he places a shell in 1, 4, 5, or 6. If he places a hole in 2 or 3, then Quinn can place a shell in hole 5. At that point, Evan will immediately lose when he places his next shell. Therefore Quinn is guaranteed a way to win the game by placing a shell in hole 7.

Connections to Computer Science

Games such as this are studied by *game theorists*. In particular, the discovery and implementation of winning strategies such as the one we discovered for Quinn is a key concept of *algorithmic game theory*.

Game theory has applications to economics, biology, and computer science. Specifically, current problems in computer science are being solved using game theory. These problems can arise anytime there is some sort of “competition”. In economics this could be competing companies. In biology, different species of plants compete for sunlight and different species of animals compete for food.

More technically, in the field of *artificial intelligence*, systems often consist of *agents* that behave like players in a game. Classical algorithms such as *minimax* and *negamax* can be used to find winning strategies and more modern *machine learning* algorithms can solve more complex problems involving large quantities of data using game-theory techniques.

Another application of game theory arises in *network security*. The interaction between an *attacker* and the *network administrator* can be modelled as an *attack-and-defense* game.

Country of Original Author

Canada

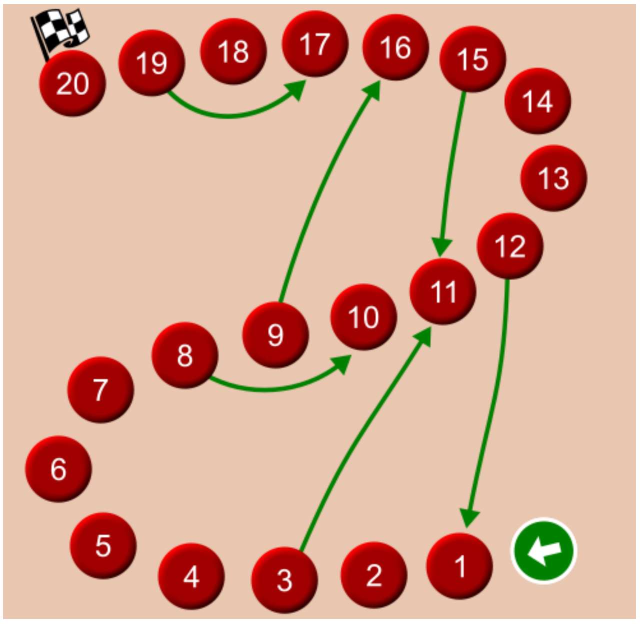
Disappearing Arrows

Story

Fleas A, B, C, and D start a race, in that order, from position 1 in the picture shown. They continue to take turns in this order $(A, B, C, D, A, B, \ldots)$ and follow two rules:

1. On their turn, a flea will jump one position forward.
2. Arrows provide a one time short cut. If a flea jumps to a position that has an arrow leading from it, it immediately jumps to the position the arrow is pointing to. The arrow then disappears so that no other flea can use that arrow.

It is possible for more than one flea to be at the same position at the same time.



Question

Which flea will finish first?

