UNIVERSITY OF WATERLOO

The CENTRE for EDUCATION in MATHEMATICS and COMPUTING

2024 Beaver Computing Challenge (Grades 9 & 10)

Questions, Answers, Explanations, and Connections

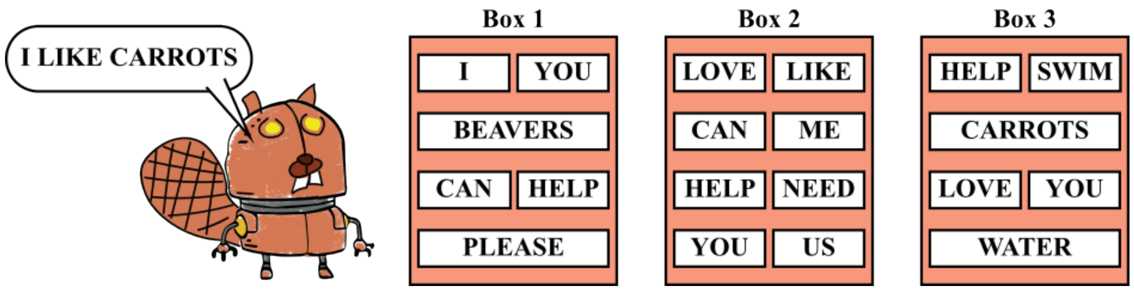
Part A

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.



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.

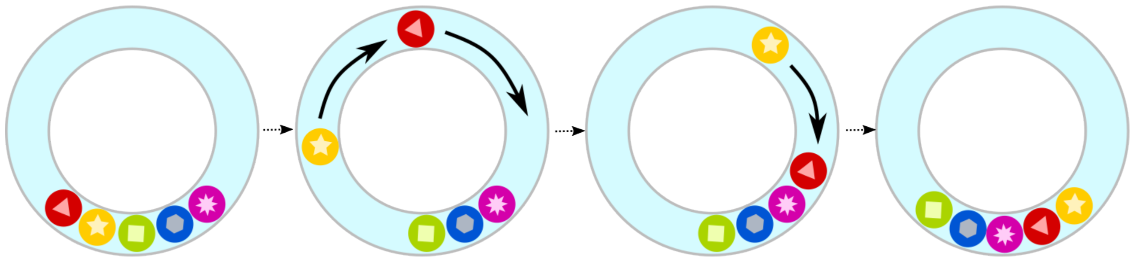
Country of Original Author

Vietnam

Rattle

Story

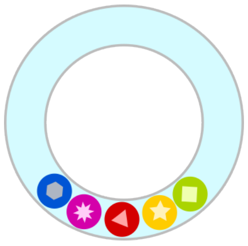
A baby's rattle is made of a clear circular tube with colourful balls inside. When the baby shakes the rattle, some balls can move through the tube. An example of the rattle before and after it is shaken is shown.

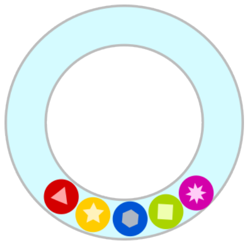


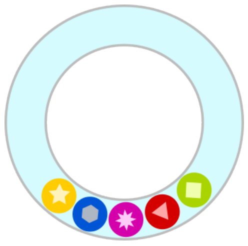
While there is room for the balls to move, there isn't room for one ball to move past another ball.

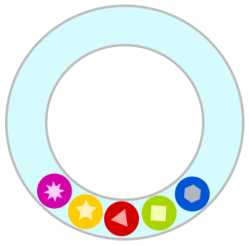
Question

Which of the following could be the baby's rattle after it has been shaken again?

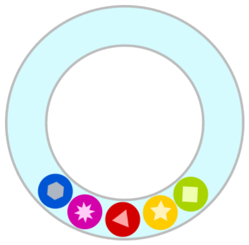
(A) 

(B) 

(C) 

(D) 

Answer

(A) 

Explanation of Answer

Option A could be the baby's rattle if the given example continues with the green ball moving to the other side.

Options B and C are not possible. They have the blue ball beside the yellow ball, but the two balls beside the yellow ball in the original rattle are the red ball and the green ball.

Option D is not possible. It has the red ball beside the green ball, but the two balls beside the green ball in the original rattle are the yellow ball and the blue ball.

Connections to Computer Science

In this task, Oliver's rattle illustrates the *data structure* called a *circular linked list*. In a circular linked list, each element has a *pointer* to the next element, and the pointer of the last element in the list points to the first element, thereby forming a circle. A circular linked list can store any type of element. In this task, the items stored are "colourful balls", but numbers, names, or other types of elements could also be stored at each position.

The usefulness of the circular linked list is that the relative order of the elements remains the same. When we check the order of the balls, we can start from any ball, and simply go through all of them (the whole list) in the correct order.

There are many applications of circular linked lists in everyday life. One such area is music playlists. For example, in many music streaming applications, a playlist can be "on repeat": once all of the songs from the playlist have been played, the music would continue from the first song in the playlist.

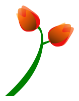
Another application of circular linked lists is in *task scheduling*: given a set of tasks that require one shared resource, such as a *CPU*, one equitable algorithm is to give each task a fixed amount of time on the CPU, and then move onto the next task. Since there is no "end" to the list, there will always be a task that will be using the CPU, allowing for complete utilization of the shared resource.

Country of Original Author

Slovakia

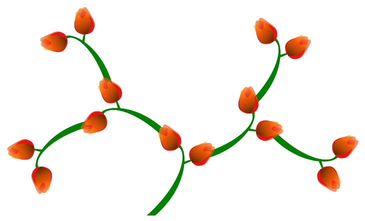
Miracle Plant

Story

A miracle plant seedling has a stem and two buds. It looks like this: 

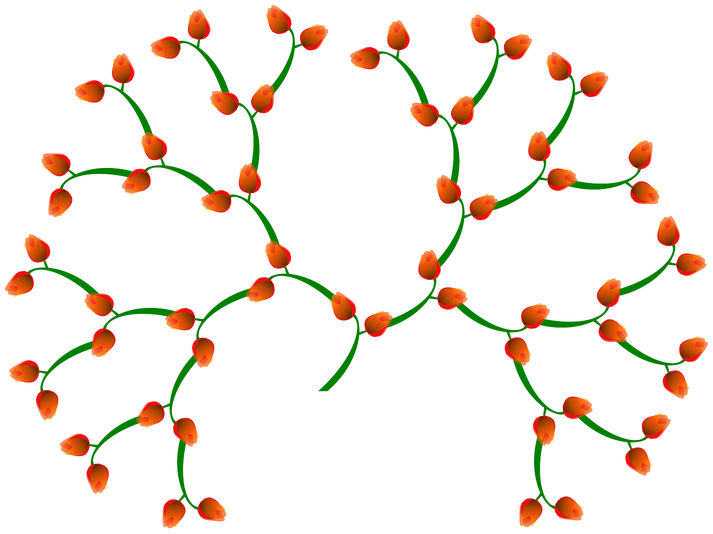
It takes one day for a bud to sprout a new stem and two new buds.

