UNIVERSITY OF WATERLOO

The CENTRE for EDUCATION in MATHEMATICS and COMPUTING

2019 Beaver Computing Challenge (Grade 5 & 6)

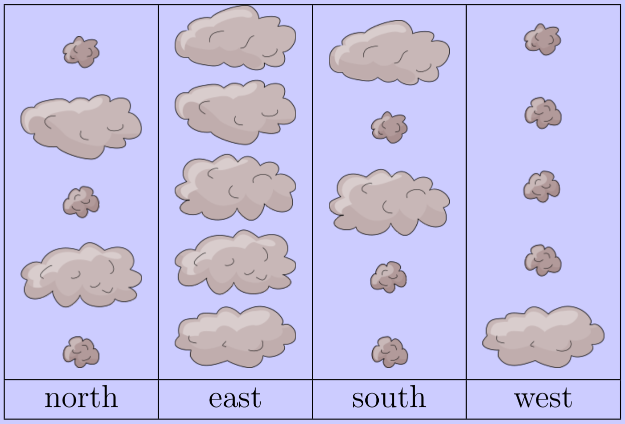
Questions, Answers, Explanations, and Connections

Part A

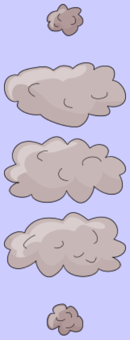
Cloud Communication

Story

Smoke signals were used by different groups of ancient peoples to send messages. A very simple code using small and large smoke clouds is given below.



Messages are read from top to bottom. The following message contains an error. Either one small cloud should be a large cloud or one large cloud should be a small cloud.



Question

What is the correct message?

(A) north

(B) east

(C) south

(D) west

Answer

(A) north

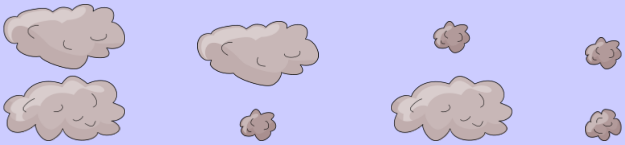
Explanation of Answer

If you change the third cloud from large to small you will get the code for north. Therefore, Option A is correct. Option B is not correct because the first and the last clouds would have to be changed. Option C is not correct because the first, second, and fourth clouds would have to be changed. Option D is not correct because all except the first cloud would have to be changed.

Connection to Computer Science

If you design sequences of symbols to be used for communication, it is better to choose the sequences so that messages can be reconstructed even if some parts of the message are lost or damaged. This can be done by sending more symbols than necessary. The extra data can essentially be used in place of any missing or damaged data. In computer science, systems like this are called *error-correcting codes* and they are used widely. For example, redundant information is used when sending music in a digital form. This means that digital music can be played correctly even if some data is corrupt.

In this problem, only two clouds are needed to represent four different messages:



Using five clouds of smoke added extra information which allowed us to determine the correct meaning even when the message contained an error.

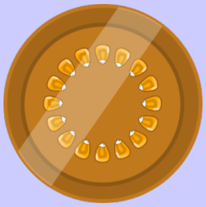
Country of Original Author

Switzerland

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.

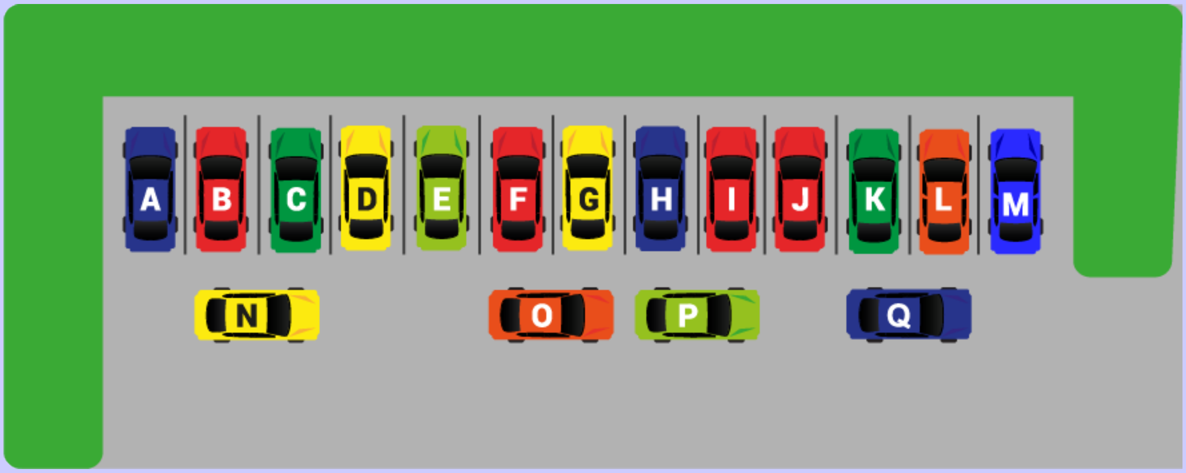
Country of Original Author

Switzerland

Push-Away Parking

Story

In the parking lot shown, each car is either parked in a parking space or in front of two parking spaces.



