Paired Socks

Question:

A clothing store has a large bin of socks. Unfortunately, each sock has been separated from its corresponding matching sock.

Thankfully, each individual sock is tagged with a special code, composed only of `0`'s and `1`'s. Two socks are paired together if their codes are different in every digit: where one code has `1`, the other code has `0` and vice versa. For example, a sock with the code `0110` is paired with the sock with code `1001`.

Which one of the following pairs of codes is not a correct pairing of socks?

Possible Answers:

`1010 - 0101`
`0011 - 1100`
`1011 - 0100`
`1101 - 0001`

Correct Answer:

`1101 - 0001`

Explanation of Correct Answer:

The pair that should match `1101` is `0010`, which is not the same as `0001`.

Connections to Computer Science:

Binary digits are the fundamental way that information is stored on a computer. Since electricity can be measured in two states ("on" and "off"), binary digits can be used to represent these two states (for example, "off" could be `0`, and "on" could be `1`).

In this question, we are concerned with the *one's complement* of a binary number. The one's complement is defined as switching `0` to `1`, and `1` to `0`.

The one's complement can be used to represent negative numbers in binary. For example, if we represent the numbers from 0 to 7 using 4 binary digits ("BInary digiTS" are called *bits*), then `0110` represents the number 6 (i.e, reading from the right, this is $0 \cdot 1 + 1 \cdot 2 + 1 \cdot 4$). We can also represent the numbers -7 to -1 using these same four bits. For example, we can represent -6 as `1001`, which is the one's complement of `0110`. If we add these two numbers, we get `1111`, which is the one's complement of 0, which is what we would expect when adding 6 and -6.

Missing Step

Question:

Beaver Johnny received a task from his computer science teacher. He has two cards, card A and card B. On card A is a number, and on card B is a different number. Each card can only hold one number at a time. He has to exchange the numbers on card A and card B by the following three steps:

$$A \leftarrow A + B;$$

$$???????$$

$$A \leftarrow A - B;$$

The first step, $A \leftarrow A+B$, means that we should add the number on card A and the number on card B and replace the number on card A with this sum.
The third step, $A \leftarrow A - B$, means that we should subtract the number on card B from the number on card A, and replace the number on card A with this difference.
However, Johnny has forgotten the second step.
Which of the following is the correct second step?

Possible Answers:

$A \leftarrow A + B$;
$B \leftarrow A - B$;
$B \leftarrow B - A$;
$A \leftarrow B$;

Correct Answer:
$B \leftarrow A - B$;

Explanation of Correct Answer:
Consider the values recorded on each card:

	Value on card A	Value on card B
Initially	X	Y
After $A \leftarrow A + B$;	$X + Y$	Y
After $B \leftarrow A - B$;	$X + Y$	$(X + Y) - Y = X$
After $A \leftarrow A - B$;	$(X + Y) - X = Y$	X

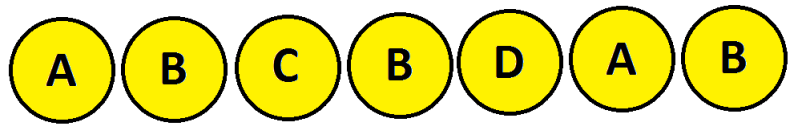
Notice at the end, the values on the two cards have been swapped.
None of the other possible steps would achieve this result.

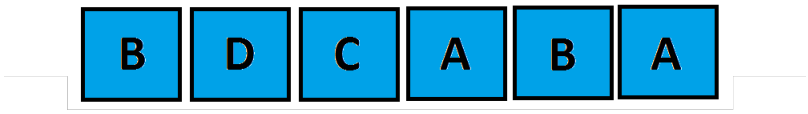
Connections to Computer Science:
This problem is focussed on following a basic *algorithm* or sequence of steps to solve a particular problem. The challenge in this problem is determine which of the possible steps make the algorithm correct.
We also use *abstraction* in solving the problem. By abstraction, we mean that we ignore some details of the problem in order to make it easier to understand. The details we ignore here are the actual numbers on the cards. Rather, in our reasoning, we generalize the problem (to use the values X and Y on the two cards initially) in order to make the analysis both more thorough and general.

Best Match

Question:

You can connect a circle and a square that both have the same letter in them using a *straight* line.