For example, two days after a seedling is planted, the miracle plant looks like this:



Question

Over time, the miracle plant becomes quite large and magnificent. How many days ago was the seedling planted?



(A) 4

(B) 6

(C) 8

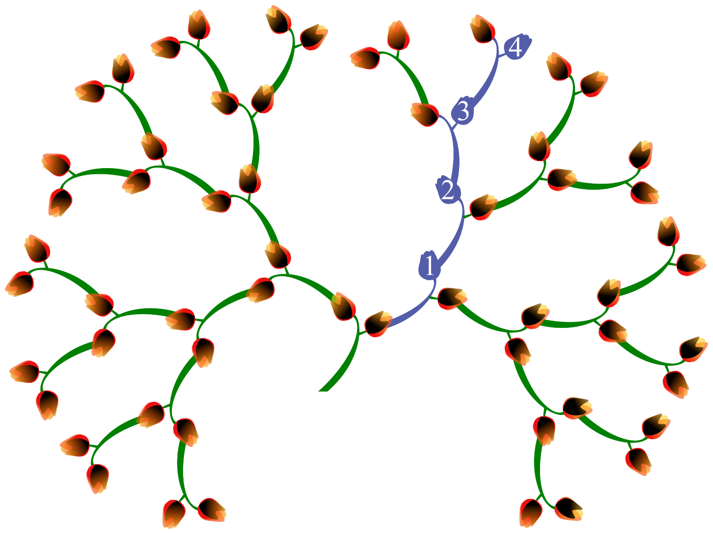
(D) 16

Answer

(A) 4

Explanation of Answer

The initial miracle plant with two buds is in the middle of the picture. If we start at one of these two buds and follow one path of stems and buds to the outer edge of the plant, the number of stems we encounter is equal to the number of days the plant has been growing. This number will also equal the number of buds we encounter. By either count, we can see that the miracle plant seedling was planted four days ago.



Connections to Computer Science

This task focusses on a *recursive definition* of a *data structure* known as a *binary tree*.

A recursive definition is where an object or structure is defined in terms of smaller instances of itself, which is the *recursive case*, ensuring that at some point, there is a smallest such object/structure, called the *base case*. For example, the structure in this task can be formed by a bud sprouting a steam and two new buds (the recursive case) or by simply being a single bud (the base case).

Binary trees are used extensively in computer science. They are used to implement *binary search trees* that allow the *searching* for an element in a collection to be done exponentially faster than a *linear search*, which examines each element in the collection. Binary search trees, especially *height-balanced binary search trees* are used in *information retrieval*, especially *database systems*.

Country of Original Author

Germany

Typos

Story

Luisa is typing a list of countries that border Argentina but her custom keyboard has malfunctioned. For most keys, when she presses the key, the character on the key to its right is displayed instead. The one exception is that when she presses the rightmost key in a row, the character on the leftmost key in that row is displayed.

For example, if she presses D, then F is displayed. If she types P, then # is displayed.

@	Q	W	E	R	T	M	O
#	A	S	D	F	G	I	P
Z	X	C	V	B	K	Y	$
Ñ	H	N	U	J	L	.	

Question

If Luisa wants to press keys so that BRAZIL is displayed, what keys should she press?

(A) V E # X G J

(B) V E # $ J G

(C) V P S X T K

(D) V E # $ G J

Answer

(D) V E # $ G J

Explanation of Answer

The characters B, R, A, I and L do not appear on a key at the beginning of a row, so to display any of these characters, Luisa needs to press the key to the left of the key containing that character. That is, she needs to press V, E, #, G and J to display B, R, A, I and L.

The character Z appears at the beginning of a row so to display Z, the key at the end of the row must be pressed. That character is $.

Therefore, in order for BRAZIL to be displayed, Luisa must press the keys shown in Option D.

Connections to Computer Science

When a user interacts with a computer, there are usually three distinct stages: *input*, *processing*, and *output*. Usually, input to a computer is given by a keyboard, though there can be touchscreens, microphones, mice or other devices that send signals, interpreted as electrical pulses. These electrical pulses, which can be thought of as a sequences of 0's and 1's, which are *binary numbers*, are processed by the CPU to cause some actions to happen. The final phase of output is usually done on a computer monitor, a printer, a speaker, or other such devices that capture that output in some way. The entire cycle then can repeat, since the output may influence what the next input is.

One more modern *preprocessing technique* is to transform the input before it is processed. For example, many applications have *autocorrect* features that fix commonly occurring typographical errors. For instance, if the user types the word `thier`, many applications would transform that into the word `their`, using knowledge from a data set of common errors, and also a *predictive* algorithm to determine the most-likely intended word, or even phrase. Both of these techniques use aspects of *data science* and *machine learning* from a large collection of information.

Country of Original Author

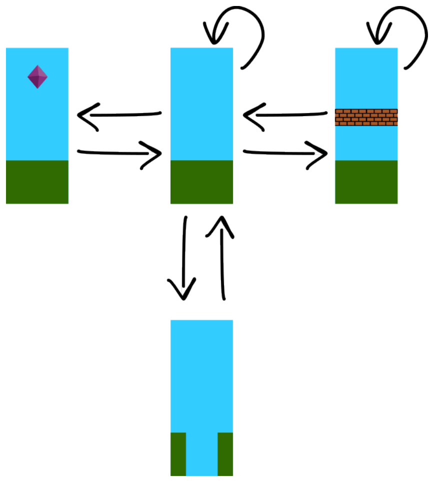
Argentina

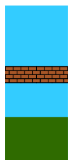
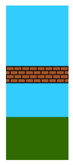
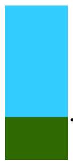
Superbebras

Story

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

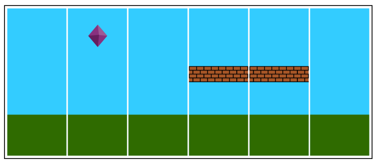
Tiles added to the right end of the sequence are chosen according to the rules in the diagram below. Arrows point directly from each tile to the only tile(s) that can be added immediately to the right of it.

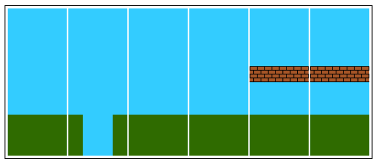


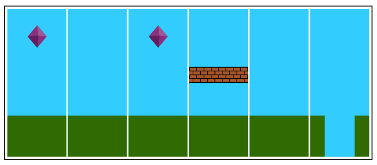
For example, a tile immediately to the right of the tile  can only be tile  or .

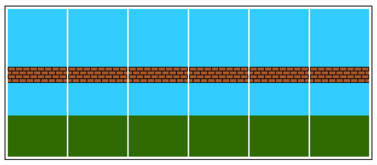
Question

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

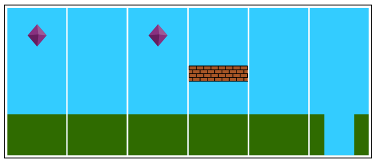
(A) 

(C) 

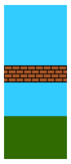
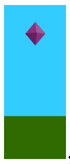
(B) 

(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

Part B

Book Return

Story

At the Hardwood Library, there is usually a long line of beavers waiting for the librarian to help them return their books. At this particular library, book returns are processed by quantity. That is, the beaver with the fewest books to return is helped first.