Cars that are parked in front of two parking spaces may be moved forward or backward in order to allow blocked cars to exit. For example, Car A is not blocked and can exit without any other cars moving; however, Car L is blocked by Car Q. If Car Q is moved, then Car L can exit.

Question

Which car **cannot** exit its parking space unless two different cars move?

(A) Car G

(B) Car H

(C) Car I

(D) Car B

Answer

(C) Car I

Explanation of Answer

If Car O moves, then Car G can exit. If Car P moves, then Car H can exit. If Car N moves, then Car B can exit. Car P must move in order for Car I to exit, but Car P cannot move far enough to allow Car I to exit without Car O or Car Q also moving. Therefore, at least two different cars must move in order for Car I to exit its parking space.

Connections to Computer Science

This problem is related to computer science in several ways.

First, one way to find a solution is to search through all the possibilities. Since there are not too many cars, it is possible to examine each car and determine which other cars need to move so that it can exit. This is called an *exhaustive search* or use of a *brute-force algorithm*.

More directly, the problem is immediately connected to *autonomous (automatic) parking algorithms*. It is more and more common for modern cars to do some of the driving for the driver of the car. This includes parking. Errors can have disastrous consequences and so extreme care must be taken when designing and implementing *real-time systems* like this.

Lastly, the idea of *blocking* is an important concept in the design of *operating systems* (e.g. Windows, MacOS, Linux). Multiple programs or *processes* need to run on the same machine and the operating system needs to minimize how often one process prevents (blocks) another from executing.

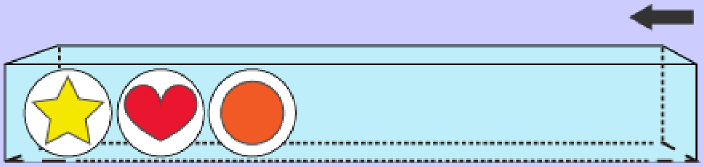
Country of Original Author

Germany

Box of Balls

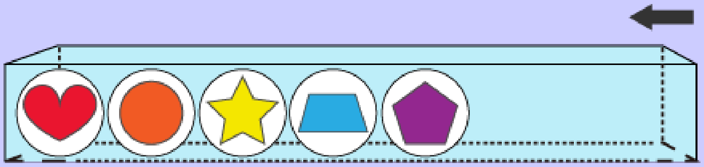
Story

A beaver has a box with an opening on the right-hand side.



At any time, the beaver can take out the rightmost ball from the box, or put in a new ball from the right. For example, if the beaver wants ▲ in between ❤ and ●, it needs to take out ●, put in ▲, and then put in ●.

Now suppose the beaver has five balls in the box as shown, and two balls (▲ and ■) out of the box.



The beaver wants the balls in the box to be in the order: ❤, ●, ★, ▲, ⏢, ■, ⬟.

Question

What should the beaver do?

(A) Take out ⬟, take out ⏢, put in ▲, put in ■, put in ⏢, and then put in ⬟.

(B) Take out ⬟, take out ⏢, put in ▲, put in ⏢, put in ■, and then put in ⬟.

(C) Take out ⬟, put in ▲, put in ■, and then put in ⬟.

(D) Take out ⬟, take out ⏢, put in ⬟, put in ⏢, put in ■, and then put in ▲.

Answer

(B) Take out ⬟, take out ⏢, put in ▲, put in ⏢, put in ■, and then put in ⬟.

Explanation of Answer

Option A is incorrect. These actions will lead to the result of ♥, ●, ★, ▲, ■, ⏢, ⬟.

Option C is incorrect. These actions will lead to the result of ♥, ●, ★, ⏢, ▲, ■, ⬟.

Option D is incorrect. These actions will lead to the result of ♥, ●, ★, ⬟, ⏢, ■, ▲.

Option B is correct. After the series of actions the balls will be in the beaver's desired order.

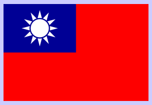
In the box, the first three balls on the left are ♥, ●, and ★. These satisfy the order given in the question, so there is no need to move these balls. The next ball should be ▲, but it is outside of the box. Therefore, the beaver needs to take out all balls on the right side of ★. The first ball that will be taken out is the one on the far right, which is ⬟. The next one will be ⏢. After taking out these two balls, the beaver can put the remaining balls into the box again in the order given in the question. It will put ▲ in first, then ⏢, then ■, and finally ⬟.

Connection to Computer Science

In this problem, you are asked to keep track of which balls are in the box as you consider different things the beaver can do. The key property is that the last ball put in the box is the first ball that must be taken out of the box if any other balls are to be removed. We say that this follows the *last-in first-out or LIFO* principle. In computer science, a collection of items that follows this principle is called a *stack*. It is one fundamental way that a collection can be stored. In particular, a *computer program* will often be broken into smaller pieces called *subroutines*. The order in which these subroutines are executed is often determined using a stack.

