



UNIVERSITY OF
WATERLOO



The CENTRE for EDUCATION in
MATHEMATICS and COMPUTING



2019
*Beaver
Computing
Challenge*
(Grade 9 & 10)

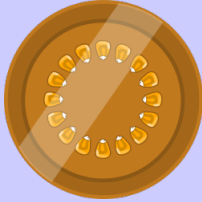
*Questions,
Answers,
Explanations,
and
Connections*

Part A

Beaver Coins

Story

Beavers use coins with the following values:



16



8



4



2



1

Question

Which of the following total values can be made using exactly three coins?

- (A) 23
- (B) 2
- (C) 38
- (D) 13

Answer

(D) 13

Explanation of Answer

Since $1 + 4 + 8 = 13$, a total value of 13 can be made using exactly three coins: one coin of value 1, one coin of value 4, and one coin of value 8. So Option D is correct but what about the other options?

It is not possible to make a total value of 2 using exactly three coins because each coin has a value of 1 or more.

To make a total value of 23, first notice that the coin of value 1 is the only coin with an odd value. The other coins all have even values and adding even valued coins can only make even total values. So, to make a total value of 23 we need one coin of value 1 and two coins that make a total value of 22. However, this is not possible. If we used a coin of value 16 we would need another coin of value 6. If we used a coin of value 8 we would need another coin of value 14. If we used a coin of value 4 we would need another coin of value 18. If we used a coin with a value less than 4, we would need another coin with a value greater than 18.

Finally, to make a total value of 38, at least one coin of value 16 is needed. This is because using three coins of value 8 would only make a total value of 24. So, to make a total value of 38 we need one coin of value 16 and two coins that make a total value of 22. As shown above, this is not possible.

Connection to Computer Science

In mathematics class, we often think of numbers obtained by sums of multiples of 1, 10, 100, 1000, etcetera. This is the decimal number system where the number 10 is the *base* of the system. In this problem, you are asked to think about which values can be obtained by using coins with the values 1, 2, 4, 8, and 16. We can think of the base here as 2 and we are exploring what is called the *binary system*. While humans generally think and communicate using the decimal system, computers generally operate in the binary system. This is because the operation of a computer is fundamentally based on whether or not there is enough electricity at some point. Either there is, or there isn't. There are exactly two possibilities.

The idea of using different bases and number systems is not new. Far before computers were invented, ancient peoples used these ideas. One example is the *abacus*, which has been used by many different civilizations for thousands of years.

Country of Original Author

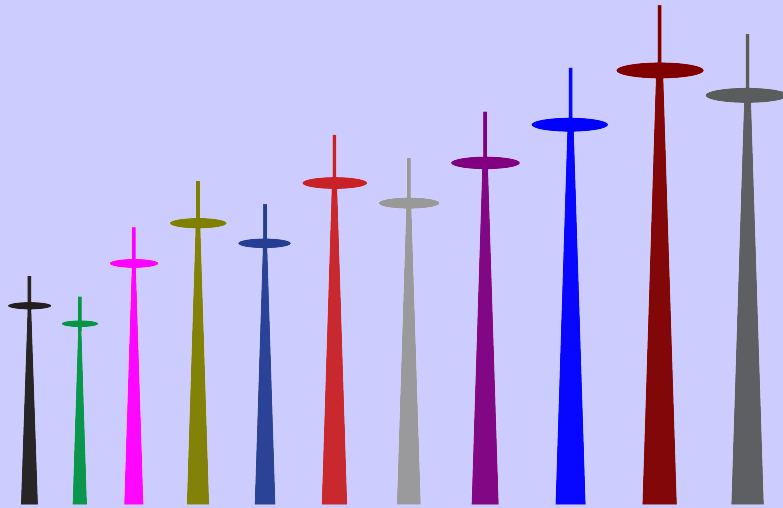
Switzerland



Special Towers

Story

Consider the towers shown.



A tower is *special* if all towers to the left of it are shorter, and all towers to the right of it are taller.

Question

How many special towers are there?

- (A) 1
- (B) 2
- (C) 3
- (D) 4

Answer

(C) 3

Explanation of Answer

Figures 1 and 2 show that the eighth tower is special. A tower is special if all the tips of the towers to the left are within the coloured rectangle (see Figure 1); and if all the tips of the towers to the right are outside the coloured rectangle (see Figure 2).

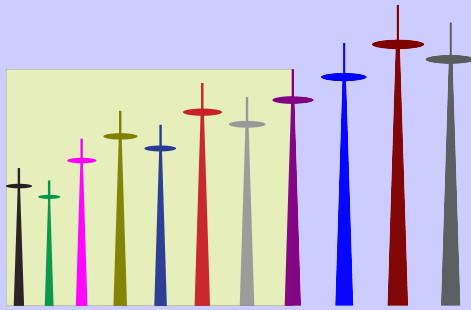


Figure 1

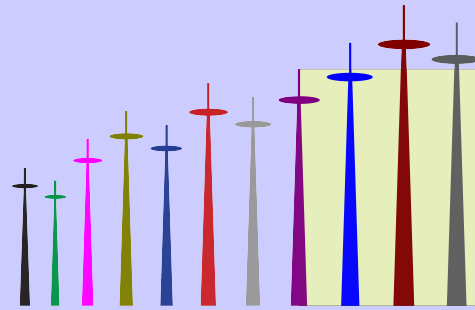


Figure 2

One possible way to find all the special towers is to move through all the towers, one by one, and mark a tower if all to the left of it are shorter (see Figure 3). Then make a similar second pass, marking a tower if all to the right of it are taller (see Figure 4).

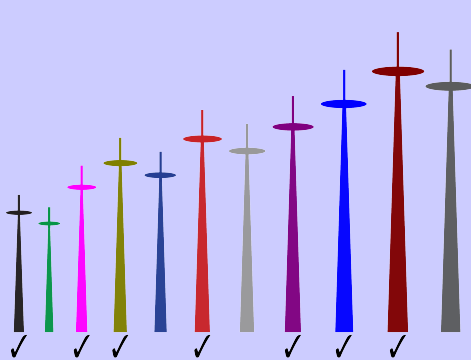


Figure 3

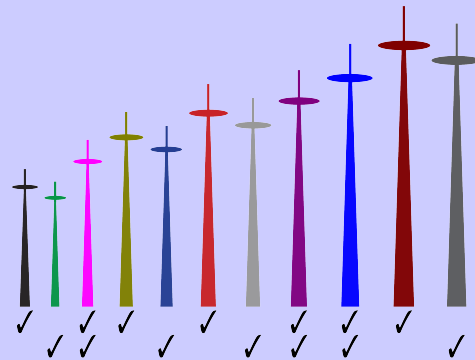
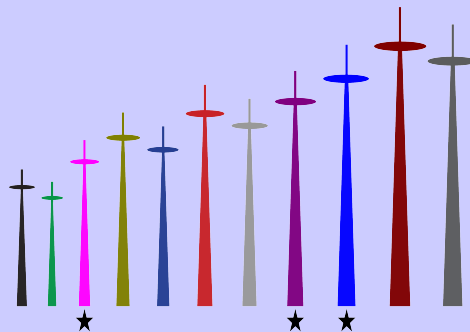


Figure 4

All towers marked twice are special towers.