The librarian can process one book per minute. After processing all of the books returned by a particular beaver, the librarian next helps the beaver waiting in line with the fewest books to return.

One morning, five beavers come to the library to return their books. Their time of arrival and their number of books are shown in the table:

Name	Time of Arrival	Number of Books
Luciano	9:00	4
Rui	9:02	6
Cene	9:03	3
Manuel	9:05	4
Priscilla	9:12	1

Luciano arrives right when the library opens, so the librarian processes Luciano's books first.

Question

In which order will the beavers be helped?

(A) Luciano, Rui, Cene, Manuel, Priscilla

(B) Luciano, Cene, Rui, Manuel, Priscilla

(C) Luciano, Cene, Manuel, Rui, Priscilla

(D) Priscilla, Cene, Luciano, Manuel, Rui

Answer

(C) Luciano, Cene, Manuel, Rui, Priscilla

Explanation of Answer

Luciano is the first to arrive at the library. As they are the only one in line, the librarian helps them right away. Since they have 4 books to return, they finish returning books at 9:04.

At 9:04 there are 2 beavers in line: Rui who arrived at 9:02 with 6 books to return and Cene who arrived at 9:03 with 3 books to return. Since Cene has fewer books to return than Rui, the librarian helps Cene next and they finish returning books at 9:07.

At 9:07 there are 2 beavers in the line: Rui with 6 books to return and Manuel, who arrived at 9:05 with 4 books to return. Since Manuel has fewer books to return than Rui, the librarian helps Manuel next and he finishes returning books at 9:11.

At 9:11 only Rui is in line, so he returns his books finishing at 9:17.

At 9:17 Priscilla, who arrived at 9:12, is the only one in the line so the librarian helps her right away. Since she has only 1 book to return, she finishes returning this book at 9:18.

This means the beavers are helped in the order Luciano, Cene, Manuel, Rui, Priscilla.

Connections to Computer Science

This problem highlights the operation of a *scheduler*, which is an *algorithm* that helps organize and coordinate how one resource can be shared amongst many users. For example, in an *operating system*, the scheduler manages which processes/applications can use the *central processing unit (CPU)*. A scheduler determines the order in which processes are executed, when, and for how long. There are a number of different scheduler policies, each with costs (e.g., additional computation needed) versus benefits (e.g., more equitable sharing of the resource). Four such example scheduling policies are shown below:

- First Come First Serve (FCFS): the process that is first in the queue is executed first, and executed in its entirety.
- Priority Scheduling: the process with the highest priority is executed first, and executed in its entirety.
- Round Robin: each process takes turns running for a certain amount of time. If a process is not yet fully executed, it returns to the end of the queue, and a new process is selected to execute.
- Shortest Job First (SJF): the process in the queue with the shortest estimated duration is executed first, and executed in its entirety.

In this problem, the Shortest Job First algorithm was used.

Country of Original Author

Slovenia

Magical Fruit

Story

A magician can transform one type of fruit into another. The magical transformations are denoted by **M**. The fruit that the transformation is applied to is put inside brackets and the fruit that the transformation produces is put on the right after an equals sign. Six magical transformations are shown in the table below. For example, the top-left entry in the table describes a magical transformation that can be applied to an 🍎 to produce a 🍐.

M(🍎) = 🍐	**M**(🍍) = 🍇
M(🍐) = 🍌	**M**(🍇) = 🍎
M(🍋‍🟩) = 🍍	**M**(🍌) = 🍋‍🟩

Transformations can be combined. For example, since we know **M**(🍎) = 🍐, we can write the following combined transformation:

M(**M**(🍎)) = **M**(🍐) = 🍌

Question

What fruit is produced by the following combined magical transformation?

M(**M**(**M**(**M**(**M**(🍌)))))

(A) 🍋‍🟩 (B) 🍐 (C) 🍎 (D) 🍇

Answer

(B) 🍐

Explanation of Answer

First we notice that this is a combination of five magical transformations. Then we can arrive at the answer step by step as shown below.

1. $\mathbf{M}\left(\mathbf{M}\left(\mathbf{M}\left(\mathbf{M}\left(\mathbf{M}\left(\text{🍌}\right)\right)\right)\right)\right) = \mathbf{M}\left(\mathbf{M}\left(\mathbf{M}\left(\mathbf{M}\left(\text{🍋‍🟩}\right)\right)\right)\right)$
2. $\mathbf{M}\left(\mathbf{M}\left(\mathbf{M}\left(\mathbf{M}\left(\text{🍋‍🟩}\right)\right)\right)\right) = \mathbf{M}\left(\mathbf{M}\left(\mathbf{M}\left(\text{🍍}\right)\right)\right)$
3. $\mathbf{M}\left(\mathbf{M}\left(\mathbf{M}\left(\text{🍍}\right)\right)\right) = \mathbf{M}\left(\mathbf{M}\left(\text{🍇}\right)\right)$
4. $\mathbf{M}\left(\mathbf{M}\left(\text{🍇}\right)\right) = \mathbf{M}\left(\text{🍎}\right)$
5. $\mathbf{M}\left(\text{🍎}\right) = \text{🍐}$

Therefore, following the series of equalities, we have $\mathbf{M}\left(\mathbf{M}\left(\mathbf{M}\left(\mathbf{M}\left(\mathbf{M}\left(\text{🍌}\right)\right)\right)\right)\right) = \text{🍐}$.

Connections to Computer Science

In many *programming languages*, a program can be divided up into more logical sections called *functions*. Each function can be thought of as a *subprogram*, which consumes some input through *parameters*, and produces some *output*. In this problem, the function is machine M, the input is a single fruit, and the output is a single fruit.

One advantage of a function is that it can be reused many times, even though it is only written once. For example, in this problem, it is applied five times in the uestion.

Additionally, functions can be *nested* to form a sequence of computations: the output of one function can be used as the input to another function. In this problem, nesting is heavily used: since the function M produces one (fruit) value, that value can be used as the input to another *application* of M.

The nesting of functions leads to the *programming paradigm* of *functional programming*. In a functional programming language, such as *OCaml* or *Haskell*, a program is a sequence of applying and nesting functions, rather than a sequence of *statements* in a *imperative programming paradigm*, which is predominantly used in the *Python* programming language.

Country of Original Author

Serbia

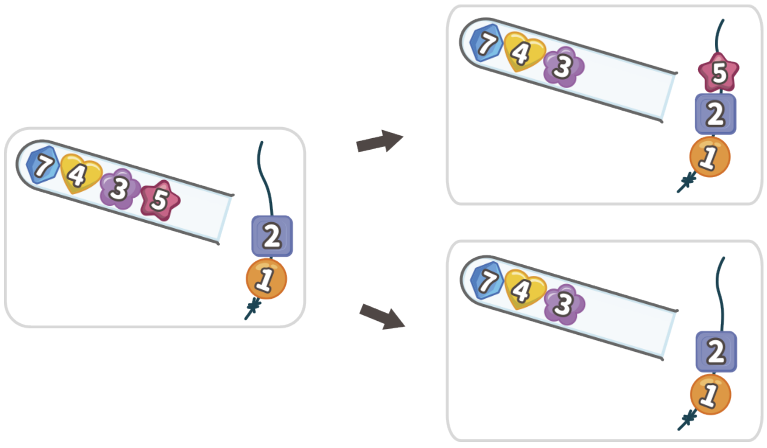
Bebracelet

Story

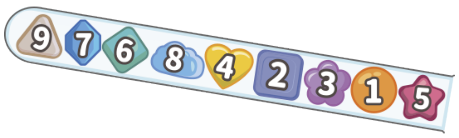
Rory is making Bebracelets. He takes numbered beads from a tube, one at a time. For each bead, he is allowed to put it on the string if:

- the string is empty, or
- the number on the bead is larger than the number on the last bead put on the string.