Country of Original Author

Taiwan

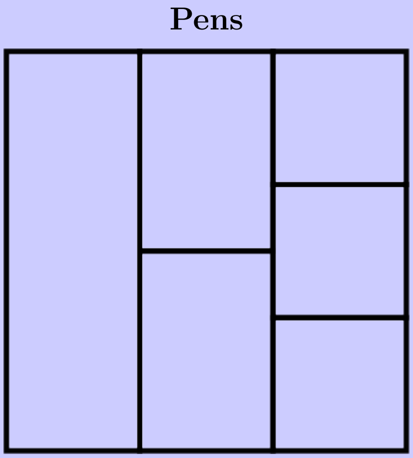


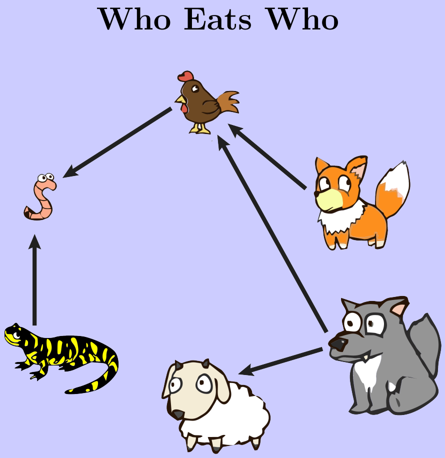
Part B

Koko's Animals

Story

Koko has six animals and needs to place each one in its own pen. Two animals cannot be placed in touching pens if one animal will eat the other. In the diagram shown, arrows point from an animal to all the other animals that it will eat.

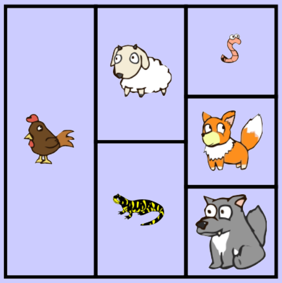


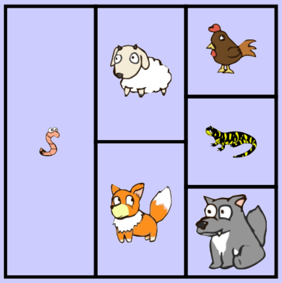


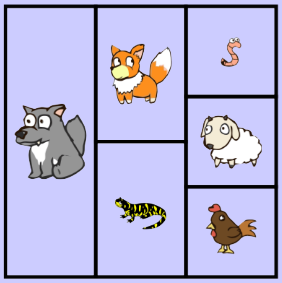
For example, the wolf will eat the chicken, but the wolf will not eat the worm.

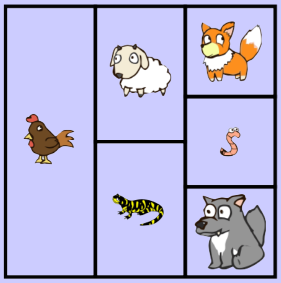
Question

Which of the following choices is **not** a good placement?

(A) 

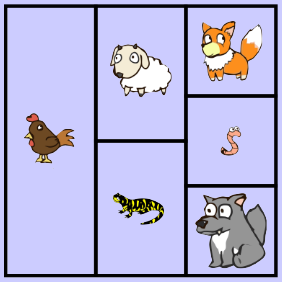
(B) 

(C) 

(D) 

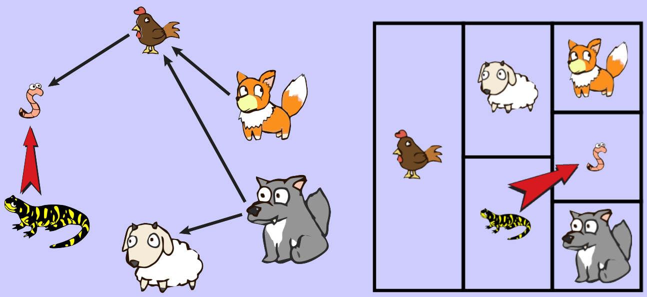
Answer

(D)



Explanation of Answer

Option D is not a good placement because the worm's pen is touching the salamander's pen and the salamander eats the worm.



For Options A, B, and C, consider the worm, chicken, and sheep. These are the three animals that can be eaten. If the worm's pen is not touching the salamander's or the chicken's, then the worm is safe. If the chicken's pen is not touching the wolf's or the fox's, then the chicken is safe. If the sheep's pen is not touching the wolf's, then the sheep is safe. Since the wolf, the fox, and the salamander cannot be eaten, they can be placed in any pen.

Connections to Computer Science

In this problem, you are asked to solve a *constraint satisfaction problem.* Constraints come from the fact that two animals cannot be placed in touching pens if one animal will eat the other. In general, a constraint satisfication problem involves finding a solution that obeys a given set of rules. Other well-known examples are Sudoku puzzles and trying to colour a map such that two neighbouring regions are not the same colour. These problems are often very difficult to solve. Trying all the possibilities often takes too long so computer scientists look for more advanced *search algorithms* that will produce a solution.