Connections to Computer Science

The order of objects and *sorting* them into order is among the most studied problems in computer science. This problem concerns itself with the order of towers based on their height. In particular, a special tower is what we call a *pivot* in the *partition* algorithm used by the *quicksort* algorithm. As indicated by its name, quicksort tends to be very fast at ordering data and is one of the most famous and frequently used sorting methods in computer science.

Country of Original Author

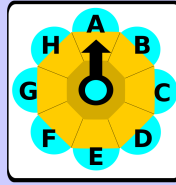
Canada



Safe

Story

A chef keeps secret recipes in a safe. It is unlocked using a circular knob with a pointer.



With the pointer at A to start, the chef unlocks the safe by turning the knob clockwise and counter-clockwise alternately as the password is spelled. For example, to enter the password BH, the chef:

turns one position clockwise	then turns two positions counter-clockwise

We represent passwords using numbers to indicate how far to turn and arrows to show the direction. For example, BH is represented by $1\curvearrowright 2\curvearrowleft$ which means turn one position clockwise and then two positions counter-clockwise.

To retrieve the secret recipes, the chef must enter the password CHEFDG.

Question

With the pointer starting at A, which of the following will unlock the safe?

- (A) $2\curvearrowright 3\curvearrowleft 4\curvearrowright 3\curvearrowleft 3\curvearrowright 3\curvearrowleft$
- (B) $2\curvearrowright 5\curvearrowleft 5\curvearrowright 1\curvearrowright 3\curvearrowright 3\curvearrowleft$
- (C) $2\curvearrowright 3\curvearrowleft 5\curvearrowright 7\curvearrowleft 6\curvearrowright 5\curvearrowleft$
- (D) $2\curvearrowright 1\curvearrowleft 4\curvearrowright 3\curvearrowleft 3\curvearrowright 2\curvearrowleft$

Answer

- (C) $2\curvearrowright 3\curvearrowleft 5\curvearrowright 7\curvearrowleft 6\curvearrowright 5\curvearrowleft$

Explanation of the Answer

Option C correctly tells the chef to move:

- two positions clockwise from A to C, then
- three positions counter-clockwise from C to H, then
- five positions clockwise from H to E, then
- seven positions counter-clockwise from E to F, then
- six positions clockwise from F to D, and finally
- five positions counter-clockwise from D to G.

Option A begins by incorrectly telling the chef to spell CHD. Option B begins by incorrectly telling the chef to spell CF. Option D begins by incorrectly telling the chef to spell CB.

Connections to Computer Science

When solving a problem it is often helpful to keep track of the current position or *state* of an object. In this problem, where the arrow is pointing is part of the state of the safe. The state of an object can also include a history of actions done on it. Here, as the knob is turned, we must remember which letters of the password have already been spelled.

Performing the same action on an object may not always have the same effect. Actions done on an object can have different effects based on its current state. For example, in this problem, turning five positions counter-clockwise after spelling “CHEFD” will open the lock. But turning it five positions counter-clockwise after spelling only “CHE” will not.

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

There are many other examples where the state is important. When following directions on a phone while walking, the important state will include your current location. When writing a computer program, a programmer has to decide what the state of the system they are working with is and write the program to correctly keep and update that state.

Country of Original Author

Taiwan



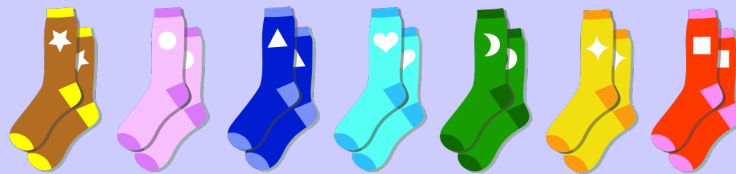
Socks

Story

Anil likes to vary the colour of socks he wears. He keeps all of his socks arranged in a line and follows the rules given below to choose a pair of socks for the day.

- Socks to be worn are always taken from the right end of the line.
- Socks are washed right after they are worn, and are then added to the left end of the line.

On the morning of November 18, Anil wakes up and sees the following line of socks.



Question

Which pair of socks will Anil wear on November 28?

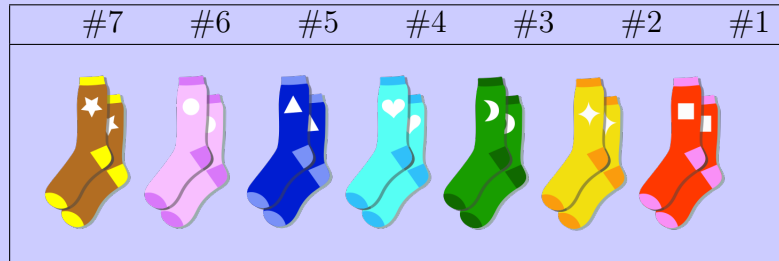


Answer



Explanation of Answer

Notice that Anil has seven pairs of socks. Since he puts worn and washed socks at the left end of the line, and he takes them from the right end of the line, he will wear each pair of socks exactly once after seven days. This means that when Anil wakes up seven mornings later on November 25, his socks will be lined up exactly as they are when he wakes up on November 18. Now, tracking things carefully, label the seven socks as shown below.



Following the rules outlined, Anil will wear the red socks (# 1) on November 25, and then place the red socks to the left of the brown socks (# 7) after wearing and washing them. Continuing in this way, we can make a table showing exactly which socks Anil will wear each day:

Nov. 25	Nov. 26	Nov. 27	Nov. 28
# 1	# 2	# 3	# 4

We conclude that Anil will wear Option A on November 28.

Connections to Computer Science

In this problem, the next socks Anil will wear are the ones that have been in the line the longest. That is, among those currently in the line, the next socks that Anil will wear are those that were washed before any other socks in the line. We say that the line of socks follows the *first-in first-out* or *FIFO* principle.

A collection of items from which items are removed according to the FIFO principle is called a *queue*. A queue is a fundamental *data structure* which is a particular way of organizing data in the memory of a computer. Compare the queue of socks in this problem to the stack of boxes in the Packing Machine problem on the Grade 7/8 Beaver Computing Challenge or the Box of Balls problem on the Grade 5/6 Beaver Computing Challenge.

Common applications of queues include lists of documents waiting to print and modelling people waiting in a line.

Country of Original Author

India



Language Detection

Story

Explorers stumble across the following ancient list of five words on the wall of a cave.

paqroob *puue* *t'seqrub* *meoub* *lai'laiqy*

The explorers use a system to try and determine which language each word is from.