For each bead that Rory is allowed to put on the string, he then chooses whether to put it on the string or discard it. For example, if bead 2 is the last bead on the string, and bead 5 is the next bead from the tube, then Rory may put bead 5 on the string or discard it.



Now, Rory is making a new bracelet from the beads in this tube:



Question

What is the largest number of beads that Rory can put on the string?

(A) 4 beads

(B) 5 beads

(C) 6 beads

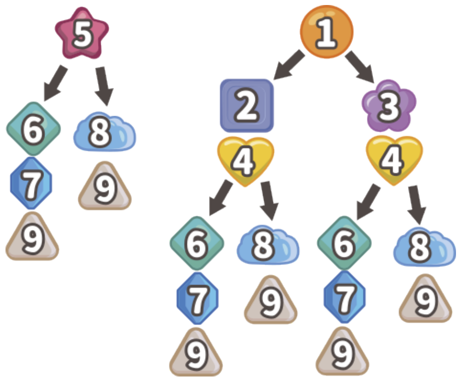
(D) 7 beads

Answer

(C) 6 beads

Explanation of Answer

Bead 5 is the first bead from the tube. The beads in the tube with larger numbers are beads 6, 7, 8, and 9. By inspecting the order of these beads in the tube, we see that the longest possible sequence that begins with 5 is 5679. The diagram below on the left shows sequences when bead 5 is used. Alternatively, if bead 5 is discarded, and the next bead (bead 1) is put on the string first, there are more and longer possible sequences. The diagram below on the right shows sequences that can be built when bead 5 is discarded.



If any other bead is put on the string first, the longest possible final sequence is already included in one of those above, starting with bead 1 or bead 5. For example, putting bead 2 on the string first leads to sequences such as 24679 or 2489, both of which are included in one of the longer sequences above (124679 and 12489, respectively). That is, a sequence that does not start with bead 1 (or bead 5), cannot be the longest sequence because it can always be made longer by using bead 1 first.

From this, the Bebracelets with the largest number of beads from the given tube start with bead 1 and consist of 6 beads each: 124679 and 134679.

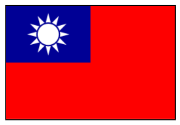
Connections to Computer Science

This problem asks us to find the longest increasing sequence of beads where some beads may be unused. This is an instance of the *longest increasing subsequence problem*, which 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.

To see the advantage of the dynamic programming solution, notice that to try every possible subsequence would mean that every element will either be in the subsequence or not. In other words, for a sequence of length n, there would be 2^n possible subsequences to test. Even for relatively small numbers, such as $n = 100$, this would take years to finish on most modern computers, since this algorithm would run in *exponential time* in terms of the input. However, the dynamic programming solution will run in time roughly proportional to $n \log n$: so even if the list had 1 000 000 elements, the longest increasing subsequence could be found in roughly 20 000 000 steps, which usually runs in less than one second on most modern computers.

Country of Original Author

Taiwan

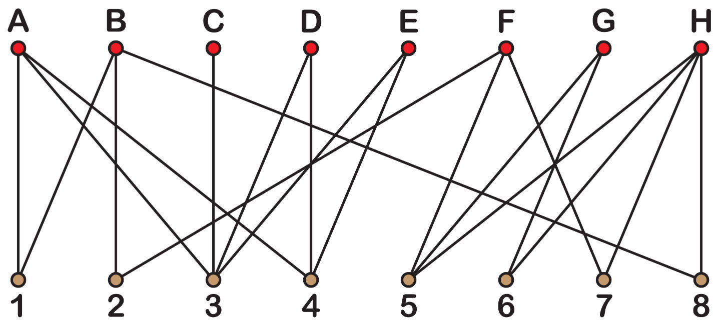


Hermit Crabs

Story

Hermit crabs wear shells in order to protect themselves. As hermit crabs grow, their shells become too small and the hermit crabs need to look for larger shells to wear.

An assortment of new shells has just been washed ashore. Several hermit crabs have gathered to try on these new shells. In the following diagram, the letters represent the hermit crabs and the numbers represent the new shells. A single line connecting a letter to a number means that the hermit crab has determined that the shell is a good fit. If there is no line connecting a letter to a number then the hermit crab has determined that the shell is not a good fit.



Question

If a shell can only be worn by one hermit crab, and a hermit crab can only wear one shell, then what is the maximum number of hermit crabs that can wear one of these new shells?

(A) 8

(B) 7

(C) 6

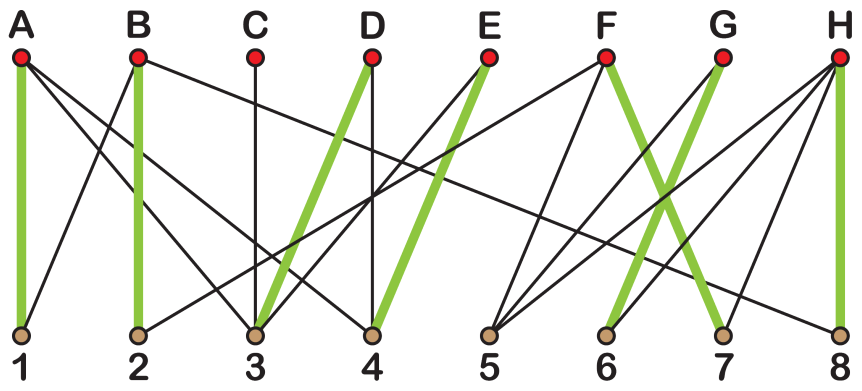
(D) 5

Answer

(B) 7

Explanation of Answer

Here is one way to match 7 of the hermit crabs to new shells.



It is not possible for all 8 of the hermit crabs to wear one of these new shells. Consider the crabs lettered C, D and E. Each of these crabs can only wear the shells numbered 3 and 4.

In other words, we have three crabs that can only wear two shells. This means it is guaranteed that one of those crabs will not get a new shell. Since we have found a way to distribute the remaining 7 shells, the maximum number of hermit crabs that can wear one of these new shells is 7.

Connections to Computer Science

In this problem, you are asked to connect hermit crabs with shells. These connections can be modelled using a *graph*. The *vertices* of the graph are the hermit crabs and the shells, and the *edges* of the graph join hermit crabs to shells that fit.

Since edges only exist between hermit crabs and shells, and there are no edges between hermit crabs, nor edges between shells, this type of graph is *bipartite*. That is, the vertices can be divided into two separate sets such that there are edges only between the two sets, and no edges between elements in the same set.