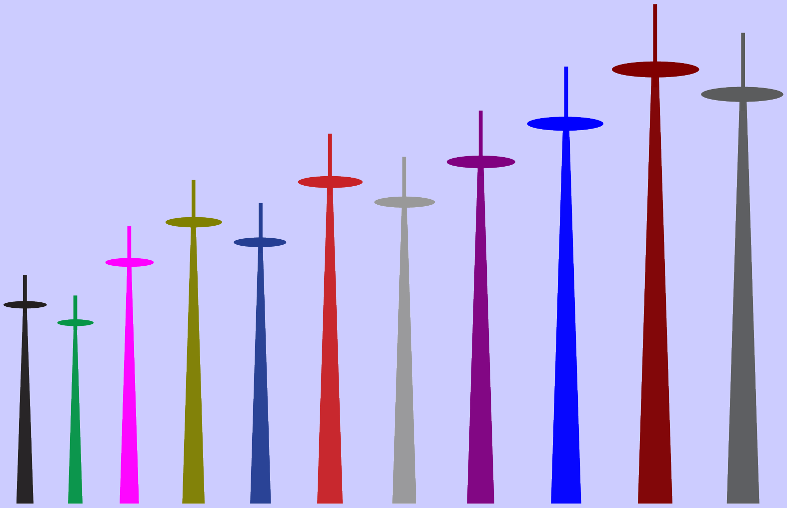
Country of Original Author

Indonesia

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).

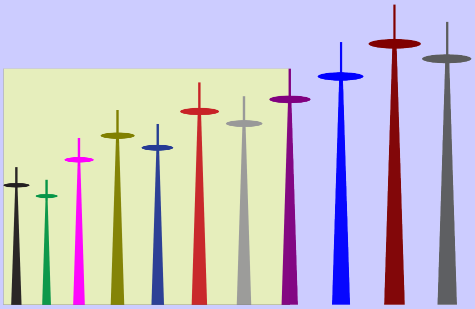


Figure 1

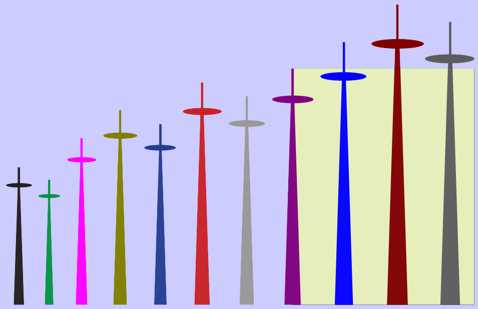


Figure 2

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).

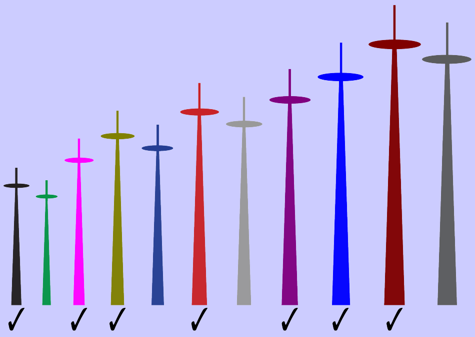


Figure 3

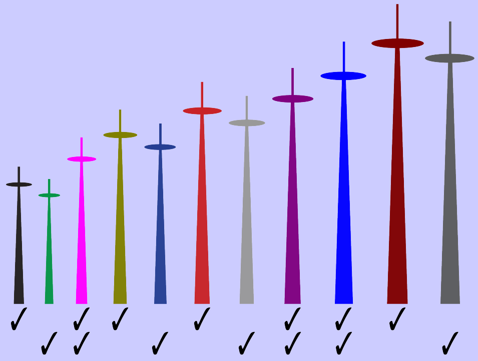
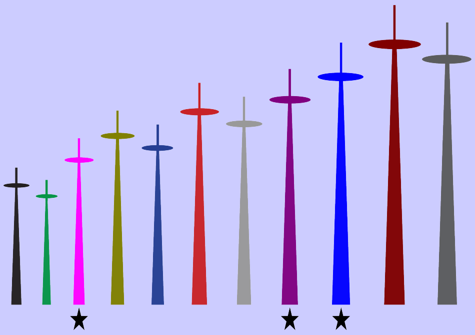


Figure 4

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.

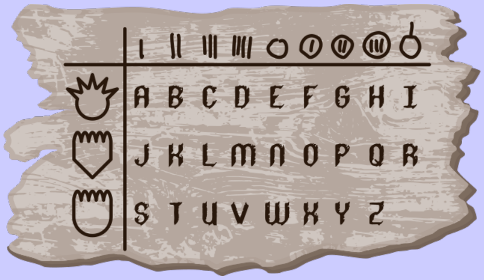
Country of Original Author

Canada

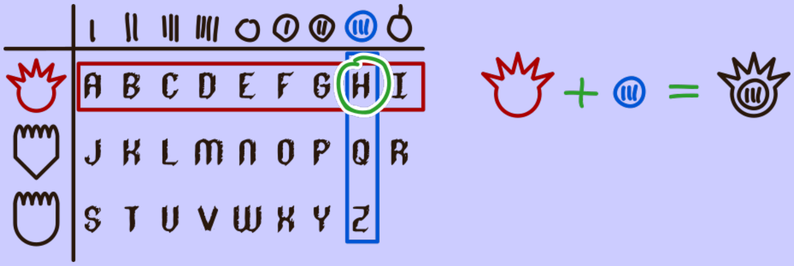
Ancient Code

Story

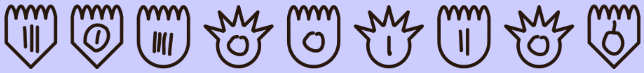
Beaver Cleveria discovered a table of symbols carved in wood.



After studying the table, Cleveria figures out that it is an ancient code. The symbol assigned to a row and the symbol assigned to a column are combined to form a single image. This image is the code for the letter where that row and column meet. For example, the letter H is encoded as shown:



Later, Cleveria sees the following coded message on a tree:



Question

What is the message?