- Each word is given an initial score of 10.
- The score is adjusted using the following rules:

starts with p	-2
ends with b	-2
more than 6 characters	+3
has q followed immediately by r or y	-4
has three vowels (a, e, i, o, u) in a row	+5
contains an apostrophe (')	+1

- If the final score is 10 or greater, the system classifies the word as Beaverish.
- Otherwise, the system classifies the word as Beaverian.

For example, the word *palliob* is given a score of $10 - 2 - 2 + 3 + 0 + 0 + 0 = 9$ and so the system classifies *palliob* as Beaverian.

Question

Using this system, how many of the five words from the cave are classified as Beaverish?

- (A) 2
- (B) 3
- (C) 4
- (D) 5

Answer

(B) 3

Explanation of Answer

The following table summarizes the system scores and classification.

word	score	classification
<i>paqroob</i>	$10 - 2 - 2 + 3 - 4 + 0 + 0 = 5$	Beaverian
<i>puue</i>	$10 - 2 + 0 + 0 + 0 + 5 + 0 = 13$	Beaverish
<i>t'seqrub</i>	$10 + 0 - 2 + 3 - 4 + 0 + 1 = 8$	Beaverian
<i>meoub</i>	$10 + 0 - 2 + 0 + 0 + 5 + 0 = 13$	Beaverish
<i>lai'laiqy</i>	$10 + 0 + 0 + 3 - 4 + 0 + 1 = 10$	Beaverish

We see that three words (*puue*, *meoub* and *lai'laiqy*) are classified as Beaverish.

Connections to Computer Science

The application of computer science to the processing of natural human languages remains an exciting and active area of study. This includes automatically converting voice to text and text to voice. Many of us own devices that already use this technology. For example, you may own a cell phone that supports voice texting or have a device set up at your house that will obey commands to play music or turn on the lights. Another common example is the translation of text from one language to another. If the originating language is unknown, the first step in this process is *language detection* which is what this particular problem is about.

In general, these problems in *natural language processing* are very difficult and perfect solutions do not (yet) exist. As in this task, we use *heuristics* which are rules of thumb that we hope give us good answers most of the time. Heuristics are very simple strategies with sometimes astonishingly good results. Another exciting application of natural language processing, where psychologists and computer scientists are working together, is the development of programs that detect someone's age or try to determine if somebody is lying.

Country of Original Author

Canada



Part B

Celebrity

Story

Members of a social network may follow other members. Within a group of members, a particular member is called a *celebrity* if they are someone who



- is followed by everyone in the group, and
- does not follow anyone in the group.

Question

What is the maximum possible number of celebrities in a group of five members?

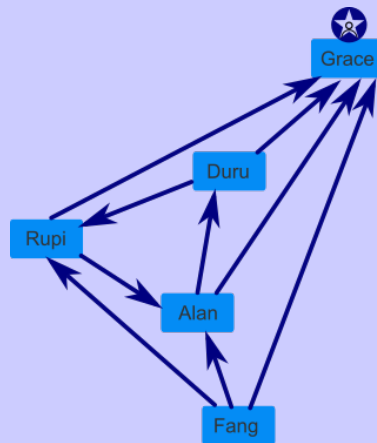
- (A) 0
- (B) 1
- (C) 2
- (D) 5

Answer

- (B) 1

Explanation of Answer

It is certainly possible to have at least one celebrity in the group. An example is shown where each arrow points from a member of the group to another member that he or she follows. In the example, Grace is a celebrity.



If there is a celebrity in the group, then he or she does not follow anyone else. This means there cannot be another celebrity in the group. That is, the maximum possible number of celebrities in a group is one.

Connection to Computer Science

Social networks are based on the relationships between their members, which can be modelled by a *directed graph*. We can represent the members of the social network by the *vertices* of the directed graph and an instance of someone following someone else by a (*directed*) *edge*. In the diagram, the vertices are the boxes labelled with names and the edges are the arrows. The reason we say “directed” is because it demonstrates which members are being followed and by whom. For example, if Member A is following Member B, then we have a directed edge from Member A to Member B.

In this problem, we consider both the *out-degree* of each vertex (how many people the corresponding member follows) and the *in-degree* (how many people follow the corresponding member). In a graph with n vertices, the out-degree and in-degree of each vertex can be at most $n - 1$. We see in the solution that if a vertex has in-degree $n - 1$ and out degree 0, then it is the only vertex with this property.

In general, graph properties can be calculated and studied. This information has proven to be very valuable in marketing and politics and raises important questions about privacy and security.

Country of Original Author

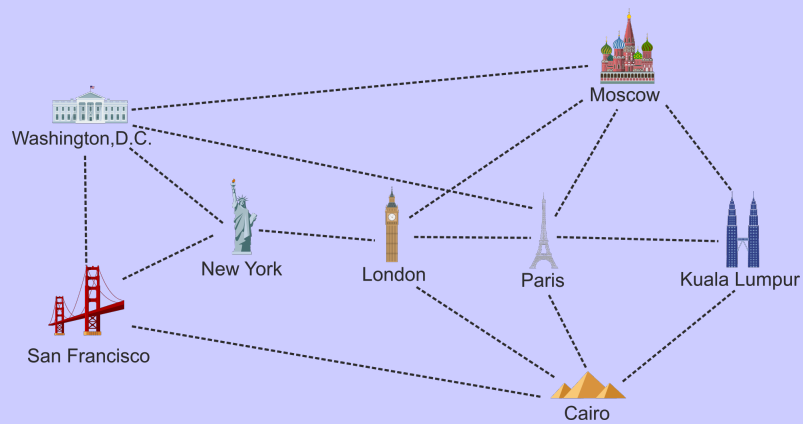
Germany



Cancelled Flights

Story

The dashed lines in the diagram represent all Bebras Air flights. Each flight operates in both directions. The airline is popular because its customers are able to fly between any two cities (possibly stopping in one or more cities in between).



The airline wants to cancel some flights but it still wants its customers to be able to fly between any two cities.

Question

What is the maximum number of flights that Bebras Air can cancel?

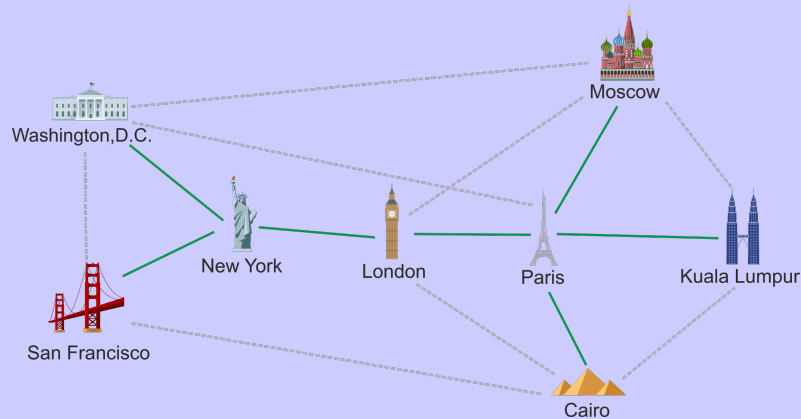
- (A) 6
- (B) 7
- (C) 8
- (D) 9

Answer

(C) 8

Explanation of Answer

The example below shows that it is possible to cancel 8 flights and still allow customers to fly between any two cities. Note that 7 flights are left.