To match hermit crabs with new shells, we need to find a subset of edges such that no two edges share an endpoint. If two edges in our subset share an endpoint that would mean that either one hermit crab is wearing two shells, or one shell is being worn by two hermit crabs. When the subset contains as many edges as possible, it is called the *maximum bipartite matching*.

This type of matching is very useful in real life, like pairing organ donors with patients who need transplants. Using computer science to find these matchings quickly and efficiently is very important.

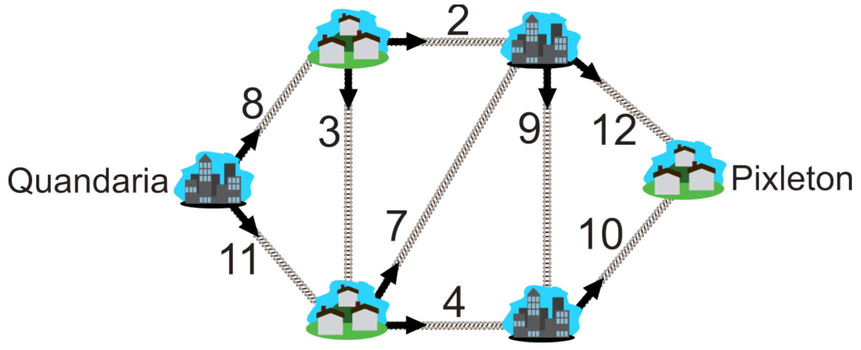
Country of Original Author

Canada

Railway Network

Story

In the land of Bebravia, there are six towns. All food is grown in Quandaria and transported by train to the other towns. The train routes are indicated by lines drawn between towns in the diagram. The arrow on each line indicates the direction in which trains carrying food are allowed to move.



Bebravia is not large so there is more than enough time for many trains to travel between any two towns. The only limitation is the daily capacity on each train route: the number on each line indicates the maximum number of trains that can travel along that route on one day.

Question

In one day, what is the largest number of trains that can travel from Quandaria to Pixleton?

(A) 13

(B) 16

(C) 19

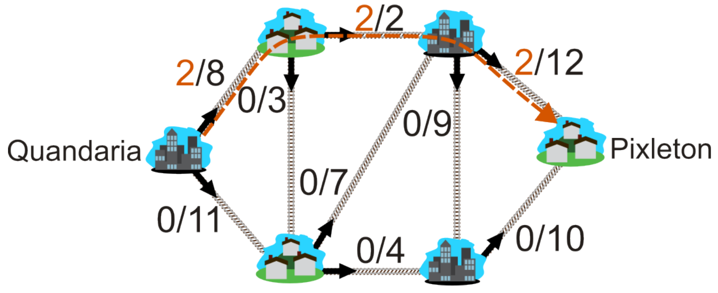
(D) 12

Answer

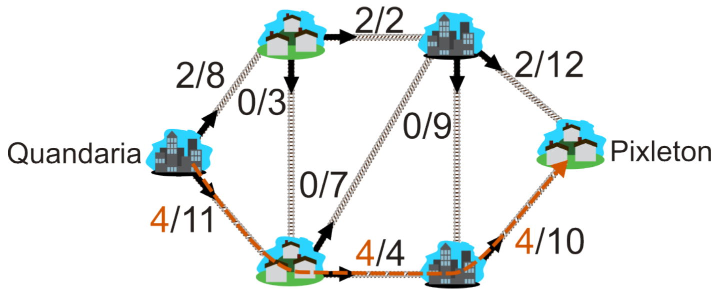
(A) 13

Explanation of Answer

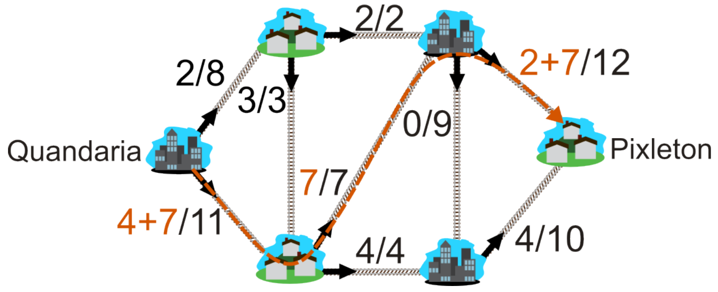
We consider different routes from Quandria to Pixleton. One path uses lines along the top of the diagram and it can accommodate up to two trains.



A completely separate second path uses lines along the bottom of the diagram and it can accommodate up to four trains.



A third path that can accommodate 7 trains is shown below.



Note that even though this third path overlaps with the two previous paths, there is still capacity for all three paths to be fully used meaning a total of $2 + 4 + 7 = 13$ trains can use these paths to travel from Quandria to Pixleton.

We can determine that 13 is the largest number of trains that can travel from Quandria to Pixleton by looking at the three lines in the middle of the diagram with capacities 2, 7 and 4. All paths from Quandria to Pixleton use one of these lines, so it is not possible for more than $2 + 7 + 4 = 13$ trains to go from Quandria to Pixleton.

Connections to Computer Science

This problem is an instance of a *maximum flow problem*. Maximum flow problems involve determining the maximum amount of *flow* from a *source* node to a *sink* node in a *graph*, where each *edge* has a maximum capacity.

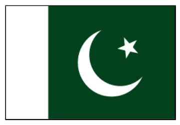
Maximum flow problems can be solved using various algorithms: three well-known algorithms are the *Ford-Fulkerson* algorithm, the *Edmonds-Karp* algorithm, and *Dinic's* algorithm. Each of these algorithms has advantages and disadvantages in terms of runtime efficiency, ease of implementation, and applicability to different types of networks. The choice of algorithm depends on factors such as the size of the network, the nature of the capacities, and the desired level of optimization.

The maximum flow is directly related to the *minimum cut*: the least amount of capacity on edges which could be removed entirely that separates the source node from the sink node. As pointed out in the Explanation of Answer, the minimum cut in this task has capacity 13, and if these edges were removed, there would be no way to travel from the source node (Quandaria) to the sink node (Pixleton).

Maximum flow algorithms have many applications: traffic optimization, distribution systems (e.g., water and electricity), resource allocation, and computer networking, to name a few.

Country of Original Author

Pakistan



Part C

Card Values

Story

Five types of cards are labelled with the letters P, Q, R, S, and T on one side. On the opposite sides, the cards are labelled with exactly one of the numbers 1, 2, 4, 8, and 16, in some random order. For each letter, all cards with that letter have the same number on the opposite side (e.g. all cards with P on one side have the same number on the opposite side.)

Multiple copies of each type of card are gathered to form a deck. Fala, Grace, and Hari each draw the smallest possible number of cards from the deck so that the numbers on their cards add up to their own age.

- Fala is 17 years old. One of the cards she is holding has an R on it.
- Grace is 18 years old. One of the cards she is holding has a Q on it. Another card she is holding is the same type as one of Hari's cards.
- Hari is 15 years old. He is the only one holding a card with a P on it. The largest number on a card he is holding has a T on the opposite side.

Question

Which number is on the cards that have the letter S on them?