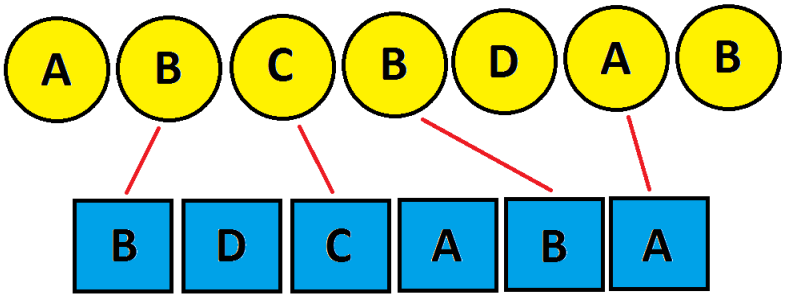
What is the maximum number of such connections you can make without crossing any lines?

Possible Answers:

2
3
4
5

Correct Answer:
4

Explanation of Correct Answer:
Here is one possible way to have 4 connections:



In order to see that there is no way to make 5 connections notice:

- we cannot connect connect both pairs of A nodes, since that forces us to have at most 3 connections (ABA);
- we cannot connect both the D and C nodes at the same time.

Thus out of the 6 blue boxes, we have can have at most 4 connections (i.e., at least one A cannot be connected, and at least one from {D, C} cannot be connected).

Connections to Computer Science:
This problem is known as the *longest common-subsequence* (LCS) problem. The solution to this problem is used in many software tools for comparing two sequences of data (say text files or DNA sequences) for commonality. The solution to the LCS problem is one involving *dynamic-programming*, which is a solution technique that involves writing down a *recurrence relation* and then solving subproblems to build a solution to the larger problem. A recurrence relation is a relationship or equation which involves a solution to a larger problem in terms of a solution to a smaller problem. For example:

$$\begin{aligned} T(n) &= T(n-1)+n \\ T(1) &= 1 \end{aligned}$$

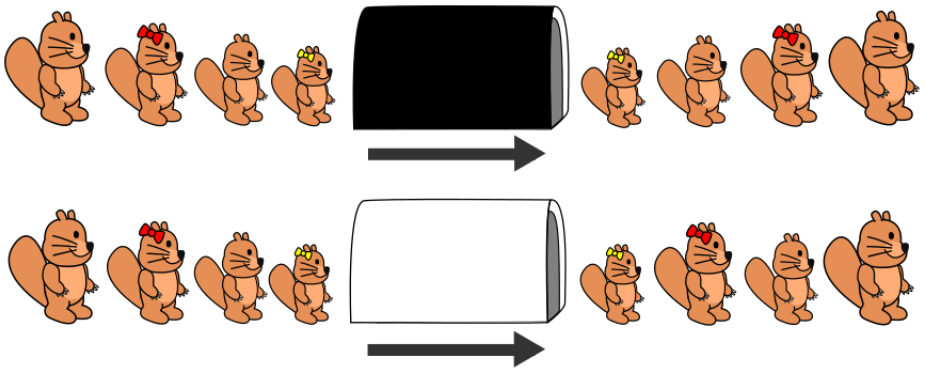
would have a solution $T(n) = \frac{n^2}{2} + \frac{n}{2}$, which can be arrived at by substituting, as in:

$$\begin{aligned} T(n) &= T(n-1)+n \\ &= (T(n-2)+n-1)+n \\ &= (T(n-3)+n-2)+n-1+n \\ &\vdots \\ &= 1+2+3+\cdots+n-2+n-1+n \\ &= \frac{n^2}{2}+\frac{n}{2} \end{aligned}$$

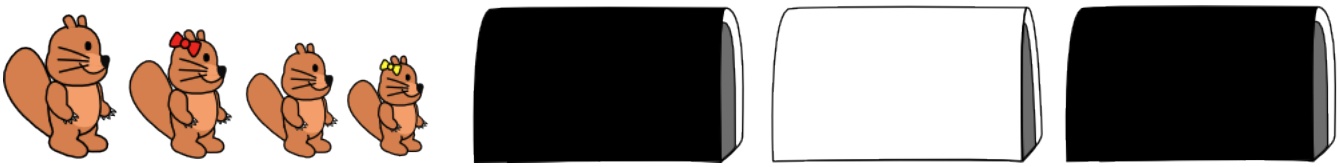
Zebra Tunnel

Question:

There are two kinds of tunnels in BeaverLand.
When a sequence of beavers enter a black tunnel one after the other, they come out in reverse order.
When a sequence of beavers enter a white tunnel one after the other, they come out with the first and the last beaver swapped.