We need to show that 8 cities cannot be connected in this manner using less than 7 flights. We will do so by showing that it cannot be done with 6 flights (and so cannot be done with fewer either).

Suppose that we have 6 flights arranged in such a way that customers are able to fly between any two of the 8 cities. Each of the 6 flights involves 2 cities and so if we list all of the cities involved in the flights we get $6 \times 2 = 12$ cities in total (with some repetitions). We know that each city must be a part of at least one flight and so each city appears in this list at least once. Since there are 8 cities, and our list contains 12 in total (with repetitions), some cities must appear only once in the list and hence cannot be involved in more than 1 flight.

Let City A be a city that is involved in exactly 1 flight. Remove this flight and City A itself from the picture. What we have left is 7 cities connected using 5 flights. Note that removing the single flight involving City A cannot break the connectivity of the remaining 7 cities. We can use similar reasoning to argue that at least one of these 7 cities, call it City B, is involved in exactly 1 flight. When we remove City B, and the accompanying flight, we are left with 6 cities connected using 4 flights. Similarly, we can produce 5 cities connected using 3 flights and then 4 cities connected using 2 flights and finally 3 cities connected using 1 flight. Now we have reached a clear problem. We cannot possibly have 3 cities connected using only 1 flight. This means that we could not have had the original scenario where we claimed to have 8 cities connected using only 6 flights.

Connections to Computer Science

There is a *graph* at the centre of this problem. A graph is used to model connections. It consists of a set of vertices and a set of edges each of which connects a pair of vertices. Here, the vertices represent cities and the edges represent flights. The diagram is a visual representation of the graph.

The goal is to remove as many edges (flights) as possible while leaving a *connected* graph. A connected graph is one for which there exists a *path* between any two vertices. In general, if a connected graph has n vertices, then there exists a set of only $n - 1$ edges, chosen from the “original” edges, for which the graph is still connected if all other edges are removed. The resulting graph with $n - 1$ edges is called a *minimum spanning tree*. If a graph with n vertices has less than $n - 1$ edges, it cannot be a connected graph.

Minimum spanning trees are important in applications such as transportation and water supply networks.

Country of Original Author

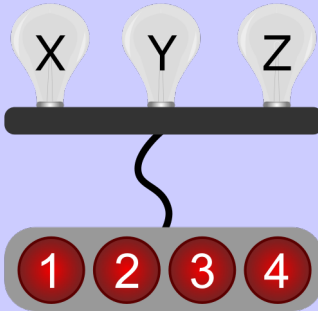
Belgium



Light Buttons

Story

There are three light bulbs, labelled X, Y, and Z, which can be turned on and off using four buttons numbered 1, 2, 3, and 4. Each button produces a different action when pressed as described below.



Button	Action
1	turn on bulb Y and turn off bulb X
2	turn on bulbs X and Y and turn off bulb Z
3	turn on bulb Z and turn off bulb Y
4	turn on bulb X

If a button attempts to turn a particular light bulb on that is already on, then that light bulb will remain on. Similarly, if a button attempts to turn a particular light bulb off that is already off, then that light bulb will remain off.

All three light bulbs are currently off. You want them all on after pressing a sequence of buttons.

Question

Which of the following sequences should you use?

- (A) 2, 3, 1
- (B) 2, 3, 4
- (C) 4, 1, 3
- (D) 3, 1, 4

Answer

(D) **3** , **1** , **4**

Explanation of Answer

First, we verify that pressing button 3, button 1, and button 4, in that order, will produce the desired configuration:

If button 3 is pressed first, then bulb X remains off, bulb Y remains off, and bulb Z is turned on.
If button 1 is pressed second, then bulb X remains off, bulb Y is turned on, and bulb Z remains on.
If button 4 is pressed third, then bulb X is turned on, bulb Y remains on, and bulb Z remains on.

Now, we also eliminate the other options. Note that button 1 turns off bulb X, button 2 turns off bulb Z and button 3 turns off bulb Y. Therefore, a sequence that ends by pressing buttons 1, 2, or 3, cannot produce the desired configuration. In particular, Options A and C cannot be correct.

The sequence in Option B also does not produce the desired configuration. It involves turning off bulb Y when button 3 is pressed and bulb Y is not turned back on at the end when button 4 is pressed.

Connections to Computer Science

If we write out which bulbs are on and which ones are off, we completely describe the relevant *state* of the light bulbs. Each bulb has two possibilities (off or on) and there are three bulbs, so there are $2 \times 2 \times 2 = 8$ different states. Each press of a button takes us from the current state to a new state. This is called a *transition*. A system with a finite set of states and actions that cause transitions between the states is called a *deterministic finite automaton (DFA)* or a *finite state machine*.

DFAs show up in more places than you might expect. For example, a traffic light has different states (e.g. red, yellow, or green) and changes between the states based on its environment (e.g. a car is waiting at an intersection). Other examples include devices with buttons (e.g. a coffee machine or elevator) which move between states with the press of a button. Careful design of states and transitions allow computer programmers to program such devices in efficient and reliable ways.

Country of Original Author

Pakistan



Classifier

Story

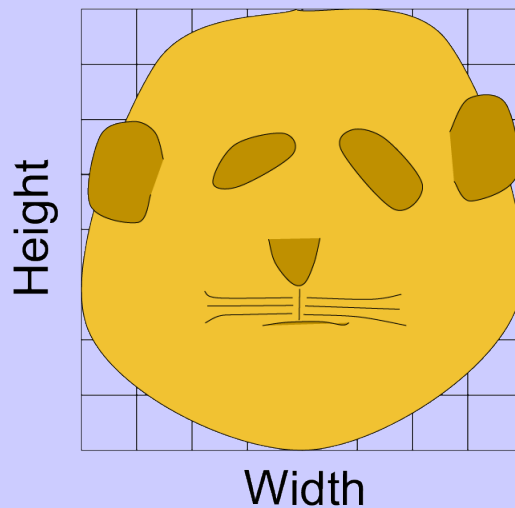
Given an image of an animal, a machine measures various parts of the animal: head, ears, and whiskers. The height of a part is the distance from its lowest point to its highest point. The width of a part is the distance from its leftmost point to its rightmost point.

These measurements are used to identify the animal based on the chart shown.

	Rabbit	Beaver	Bear	Cat
ear height	$\frac{1}{2}$ of head height	$\frac{1}{4}$ of head height	$\frac{1}{4}$ of head height	$\frac{1}{2}$ of head height
whiskers width	head width	$\frac{1}{2}$ of head width	$\frac{1}{2}$ of head width	head width
head width	$\frac{1}{2}$ of head height	$\frac{1}{2}$ of head height	head height	head height

Question

What type of animal does the machine identify the following image as?