(A) 1

(B) 2

(C) 4

(D) 16

Answer

(B) 2

Explanation of Answer

Fala is 17 years old, so is holding the cards with numbers 1 and 16.

Grace is 18 years old, so is holding the cards with numbers 2 and 16.

Hari is 15 years old, so is holding the cards with numbers 1, 2, 4 and 8.

Since 8 is the largest number on a card that Hari is holding, and we are told that the largest number on a card he is holding has a T on it, the T cards must have 8 on them. Then, since Hari and Fala both have a card with a 1 on it, Hari and Grace both a card with a 2 on it, and Hari is the only one holding a card with a P on it, we can conclude that the P cards must have 4 on them.

Grace took a Q card, but it's not the one she shares with Hari and that shared card must have the number 2 on it. Therefore, the Q cards must have 16 on them. Now Fala also has one of these Q cards and the first statement tells us that their other card has an R on it. Therefore the R cards must have 1 on them.

By process of elimination, the S cards must have 2 on them.

Connections to Computer Science

The most common representation of numbers in computer science is the *binary number system*. The binary number system uses *base-2*, which uses the digits 0 or 1, unlike the more common base-10 system, which uses digits 0 to 9. To understand base-2 numbers, we can compare them to base-10 numbers. For example, the base-10 number 123 is a representation of $100 + 20 + 3 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$. That same number can be represented in base-2 as

$$\begin{aligned} 1111011 &= 1 \cdot 64 + 1 \cdot 32 + 1 \cdot 16 + 1 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 \\ &= 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0. \end{aligned}$$

Every number which can be represented in base-10 can also be represented in base-2. Using that fact in this task, we can write the age of Fala, Grace, and Hari in base-2. For example, since Grace is 18 years old, that binary representation of 18 is $18 = 16 + 2 = 10010$. Converting the ages into binary numbers allows the task to be more easily solved.

Computers store all information, whether that information is a number, character, image or sound, as sequences of binary digits, or *bits*. The reason why bits are used is because electricity, which is a necessary resource for modern computers, is either "on" (usually represented as 1) or "off" (usually represented as 0).

Country of Original Author

Slovakia

Finding Treasure

Story

Captain Bojana is looking for a treasure hidden on an island. The island is divided into 16 regions, each marked with a letter from A to P as shown.

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P

Captain Bojana can enter any number of regions into a special device and the device will tell her whether or not the treasure is in one of those regions. For example, if she enters regions A, C, and D into the device and the device shows "yes", then the treasure is in exactly one of regions A, C, or D.

Question

What is the least number of times Captain Bojana needs to use the device to guarantee she will determine which region contains the treasure?

(A) 4

(B) 6

(C) 8

(D) 12

Answer

(A) 4

Explanation of Answer

Suppose Captain Bojana enters letters of half of the regions into the device first. Then,

- If the device indicates that the treasure is in the given regions, the captain now further divides these regions into two halves and enters one half into the device.
- If the device indicates that the treasure is not in the given regions, the captain divides the other regions into two halves and enters one of the halves into the device.

Captain Bojana continues this process until she finds the region that has the treasure. Since there are 16 regions to start with, after she uses the device for the first time, there will be 8 regions remaining that might contain the treasure. After she uses the device for the second time, there will be 4 regions remaining that might contain the treasure. After she uses the device for the third time, there will be 2 regions remaining that might contain the treasure. After she uses the device for the fourth time, she will have determined which region contains the treasure.

We will now explain why 4 is the minimum number of times Captain Bojana needs to use the device.

At any point in time, we call all the regions that might contain the treasure, the *remaining regions*. To begin, all 16 regions are the remaining regions. Each time Captain Bojana uses the device, she divides the remaining regions into two smaller groups: those she enters letters for and those she does not enter letters for. One of these smaller groups becomes the new group of remaining regions.

Now, if Captain Bojana enters letters of fewer than half of the regions into the device first, then the number of remaining regions could be at least 9. For example, if she enters the letters of 3 regions into the device and receives the answer “no”, then there will be $16 - 3 = 13$ remaining regions. Or, perhaps she enters the letters of 10 regions and receives the answer “yes” in which case there will then be 10 remaining regions.

Since there is no guarantee that the number of remaining regions is less than 9 after using the device once, Captain Bojana could be faced with the possibility that there are 9 or more remaining regions. In that case, when Captain Bojana uses the device a second time, these remaining regions will be divided into two groups and at least one of these groups will have at least 5 regions. So there could be at least 5 remaining regions; Captain Bojana cannot guarantee that there will fewer than 5 remaining regions. Similarly, after Captain Bojana uses the device a third time, Captain Bojana cannot guarantee there are fewer than 3 remaining regions. At this point, she must use the device at least one more time and thus at least four times in total to determine which region contains the treasure.

Connections to Computer Science

In this task we need to repeatedly divide the regions in half to find the treasure in the most efficient way. More generally, in computer science, we can a similar *searching technique* to find an item in a sorted list by repeatedly dividing the list in half. Using this technique, we can eliminate half of the items each time and find the item in the fewest number of steps. This search technique is called *binary search*.

Notice that by reducing the size of the problem by half at each step, the *running time* of searching is *exponentially faster* than searching through the entire set of all possible values using *linear search*. This approach of repeatedly reducing the problem by half is known as a *divide and conquer* technique in computer science.

Binary search has many applications, such as searching for a word in a dictionary or finding the location of a file or a record in a *database*.

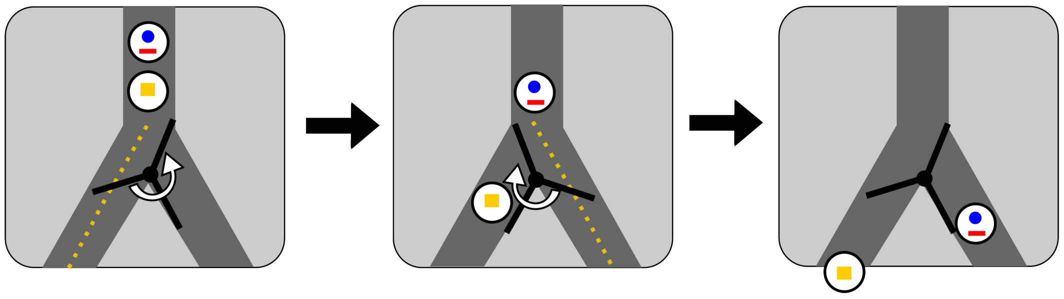
Country of Original Author

Hungary

Floor Patterns

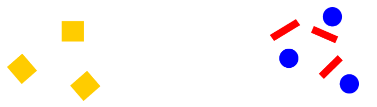
Story

A machine uses gates to redirect a sequence of balls in order to produce a pattern on the floor when viewed from above. Each ball follows a path directed by the gates. After a ball passes through a gate, the gate switches direction, sending the next ball the other way, as shown below.



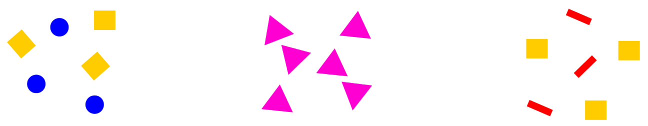
In the example above, the gate initially opens to the left, sending the first ball left. As the first ball goes through the gate, it causes the gate to switch so that it will send the next ball to the right. As the second ball goes through the gate, it causes the gate to switch back again.