(A) LOVEWATER

(B) SLEEPDAYS

(C) LOVEMYSUN

(D) CAREFORME

Answer

(A) LOVEWATER

Explanation of Answer

Cleveria can use the table to decode each image in the message.

The first image in the message is ✋. This corresponds to the third column and second row, and so it is the code for "L". This means that the message cannot be Option B nor D.

Cleveria notices that Options A and C begin with the same four letters, so she skips over to the fifth image. The fifth image is ✋ which is the code for "W". This means that the message cannot be Option C and must be Option A.

To verify that the message is in fact LOVEWATER, Cleveria can continue to decode all the remaining images.

Connections to Computer Science

Data security is an important issue in society. One of the methods to protect data from unauthorized persons is to *encrypt* it using a *cipher*. The study of these techniques is called *cryptography* and some historians believe it began approximately 3500 years ago. Some of these ancient systems involved replacing each letter by another letter.

In this problem, new symbols are created for letters of the alphabet in a way that allows us to easily produce coded messages as long as we know the secret table. The secret table is called a *private key*. The security of the system depends on keeping this table secret. Someone who does not have access to the private key might try to break the code by looking at how often each letter appears and trying to observe patterns. Studying possible attacks on a cipher is called *cryptanalysis*.

Many of the world's brightest computer scientists and mathematicians work for governments and agencies where their job is to perform cryptanalysis and design secure *cryptosystems*.

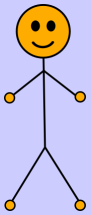
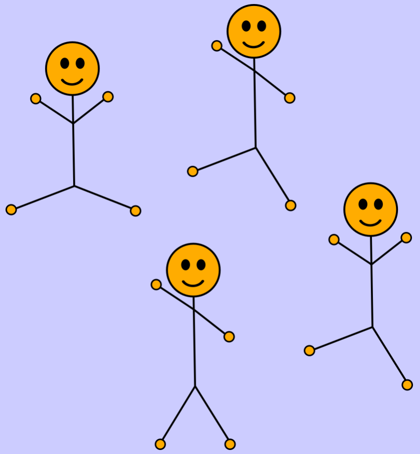
Country of Original Author

Switzerland 🇨🇭

Beaverumba

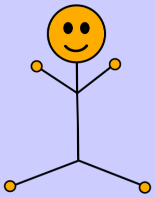
Story

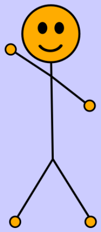
There are five positions in the beaverumba dance. Each position after the first involves moving either exactly one arm or exactly one leg from the position before it. Robert remembers the first position of the dance but forgets the correct order of the other four positions.

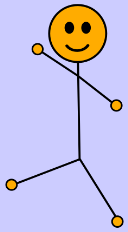
	
first position	other four positions

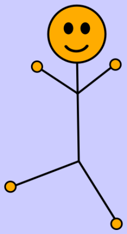
Question

What is the third position of the dance?

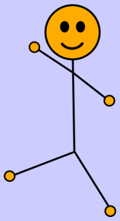
(A) 

(B) 

(C) 

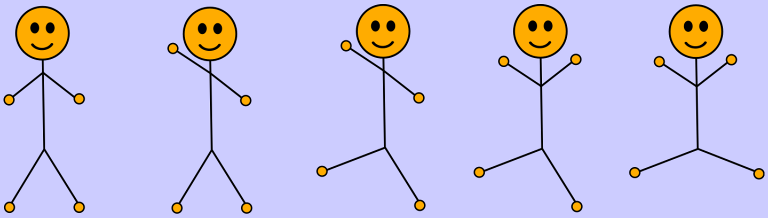
(D) 

Answer

(C) 

Explanation of Answer

Option B must be the second position of the dance because only it can be reached from the first position by moving exactly one limb. Similarly, Option C must be the third position of the dance because other than the first position, only it can be reached from the second position by moving exactly one limb. Continuing in this way, we can determine that the correct order of the positions is as shown below.



Therefore, Option C is the third position of the dance.

Connection to Computer Science

In our solution to this problem, we considered how many moves different positions are from each other. This is an application of *hamming distance* which is important in *information theory*. Information theory strongly influences *error-correcting codes* and *cryptography*.

The problem also involves determining the correct sequence of steps in the dance. There is an analogy between these steps and computer *instructions*. However, in some ways, your task in this problem is the "opposite" of *computer programming*. When writing a computer program, you need to write a sequence of instructions that will yield the desired *output*. In this problem, we have the desired output and are asked for the corresponding sequence of steps. This "opposite" activity is also something computer programmers need to do all the time when *debugging* their programs. That is, if the output they are getting is not what they expected or wanted, they need to figure out what sequence of instructions are producing the incorrect output.