- (A) Rabbit
- (B) Beaver
- (C) Bear
- (D) Cat

Answer

(C) Bear

Explanation of Answer

In this image, the head width is equal to that of the head height, the whiskers width is half of the head width, and the ear height is a quarter of the head height. These measurements match all the chart entries for a bear.

The image is not identified as a rabbit because it does not match any entries in the chart. The image is not identified as a beaver because the head width is equal to that of the head height. The image is not identified as a cat because the ear height and whiskers width do not match the chart.

Connection to Computer Science

We can imagine that the chart in this problem is produced by inspecting a large set of images of rabbits, beavers, bears, and cats. This would be an example of using *machine learning* to recognize images. The immense power in today's modern computers has made machine learning an increasingly important subdiscipline of *artificial intelligence*. This is because, to be effective, a machine learning *algorithm* typically needs to be trained on a large data set.

Another example of an application of machine learning is *autonomous driving*, where a machine learning algorithm is trained to recognize the lines and signs along a road as well as all possible objects encountered by a car.

Country of Original Author

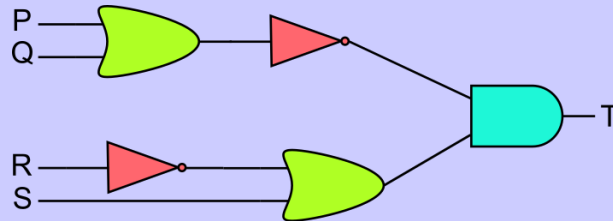
Pakistan




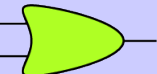
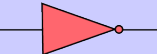
Escape Room

Story

An escape room uses the following design for the inner-workings of a lock. The lock is made using nine rods and five blocks.



Each rod turns like a key either clockwise or counter-clockwise. Each block has one or two input rods coming in from the left and one output rod coming out to the right. The table below describes how input rods and the type of block determine how output rods turn.

	If both input rods are turned clockwise, then the output rod will turn clockwise. Otherwise, the output rod will turn counter-clockwise.
	If both input rods are turned counter-clockwise, then the output rod will turn counter-clockwise. Otherwise, the output rod will turn clockwise.
	The output rod will turn in the opposite direction as the input rod.

Question

Each input rod at P, Q, R, and S is turned once causing the output rod at T to turn clockwise. How might have the input rods been turned?

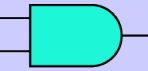
(A)	P clockwise	Q clockwise	R counter-clockwise	S clockwise
(B)	P counter-clockwise	Q counter-clockwise	R counter-clockwise	S clockwise
(C)	P clockwise	Q counter-clockwise	R counter-clockwise	S clockwise
(D)	P counter-clockwise	Q counter-clockwise	R clockwise	S counter-clockwise

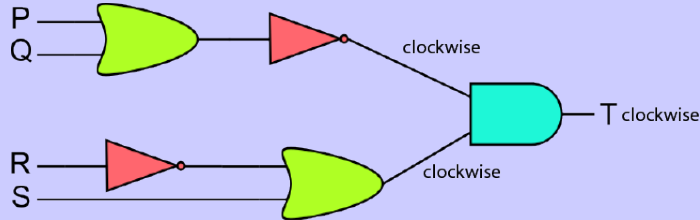
Answer

(B)	P counter-clockwise	Q counter-clockwise	R counter-clockwise	S clockwise
-----	------------------------	------------------------	------------------------	----------------

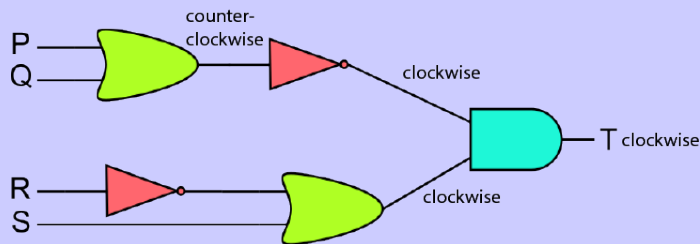
Explanation of Answer

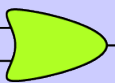
One way to solve this problem is to work backwards, starting with the output rod at T.

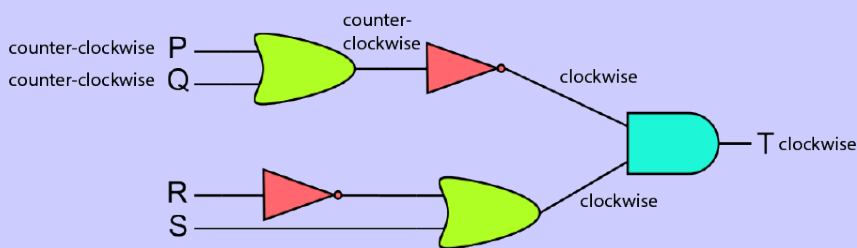
In order for T to turn clockwise, the input rods of block  must both turn clockwise.

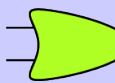


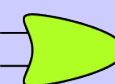
For the output rod of block  to turn clockwise, the input rod must turn counter-clockwise.



For the output rod of block  to turn counter-clockwise, both input rods must turn counter-clockwise.



This eliminates Options A and C. One way for block  to turn clockwise is for both input rods to turn clockwise. This would mean that S turns clockwise and R turns counter-clockwise, which is Option B.

The other way for block  to turn clockwise is for both input rods to turn in opposite directions. This would mean that S and R turn in the same direction. This eliminates Option D.

Connections to Computer Science

All computers contain *circuits*. Circuits are built using many small components called *logic gates*. A logic gate takes in *binary input* and produces *binary output*. Binary data is simply data with two possibilities often considered 1 or 0 or sometimes *true* or *false*. In this problem, our binary data is “clockwise” or “counter-clockwise”. The escape room lock is like a circuit where the blocks are the gates and binary data is transferred using the rods.

In this problem, the specific gates in the table are commonly called AND, OR, and NOT respectively. Their meaning corresponds to the formal and mathematical definitions of the words “and”, “or”, and “not”. It is an amazing fact that everything a modern computer can do, can be achieved by connecting gates in clever ways.

Country of Original Author

Romania

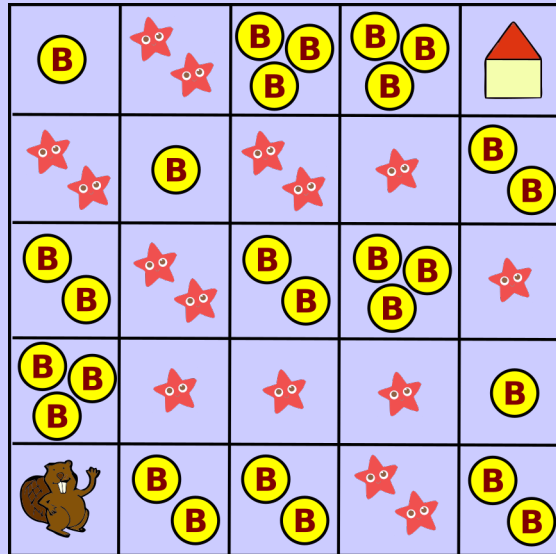


Part C

Coins and Monsters

Story

Tara starts in the lower left corner of the map shown. She can only move up or to the right as she makes her way to her home in the upper right corner.