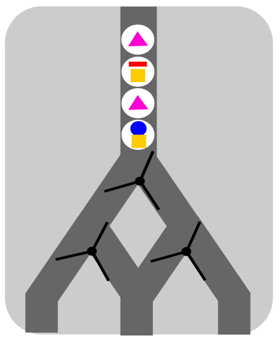
Every ball is labelled with one or more shapes. When a ball hits the floor, the pattern produced on the floor is triple the shape(s) shown on that ball. When two or more balls land at the same location, their patterns will combine on the floor. The pattern created on the floor by the example above is shown below.

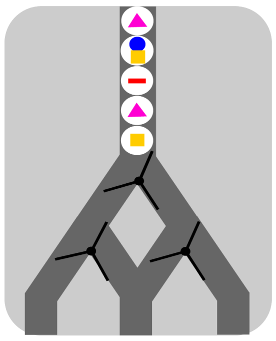


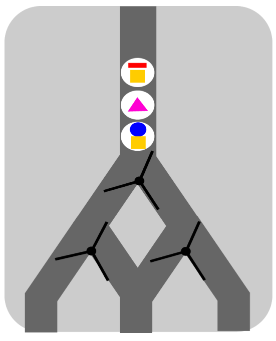
Question

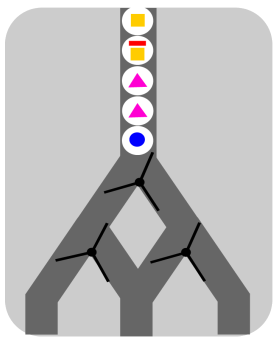
Which sequence of balls will create the following pattern on the floor?



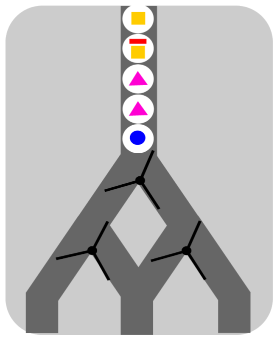
(A) 

(B) 

(C) 

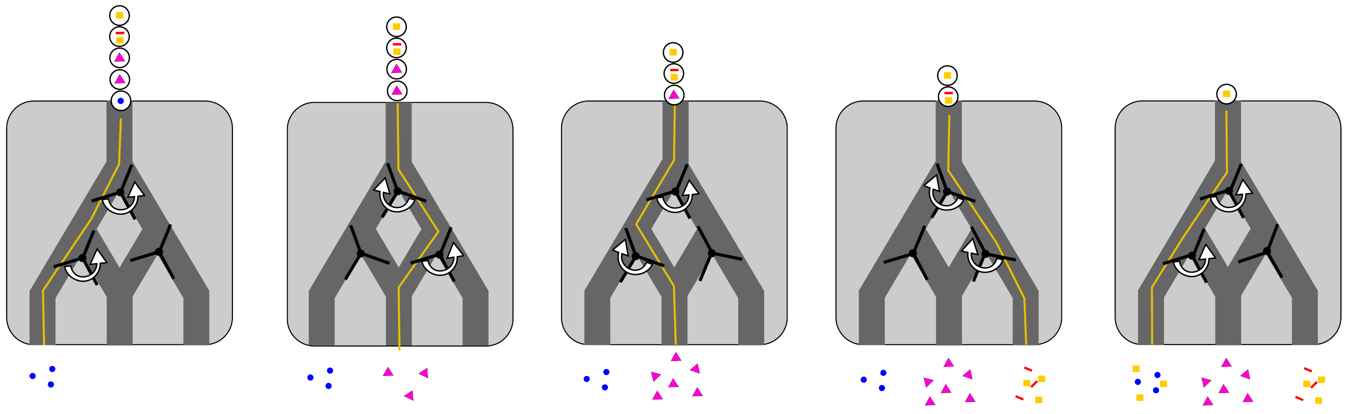
(D) 

Answer

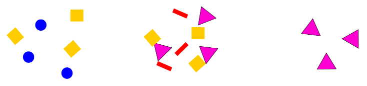
(D) 

Explanation of Answer

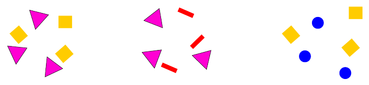
The following diagram shows how Option D creates the desired pattern.



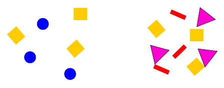
Option A is incorrect because it will create the following pattern.



Option B is incorrect because it will create the following pattern.



Option C is incorrect because it will create the following pattern.



Connections to Computer Science

Computer chips, such as the *central processing unit (CPU)* or *random access memory (RAM)* are built using extremely small electrical components known as *transistors*. A transistor can be thought of as a switch, which can be opened (or “on” or “1”) if there is electrical current applied to it, or closed (or “off” of “0”) if there is no electrical current. Combining multiple transistors together allows computers to store more information, such as a sequence of binary numbers, or *bits*.

In this task, each gate represents a transistor, which can be “on” (meaning the next ball will fall to the left) or “off” (meaning the next ball will fall to the right).

Modern *microchips* are composed of millions of transistors packed into an area that is less than 2cm by 2cm.

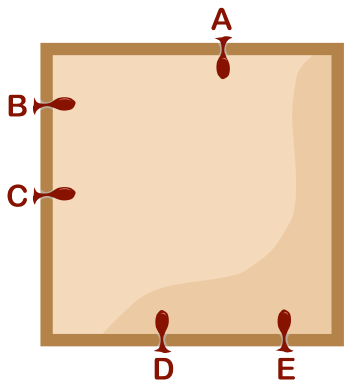
Country of Original Author

Ireland

Balloon Machine

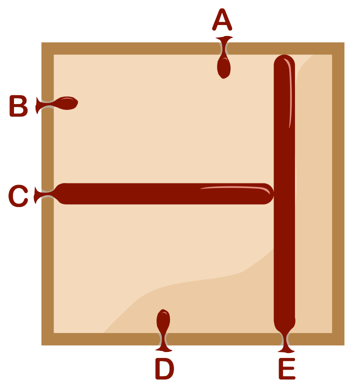
Story

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



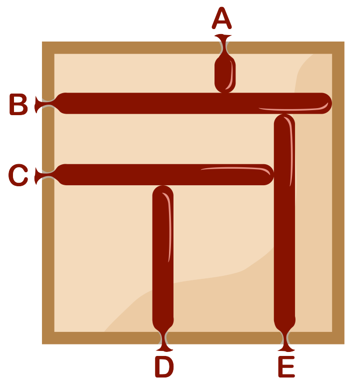
All balloons start deflated. To create a design, the machine reads a sequence of letters, from left to right, and inflates the corresponding balloons in that order. When a particular balloon is inflated, it will continue to extend until it reaches either another balloon or the opposite edge of the frame.

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



Question

Starting with all balloons deflated, the machine reads a sequence consisting of the five letters A, B, C, D, and E in some order and creates the design shown. How many different possibilities are there for the sequence?