Country of Original Author

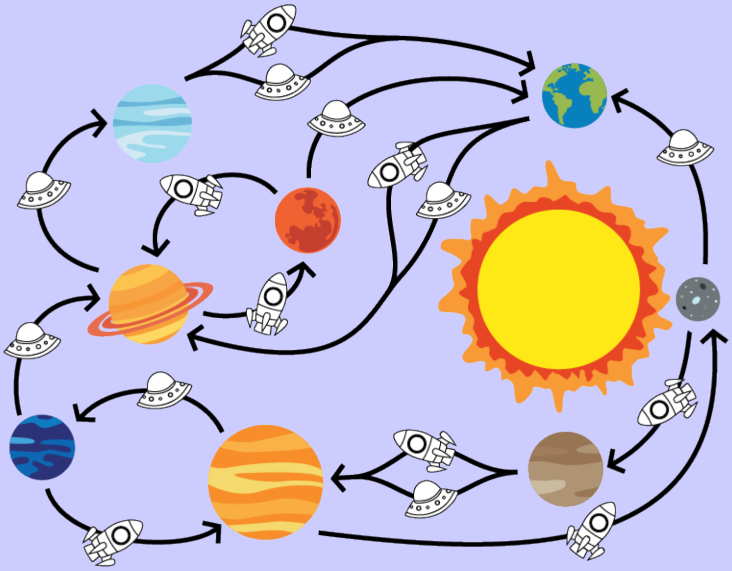
Vietnam

Part C

Space Travel

Story

An astronaut's map is shown. It shows all of the possible travel routes between planets. For each route, it shows whether it can be travelled by rocket 🚀 or by spaceship 🛸, or by both rocket and spaceship.



For example, 🚀🛸🛸 is a list of space vehicles that will take the astronaut from 🟤 to 🪐.

Suppose the astronaut wants to travel from 🔵 to 🌍.

Question

Which of the following is **not** a list of space vehicles that the astronaut could take?

(A) 🛸🛸🚀

(B) 🚀🛸🛸🛸🚀

(C) 🚀🚀🛸

(D) 🚀🛸🚀🛸

Answer

(D) 🚀🛸🚀🛸

Explanation of Answer

From 🔵 a rocket will take the astronaut to 🟠 and then taking a spaceship forces the astronaut to return to 🔵. So Option D forces the astronaut to do this loop twice and is not a way to end at 🌍.

For Option A, the astronaut can take a spaceship to 🪐 and then 🩵 before taking a rocket to 🌍. For Option B, the astronaut can do one loop as in Option D and then go to 🌍 as in Option A.

For Option C, the astronaut can take a rocket to 🟠 and then 🌑 before taking a spaceship to 🌍.

Connections to Computer Science

The idea behind this problem is that the planets correspond to different *states* the astronaut can be in, and the astronaut can change states by following a *transition* labelled with the appropriate space vehicle. In general, this type of object is called a *deterministic finite automaton* or *DFA* for short.

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). Many devices with buttons (e.g. a coffee machine or automated teller machine) move between states with the press of a button. Careful design of states and transitions allow computer programmers to program devices like this in efficient and reliable ways.

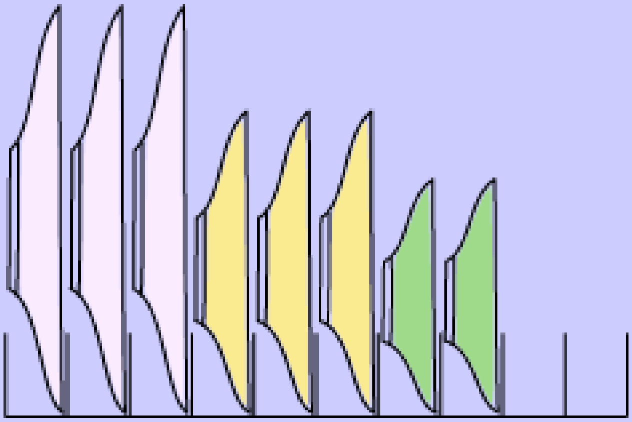
Country of Original Author

Slovenia

Plates

Story

A beaver believes plates are only arranged properly if all the large plates are on the left, followed by all the medium plates, followed by all the small plates. For example, the beaver believes the three large plates, three medium plates, and two small plates shown are arranged properly.



The beaver would like to add a large plate and arrange them properly.

Question

Of the eight original plates, what is the fewest number of plates that must be moved?

(A) 2

(B) 3

(C) 4

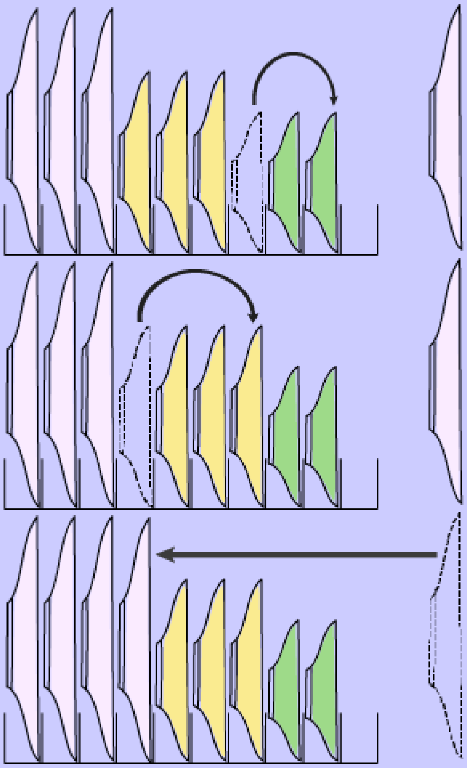
(D) 5

Answer

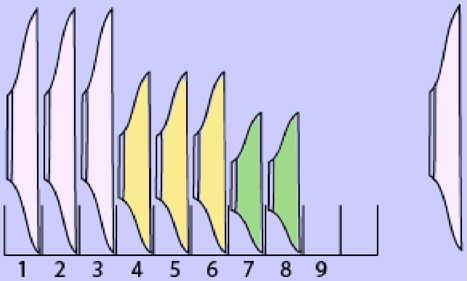
(A) 2

Explanation of Answer

The task can be solved by moving two of the original plates. The sequence of moves is shown below:



To see that the task cannot be done by moving only one of the original plates, consider the picture below:



Once the plates are arranged properly, there must be a medium plate in slot 7, and a small plate in slot 9. Neither of these conditions are satisfied initially, which means at least two of the original plates must move.

Connection to Computer Science

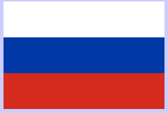
In this problem, plates are stored in a sequence and you want to arrange them in a specific way by moving the plates one at a time. It is very common for data to be sorted on a computer in a sequence and to then perform operations changing the order of the data in this sequence. The types of operations we expect to perform can affect how we might choose to store the sequence in memory.

In general, there are two main ways to store a sequence in memory. One way involves storing each data item in an *array* which means that the items are right beside each other or "neighbours" in computer memory. This means that it is easy to find a data item in a given position. However, trying to insert an item into the middle might involve shuffling lots of other items which takes lots of time. When this happens, one could look for a different set of movements as in the solution to this problem.

Another way to store sequential data in memory is to use a *linked list*. In a linked list, items are "chained together" but not necessarily one beside the other. This makes it quick to insert items in the middle but hard to jump to an item at a given position.

Country of Original Author

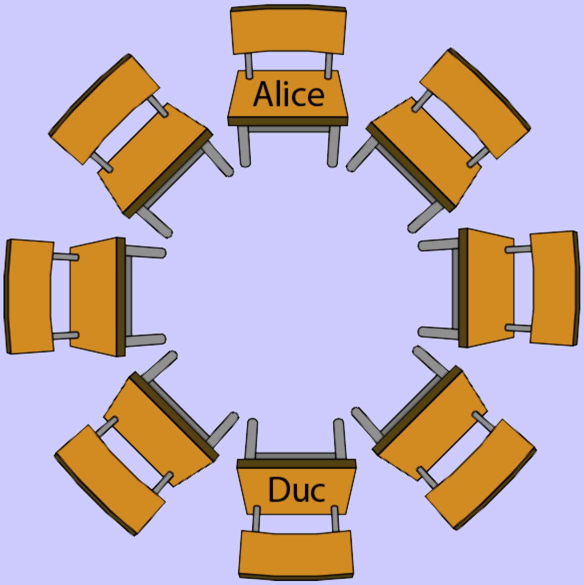
Russia



Seating Plan

Story

Berto and seven of his friends are sitting in a circle. They are all facing inwards.



We know the following facts about where the friends are sitting:

1. Alice is sitting directly across from Duc, as shown.
2. Greta and Eugene are both sitting beside Haakim.
3. Franny is not sitting beside Alice or Duc.
4. There is someone who is sitting next to both Greta and Chika.
5. Eugene is beside Duc, on Duc's left.

Question

Which of these orders of friends, in a clockwise manner, is correct?

(A) Alice, Berto, Greta, Duc, Chika, Eugene, Franny, Haakim

(B) Alice, Greta, Haakim, Eugene, Duc, Berto, Franny, Chika

(C) Alice, Chika, Franny, Berto, Duc, Eugene, Haakim, Greta

(D) Alice, Haakim, Eugene, Greta, Duc, Franny, Berto, Chika

Answer

(C) Alice, Chika, Franny, Berto, Duc, Eugene, Haakim, Greta

Explanation of Answer

Alice and Duc are sitting across from each other, so there are three people between them in either direction. Part of the seating arrangement is

Alice, _____, _____, _____, Duc, _____, _____, _____.

We know that Eugene is sitting next to Duc on Duc's left, so we can place Eugene on Duc's left. Remembering that the friends are facing inwards, Eugene should go in our clockwise seating like this:

Alice, _____, _____, _____, Duc, Eugene, _____, _____.

Haakim is sitting between Greta and Eugene, so Haakim and Greta must be sitting as follows:

Alice, _____, _____, _____, Duc, Eugene, Haakim, Greta.

Franny must be in the middle of the three remaining positions in order to satisfy the third condition (Franny is not beside Alice or Duc), which gives us the following:

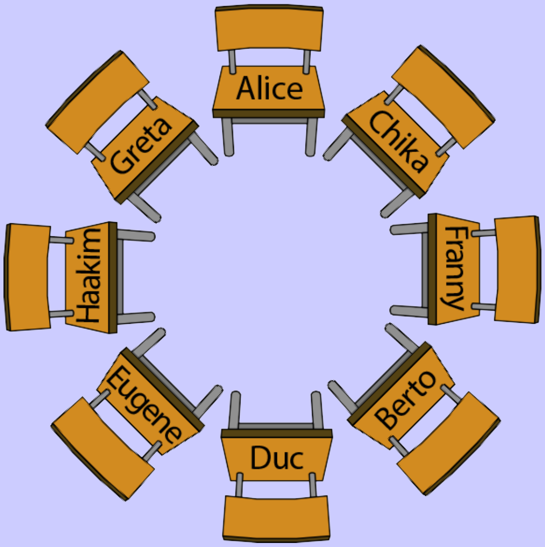
Alice, _____, Franny, _____, Duc, Eugene, Haakim, Greta.