Each square along the way contains either monsters , or coins . Tara collects all the coins along the path she follows. When she enters a square with monsters, she must give one coin to each monster in that square.

Question

If Tara starts with 10 coins, what is the maximum possible number of coins she could have with her when she arrives home?

- (A) 18
- (B) 19
- (C) 20
- (D) 22

Answer

(C) 20

Explanation of Answer

One way to solve the problem is by putting a value in each square of the map. Each positive value is the number of coins that Tara collects in a square. Each negative value is the number of coins that Tara gives up in a square. There are zeros in Tara's home and starting position.

1	-2	3	3	0
-2	1	-2	-1	2
2	-2	2	3	-1
3	-1	-1	-1	1
0	2	2	-2	2

We label the rows 1 to 5 from bottom to top and the columns 1 to 5 from left to right. In the beginning, (row 1, column 1), Tara has 10 coins. If she moves up, she gets 3 coins. If she moves to the right, she gets 2 coins. Tara can get to the square in (row 2, column 2) in two ways. If she comes from the left, she will have gathered 13 coins before reaching that square. If she comes from below, she will have gathered 12 coins before reaching that square. Since she is trying to gather as many coins as possible, she chooses the path that produces the larger value. So we record that she moves to (row 2, column 2) from the left through (row 2, column 1) and she has $13 - 1 = 12$ coins at that point.

In this way, row by row, for each square, we compute the total coins possible by coming from the left or from below and record the maximum, and which path should be followed. These computations are shown below. Each arrow indicates the direction(s) from which a maximum value can be derived. The blue values indicate one possible best path for Tara to reach home.

$13 + 1 = 14$	→	$14 - 2 = 12$		$13 + 3 = 16$		$17 + 3 = 20$	→	20
↑		↑		↑		↑		
$15 - 2 = 13$	→	$13 + 1 = 14$		$15 - 2 = 13$		$18 - 1 = 17$	→	$17 + 2 = 19$
↑		↑		↑		↑		↑
$13 + 2 = 15$	→	$15 - 2 = 13$	→	$13 + 2 = 15$	→	$15 + 3 = 18$	→	$18 - 1 = 17$
↑				↑				
$10 + 3 = 13$	→	$13 - 1 = 12$		$14 - 1 = 13$	→	$13 - 1 = 12$		$14 + 1 = 15$
↑				↑				↑
10	→	$10 + 2 = 12$	→	$12 + 2 = 14$	→	$14 - 2 = 12$	→	$12 + 2 = 14$

Connections to Computer Science

To solve the task, we have to calculate the best way of reaching each square. There are many possible paths to a square, and exploring all of them would be very tedious and time consuming. In our solution, we realize that each square can only be reached from two directions, left and below, so we can use the values calculated for those squares to derive the value for the current square. In this way, we can systematically compute the values for all squares, row by row from bottom to top, or column by column from left to right. This is an example of *dynamic programming*.

Dynamic programming is a type of *algorithm* that can solve *optimization problems*. The principle of dynamic programming is that it finds the optimal solution to a full problem by using intermediate results of the *subproblems*. The key to doing this quickly is that we can remember or store these intermediate results as they are calculated. Since the same intermediate result is used more than once, this speeds up the overall process considerably.

Country of Original Author

Indonesia



Aircraft Scheduling

Story

When an aircraft lands at an airport, it is assigned a designated airspace called a *corridor*. By ensuring that flights with similar landing times are in different corridors, air traffic controllers can help to avoid accidents.

At the Bebrasland airport, two aircraft cannot have the same corridor if their landing times are within 15 minutes of each other.

For example, if Flight #1 lands at 6:07 a.m., Flight #2 lands at 6:10 a.m., and Flight #3 lands at 6:25 a.m., then Flights #1 and #2 cannot be assigned the same corridor and Flights #2 and #3 cannot be assigned the same corridor. However, Flights #1 and #3 could be assigned the same corridor.

You are the Air Traffic Controller at the airport and your job is to assign corridors for the flights that are due to land at the times shown in the table.

Flight	Time
9W2400	7:00 a.m.
9W1321	7:21 a.m.
AI561	7:20 a.m.
AI620	7:18 a.m.
EK427	7:03 a.m.
SG147	7:12 a.m.

Question

What is the minimum number of corridors needed to ensure that the flights in the above table are assigned corridors according to the rules at the Bebrasland airport?

- (A) 2
- (B) 3
- (C) 4
- (D) 5

Answer

- (C) 4

Explanation of Answer

The table below has the flights and their landing times again, but this time in increasing order of landing time.

Flight	Time
9W2400	7:00 a.m.
EK427	7:03 a.m.
SG147	7:12 a.m.
AI620	7:18 a.m.
AI561	7:20 a.m.
9W1321	7:21 a.m.

The last four flights, SG147, AI620, AI561, and 9W1321 all land within nine minutes of each other. This means these four flights must be in separate corridors, so at least four corridors are needed. Flights EK427 and 9W1321 land 18 minutes apart, so they can be assigned the same corridor. As well, 9W2400 and AI561 land 20 minutes apart, so these two flights can also be assigned the same corridor. This gives an assignment of all six flights using four corridors. We also established that the flights cannot be accommodated using fewer than four corridors. Therefore, the minimum number of corridors needed is four.

Connections to Computer Science

In the above problem you have *conflicts* (two aircraft cannot have the same corridor if they are landing within 15 minutes of each other) and *resources* (corridors). You are asked to assign resources, ensuring that no conflicts arise. There are many problems of this type. When there are lots of objects (e.g. aircraft) a special approach may be needed to solve the problem quickly.

One approach is to model the problem as a *graph*. Here, the *vertices* are the flights and the *edges* represent the possible conflicts (two flights are joined by an edge when they land within 15 minutes of each other). We can think of our goal as *colouring* the vertices so that no two vertices connected by an edge are the same colour. That is, the colours correspond to corridors. Other applications of *graph colouring problems* include internet bandwidth allocation, pattern matching, sports scheduling, and exam timetabling. Computer scientists have designed advanced algorithms to solve these problems relatively quickly.