(A) 1 (B) 2 (C) 4 (D) 5

Answer

(C) 4

Explanation of Answer

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

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

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

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

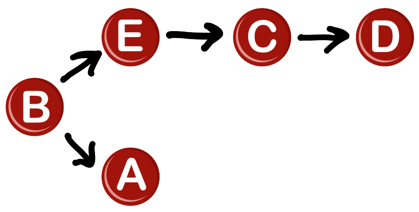
- $B\ A\ E\ C\ D$
- $B\ E\ A\ C\ D$
- $B\ E\ C\ A\ D$
- $B\ E\ C\ D\ A$

Connections to Computer Science

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

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

Another computational way to reason about this problem is using a *directed acyclic graph (DAG)*. Looking at which ballons reach "as far as they can", we can draw a *directed edge* from a node u to node v if the node u must have been inflated before node v. This graph is *acyclic*, meaning "without a cycle", since a node cannot be inflated before itself. For this task, the DAG is shown below:



All four sequences described in the Explanation of Answer can be thought of as moving from left-to-right in this DAG. This arrangement of the elements is known as a *topological ordering*.

One main application of DAGs is to model *task dependencies* (i.e., which tasks must occur before other tasks), which is crucial for *scheduling management*. DAGs are also used in *parallel processing* and *data compression*.

Country of Original Author

Germany

Word Chains

Story

Mr. Castor is teaching his students how to read. To help them learn he creates word chains, which are sequences of words in which exactly one letter in a word is changed in order to create the next word. For example, MUG → MUD → MAD → FAD is a word chain.

Mr. Castor has the following nine words: BOT, SAD, BAT, CAB, COT, BAD, COB, CAT, and SAT. He groups them into three word chains, each with three words, so that each of the nine words is used in exactly one of the word chains.

Question

Which of the following **cannot** be one of Mr. Castor's three word chains?

(A) SAD → BAD → BAT

(B) COT → COB → CAB

(C) BAT → CAT → COT

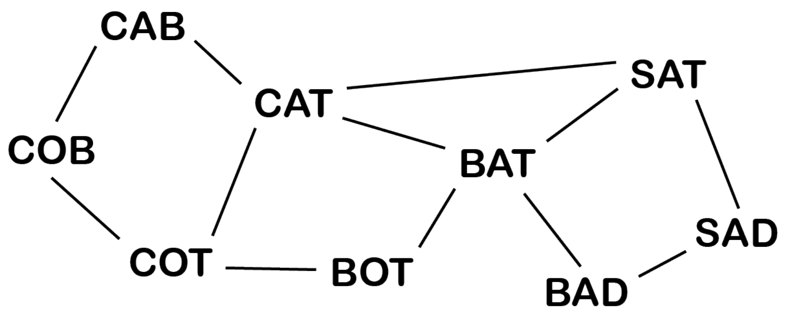
(D) CAT → CAB → COB

Answer

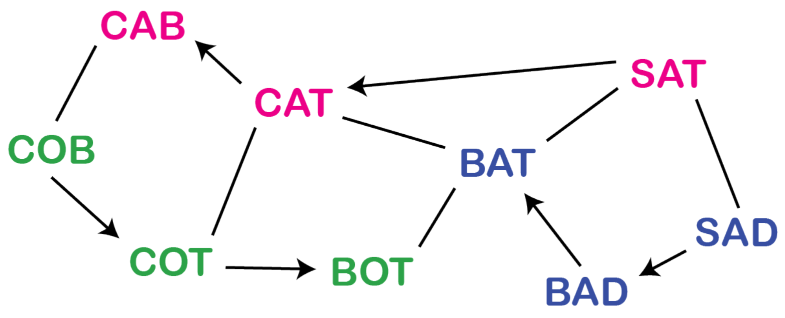
(C) BAT → CAT → COT

Explanation of Answer

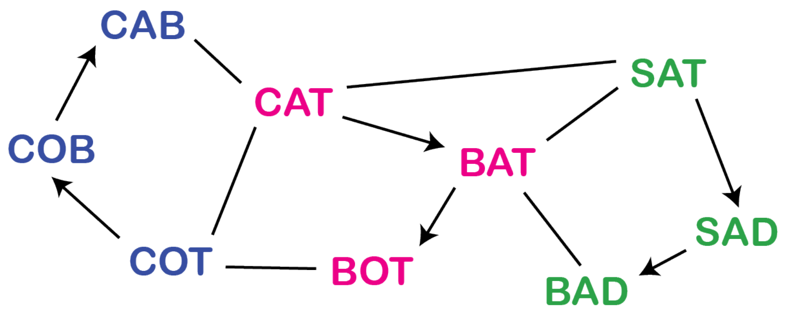
To solve this problem, it is helpful to draw a diagram. In this diagram, two words are connected by a line if they can follow each other in a word chain.



Option A is a possible word chain. In this case, the other two word chains could be SAT → CAT → CAB and COB → COT → BOT, as shown.

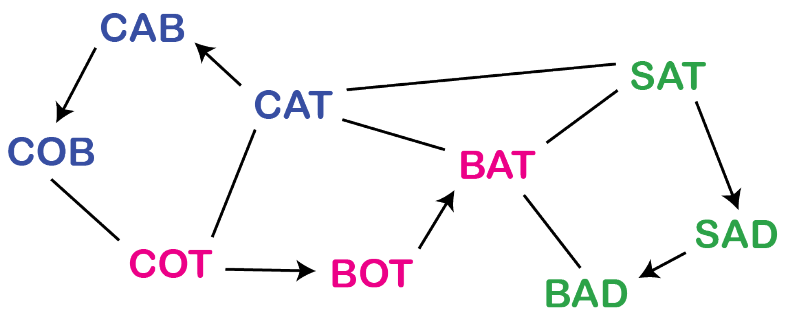


Option B is a possible word chain. In this case, the other two word chains could be CAT → BAT → BOT and SAT → SAD → BAD, as shown.



Explanation of Answer

Option D is a possible word chain. In this case, the other two word chains could be COT → BOT → BAT and SAT → SAD → BAD, as shown.



Option C is not a possible word chain because in the diagram, BOT is connected only to BAT and COT. Thus, BOT must be in a word chain with at least one of these words. Since BAT and COT are both in the word chain for Option C, and BOT is not, that leaves no possible word chain for BOT.

Connections to Computer Science

In computer science, this type of diagram is called a *graph*, and it is used to show connections between elements. A graph consists of a set of *vertices* and a set of *edges*, where each edgeconnects a pair of vertices. In this task, the vertices represent the words, and there is an edge between two vertices if they can follow each other in a word chain (i.e., if the words differ by exactly one letter).

A word chain is found by recording the words along a sequence of connected edges, known as a *path* in the graph. Since there exists a path between any two vertices in this graph, this graph is a *connected graph*. Grouping the nine words into three word chains, each with three words is the equivalent to removing edges of the graph in order to create a *disconnected graph* with three *connected components*, each with three vertices. In this task, we are determining whether or not a given set of three words could be a component of such a disconnected graph.

Determining the connected components of a graph is a fundamental concept in *image analysis*. For example, determining which regions of the same (or similar) colour connect together can help determine the outline of objects in photographs.

Country of Original Author

Canada