(A) A

(B) B

(C) C

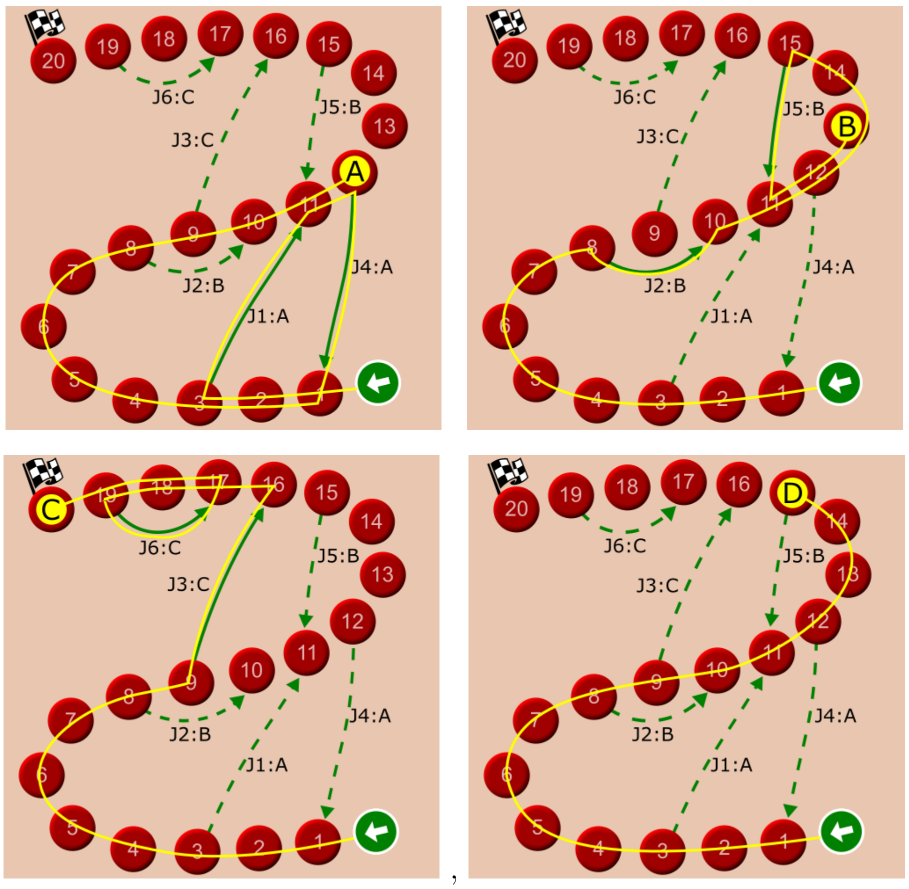
(D) D

Answer

(C) C

Explanation of Answer

A scheme for the route of each flea is shown from which we can see that flea C finishes first. Yellow circles represent the positions of individual fleas A, B, C, and D after the 15th turn (when flea C reaches the final position). Yellow lines show their routes. The jumping arrows are numbered as J1 to J6 followed by A, B, C, or D indicating the flea using the arrow, after which the arrow disappears. A solid arrow indicates an arrow available for use. A dashed arrow means that it has disappeared.



,

A few observations can make it more clear why C is the winner:

- Since flea A is the first to move, it comes to arrow J1 first and uses it. At the next turn it must use arrow J4 causing it to return to the 1st position, more than two positions behind the others.
- Flea B is the first one that comes to the arrow J2 so it skips past arrow J3. This means that the first flea reaching arrow J3 is flea C.
- Using the arrow J3, flea C moves to the 16th position moving well past the other fleas. Then, even though it goes back 2 positions from the 19th position, it still retains the lead.
- Flea D doesn't use any of arrows so it cannot overtake flea C.

Connections to Computer Science

Computers can *simulate* many phenomena and processes. For example, a competitive struggle between various opponents, where one player may gain a short-term advantage or, conversely, a short-term disadvantage, depending which decisions were made. If the phenomena can be *modelled* and changes *modelled* with an *algorithm*, a computer will be able to predict and simulate the result.

This task is an example of a *dynamic system*, where the *state* of the system changes when flea reaches a position with a green arrow. This operation of updating a state is one of the fundamental operations of the *central processing unit (CPU)*: given the current state and some input, what new state should the computer be in?

A very well-known case of evolution simulation is the *fox-rabbit population problem*. It predicts the development of the rabbit and fox population on an isolated island. Foxes eat rabbits, and when they eat a lot of them, they run out of food, starve, and die too. Fewer foxes will then allow rabbits to multiply faster.

A computer can be a very helpful tool to model, analyze, and predict outcomes based on changes to the system. The fox-rabbit population problem can be visualized using the following website:

`https://www.geogebra.org/m/FwQfAxqE`

Many dynamic systems, such as weather systems or stock markets, can be modelled and analyzed using computer models.

Country of Original Author

Czechia