Country of Original Author

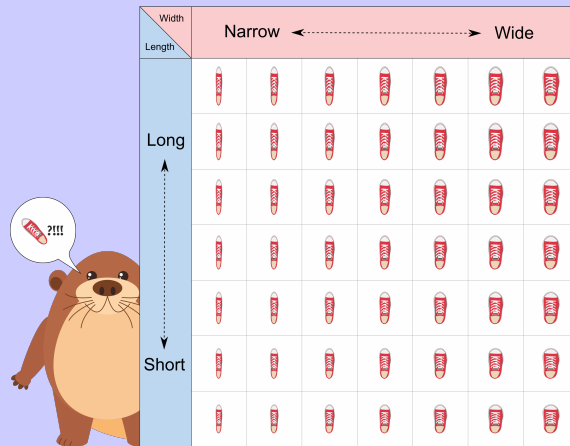
India



Buying Shoes

Story

A beaver would like to buy a pair of shoes imported from Triušisland. The shopkeeper tells the beaver that the shoes are arranged in a 7×7 grid so that in each row all shoes have the same length and different widths, and in each column, all shoes have the same width and different lengths. The shoes in each row are arranged from narrowest to widest going from left to right, and the shoes in each column are arranged from longest to shortest going from top to bottom.



The beaver is unfamiliar with the shoe sizes of Triušisland. However, by trying on a pair of shoes, the beaver can tell if the shoes are too wide, too narrow, or the correct width, as well as if they are too long, too short, or the correct length. A shoe *fits* if it is both the correct length and correct width.

The shopkeeper says:

“No matter what your shoe size is, I guarantee that there is a pair of shoes that fits you and a pair that fits can be identified by trying on no more than n pairs of shoes. You might not even have to try on a particular pair of shoes to know that it fits!”

Question

Assuming the shopkeeper is correct, what is the smallest possible value of n ?

- (A) 2
- (B) 3
- (C) 4
- (D) 5

Answer

(A) 2

Explanation of Answer

Divide the 49 pairs of shoes into 9 zones as shown.

The beaver should first try on the pair of shoes in the centre, which is the only pair of shoes in zone 5. From this, the beaver will be able to determine which of the 9 zones contains the pair of shoes that fits. For example, if the shoes in the centre are too wide but not long enough, the pair of shoes that fits must be narrower and longer, and hence, must be in zone 1.



The following table records which zone contains the pair of shoes that fits based on what is observed about the shoes in zone 5:

	too wide	correct width	too narrow
too short	zone 1	zone 2	zone 3
correct length	zone 4	zone 5	zone 6
too long	zone 7	zone 8	zone 9

If the shoes in zone 5 fit, then the beaver will stop searching. Otherwise, the search has been narrowed down to either 3 or 9 pairs of shoes. In either case, the next pair of shoes that the beaver should try on is the pair in the centre of the zone indicated by the table above.

For example, if the pair of shoes that fits is in zone 1, the beaver should try on the pair of shoes in the centre of zone 1 (marked by the number 1 in the picture). These shoes will be either too wide, the correct width, or too narrow. They will also be either too long, the correct length, or too short. There are 9 possible combinations of these pieces of information. Each combination corresponds to exactly one of the 9 pairs of shoes in zone 1. Thus, after trying on the pair of shoes in the centre, the beaver will know which pair of shoes fits (even though he may not have actually tried on the pair which fits).

Similarly, if the pair that fits is in zone 3, zone 7, or zone 9, then trying on the shoes in the centre will tell the beaver which pair of shoes fits.

If the pair of shoes that fits is in zone 2, then the beaver already knows the correct width. Trying on the pair of shoes marked with the number 2 will allow the beaver to determine the correct length and therefore the location of the pair of shoes that fits. Similar reasoning will work if the pair that fits is in zone 4, zone 6, or zone 8.

This shows Option A is correct but why is it impossible to find a way that will always identify a pair of shoes that fits while only allowing the beaver to try on a single pair of shoes? One reason is that no matter which pair of shoes the beaver tries on, there must be at least two pairs that are shorter or at least two pairs that are longer. Since the pair that fits could be among these shorter or longer shoes, at least one more pair must be tried on to identify a fitting pair.

Connections to Computer Science

In this problem, the shoes on display in the store are arranged in increasing order of width and length along multiple dimensions. Putting objects in an order such as this is called *sorting*. Once sorted, objects in a sequence can be found very quickly using the famous *binary search* algorithm. The idea is that you can jump into the middle of the sequence and if you don't find the object you are looking for, you can immediately conclude whether the object must be to the left or right of your current position. This process can be repeated until you find the desired object. Implemented correctly, this technique can find an object in a sequence of size 2^{100} by looking at no more than 101 of the objects!

This particular problem builds on the idea of binary search by extending it to two-dimensions.

Country of Original Author

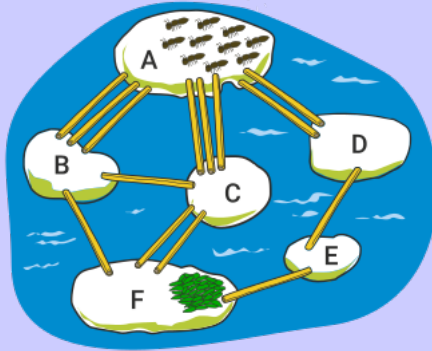
South Korea



Ants in a Swamp

Story

Ten ants are located on Stone A and seek to reach the food on Stone F.



The ants can only travel between the stones by walking along the straws joining the stones. No two ants can be on the same straw at the same time. It takes each ant 1 minute to travel from a stone to any other stone connected to it by a straw.

Question

What is the maximum number of ants that can reach the food on Stone F after 3 minutes?

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

Answer

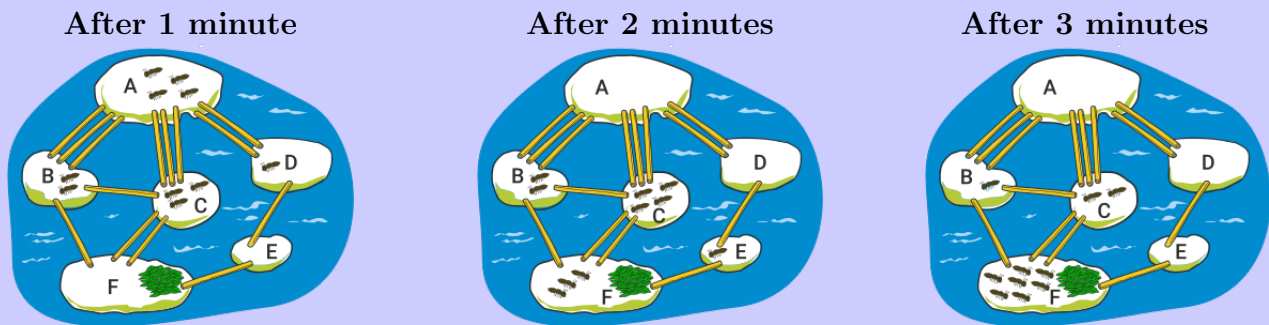
- (B) 7

Explanation of Answer

First, we argue that there cannot possibly be more than seven ants on Stone F after 3 minutes have passed. Note that ants must arrive at Stone F via Stone B, Stone C, or Stone E.