There is one person between Greta and Chika, which means Chika must go between Alice and Franny:

Alice, Chika, Franny, _____, Duc, Eugene, Haakim, Greta.

Berto goes in the final position:

Alice, Chika, Franny, Berto, Duc, Eugene, Haakim, Greta.



Connections to Computer Science

This is a *logic* problem. We are given a set of *statements* and asked to determine an order of the friends for which each statement is true. A statement is a sentence that is either true or false. Logic is the study of statements, determining if they are true, and arguing whether they are true or false.

In logic, we can take a simple statement like “Franny is not sitting beside Alice” as well as another simple statement like “Franny is not sitting beside Duc” to create a *compound statement* like “Franny is not sitting beside Alice AND Franny is not sitting beside Duc”. An even more complex statement can be formed by *negating* this statement to get “Franny is not sitting beside Alice OR Duc”. Can you see why this is equivalent to “ Franny is not sitting beside Alice AND Franny is not sitting beside Duc”?

Knowing how logical statements can be rewritten, combined, or simplified is a very useful skill in computer science.

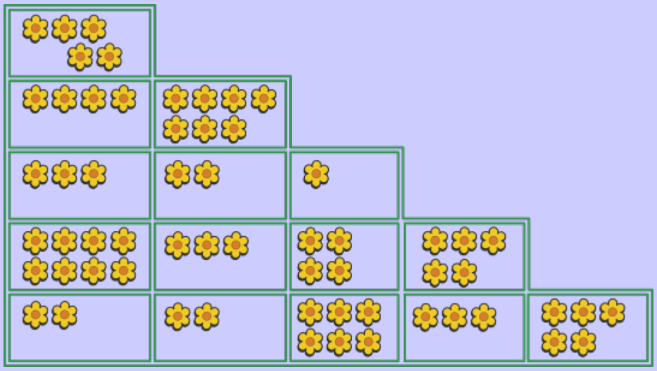
Country of Original Author

Canada

Picking Flowers

Story

A beaver visits 9 of the 15 sections in the garden shown. It begins at the top left section and ends at the bottom right section. The beaver only moves down or right and it picks all the flowers in each section it visits.



Question

What is the maximum number of flowers that the beaver can pick?

(A) 39

(B) 38

(C) 58

(D) 41

Answer

(D) 41

Explanation of Answer

By going down, down, down, right, right, down, right, and right, the beaver can pick $5 + 4 + 3 + 8 + 3 + 4 + 6 + 3 + 5 = 41$ flowers. This is Option D and rules out Options A and B. We still have to rule out Option C. The beaver visits nine sections no matter which route it takes because from the top left section, it must move down a total of four times and right a total of four times. It will pick $5 + 4 = 9$ flowers in the first two sections and $3 + 5 = 8$ in the last two sections. So to pick 58 flowers in total, it must pick $58 - 9 - 8 = 41$ flowers in the five "middle" sections. This is not possible because the most flowers in any section is 8 and $5 \times 8 = 40$. So, of the choices given, the correct answer must be D.

If we want to make sure that 41 is the maximum number of flowers among all possibilities, we can observe that the maximum number of flowers that can be picked by the beaver upon reaching a section is the number of flowers in that section **plus** the maximum of

- the total number of flowers that can be picked upon reaching the section to the left, and
- the total number of flowers that can be picked upon reaching the section above.

We can compute these values starting at the top left. These computations are shown below. Each arrow pointing to a section indicates whether the maximum comes from the section to the left or comes from the section above. When we reach the bottom right section, we compute the value 41 which tells us that 41 is the maximum number of flowers among all possibilities. To pick this maximum number, the beaver can follow the path shown by the red arrows.

$$\begin{tikzcd}
\boxed{5} \arrow[d, red] & & & & \\
\boxed{4+5=9} \arrow[r] \arrow[d, red] & \boxed{7+9=16} \arrow[d] & & & \\
\boxed{3+9=12} \arrow[d, red] & \boxed{2+16=18} \arrow[r] & \boxed{1+18=19} & & \\
\boxed{8+12=20} \arrow[r, red] \arrow[d] & \boxed{3+20=23} \arrow[r, red] \arrow[d] & \boxed{4+23=27} \arrow[r] \arrow[d, red] & \boxed{5+27=32} & \\
\boxed{2+20=22} & \boxed{2+23=25} & \boxed{6+27=33} \arrow[r, red] & \boxed{3+33=36} \arrow[r, red] & \boxed{5+36=41}
\end{tikzcd}$$

Connection to Computer Science

Our full solution to this difficult problem 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 using intermediate results to *subproblems*. In this problem, we are looking for the maximum number of flowers that can be picked from the starting section to any of the intermediate sections. 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

Romania