- After 1 minute, there cannot be any ants on Stone F. Any moving ants can only have made it as far as Stone B, C, or D over the first minute and so there are no ants on either Stone E or Stone F.
- After 2 minutes, there can be at most three ants on Stone F. At most one ant may have travelled from Stone B to Stone F, at most two ants may have travelled from Stone C to Stone F, and no ants may have travelled from Stone E to Stone F. (Remember that there could not have been any ants on Stone E after the first minute.)
- After 3 minutes, there can be at most seven ants on Stone F. There are at most three ants already on Stone F (from the previous bullet), and at most four new ants arriving. At most one ant may have travelled from Stone B to Stone F, at most two ants may have travelled from Stone C to Stone F, and at most one ant may have travelled from Stone E to Stone F.

We have shown that at most seven ants can reach the food after 3 minutes. Now we need to show that this number of ants can actually be achieved. One possible way for the ants to move in order for seven ants to reach the food after 3 minutes is shown below. Note that there are other ways that involve more ants moving at each step but we have shown that there cannot possibly be more than seven ants on Stone F after 3 minutes have passed.



It should be reasonably clear from the images what is happening during the first and third minutes, but here is an outline of what moves occur during the second minute:

- one ant travels from Stone B to Stone F,
- two ants travel from Stone C to Stone F,
- one ant travels from Stone A to Stone B,
- three ants travel from Stone A to Stone C, and
- one ant travels from Stone D to Stone E.

It can be verified that each of these moves is possible and would indeed produce the second image.

Note that we could have left three of the ants on Stone A and still have achieved the final result on Stone F. However, this particular sequence of moves also shows that exactly 4 minutes are needed for all ten ants to arrive at the food!

Connections to Computer Science

The goal here is to optimize the flow of the ants through the network so that as many ants as possible arrive at the food within three minutes. This is called an *optimization problem*. In particular, it is a *maximum flow problem*.

Ants that are not aware of the structure of the network they travel through will not be able to find the best solution. But an observer, that can see the network's structure, can find an optimal strategy. In this task, we assume that the ants are aware of the structure of the network, and as such will move along specific pathways. General algorithms have been devised by computer scientists allowing the maximum flow problem to be solved for very large networks. This has applications to problems involving the shipment of goods and finding solutions to what is known as the *baseball elimination problem*.

Country of Original Author

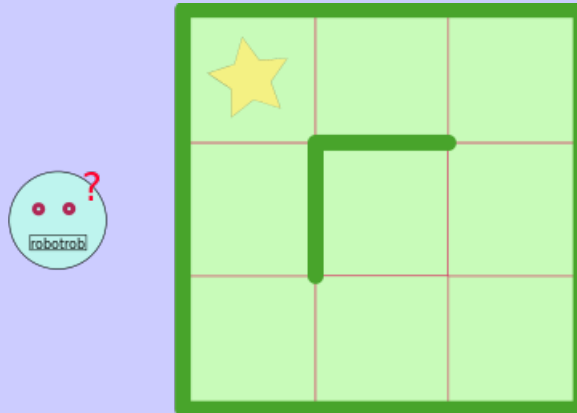
Lithuania



Recover My Robot

Story

Natasha lost her robot but she knows that it is in one of the nine squares in the following 3×3 grid.



Natasha can remotely send a sequence of commands to the robot. She can command it to move one square UP, LEFT, RIGHT, or DOWN. The robot will not move if there is a wall in the way. For example, if the robot is in the centre square and is told to move LEFT, it will ignore that command and move on to the next command. The walls are drawn on the picture by a thick (green) line.

Question

Which of the following sequences of commands guarantees that the robot will reach the square marked with a star no matter where the robot begins?

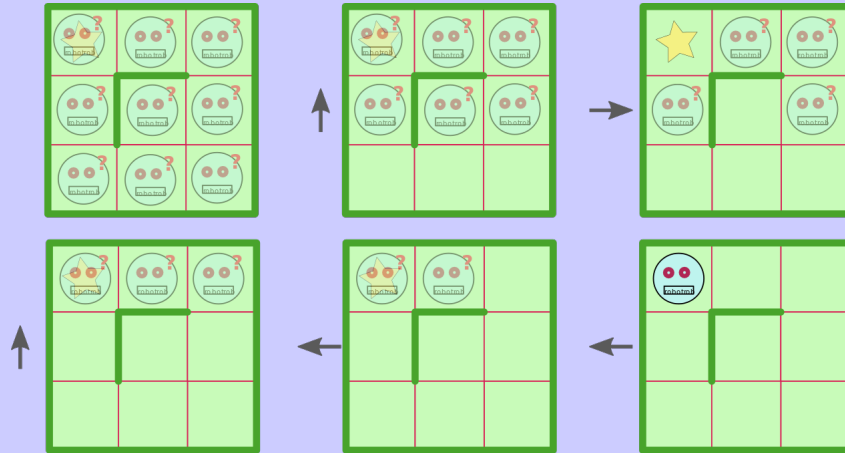
- (A) UP - UP - LEFT - LEFT - UP
- (B) RIGHT - UP - UP - LEFT - LEFT
- (C) DOWN - LEFT - LEFT - UP - UP
- (D) UP - RIGHT - UP - LEFT - LEFT

Answer

(D) UP - RIGHT - UP - LEFT - LEFT

Explanation of Answer

The diagram below demonstrates that executing the commands in Option D will get the robot to the desired square regardless of where it started. The top-left picture shows all nine starting positions. The top-middle picture shows the possible positions of the robot after the first command (UP) is executed, the top-right picture shows the possible positions of the robot after executing UP - RIGHT, and so on.



Regardless of where the robot started, it will end up in the square with a star.

Option A does not guarantee that the robot will reach the square marked with a star. For example, if the robot begins in the centre square, none of the commands in this sequence will have any effect.

Options B does not guarantee that the robot will reach the square marked with a star. For example, if the robot begins in the bottom-left square, this sequence will move the robot to the centre square after two commands and none of the later commands will have any effect.

Option C does not guarantee that the robot will reach the square marked with a star. For example, if the robot begins in the top-right square, this sequence will move the robot to the centre square after two commands and none of the later commands will have any effect.

Connection to Computer Science

A *computer program* is a sequence of instructions. In this way, it is like the sequence of commands controlling the robot in this problem. We often want to guarantee that our computer program ends in a certain way, much like in this problem when we want to guarantee that the robot ends in the same location no matter where it begins. This is particularly difficult when writing a computer program that involves a *user interface* which allows humans to enter input. We have to consider all possible inputs ensuring the program ends reasonably (without “crashing”) in each case.

Like the Light Buttons problem, this problem is also related to *deterministic finite automata*. The locations in the grid correspond to *states* and the robot *transitions* to different states based on its input. We are interested in inputs for which the robot always ends up in the same state no matter which state it begins in. This is the problem of finding *synchronizing words* for an automaton.

Country of Original Author

Russia