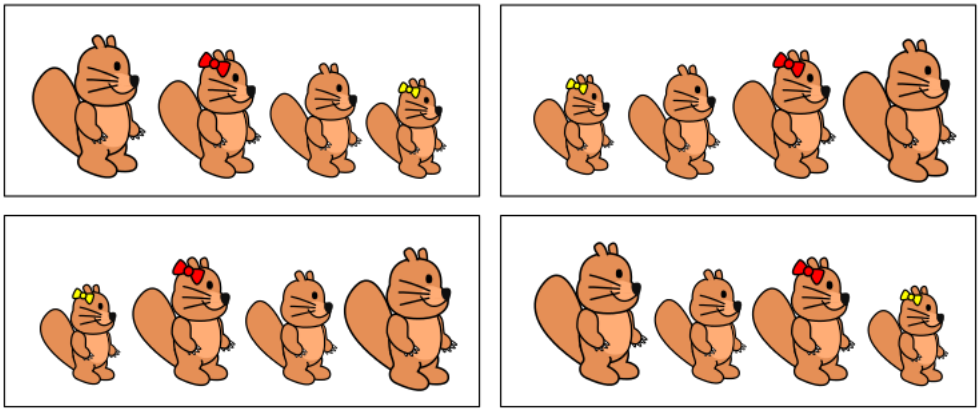


A Beaver family goes through three tunnels as shown.

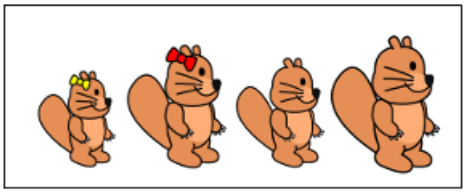


In what order are they arranged in when they come out of the last tunnel?

Possible Answers:



Correct Answer:



Explanation of Correct Answer: If the beavers are numbered (4,3,2,1) as the enter the tunnel (where 1 is the shortest beaver, who enters first, and 4 is the tallest beaver who enters last), then:

- leaving the first black tunnel, the order would be (1,2,3,4) when entering the white tunnel
- leaving the white tunnel, the order would be (4,2,3,1) when entering the second black tunnel
- leaving the second black tunnel, the order would be (1,3,2,4)

It is worth noticing that the two black tunnels could be ignored, since the reverse of the reverse leaves the list unchanged. So, the sequence (4,3,2,1) entering the white tunnel would cause (1,3,2,4) to leave the white tunnel.

Connections to Computer Science:

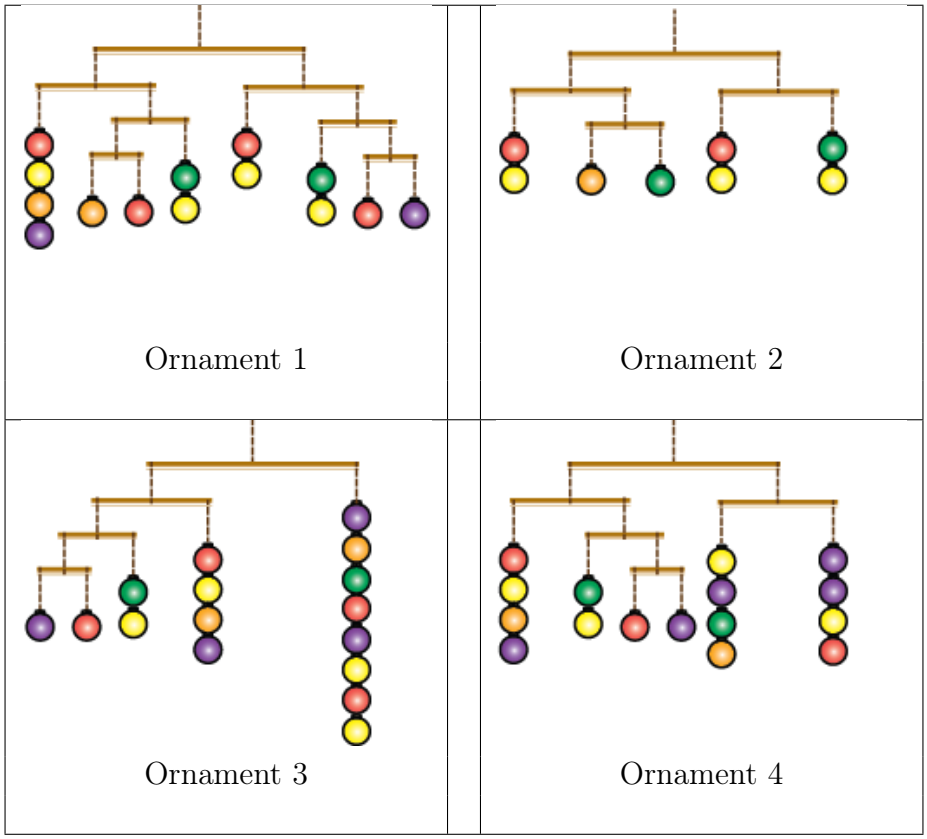
One fundamental step in understand an *algorithm* (i.e., a finite sequence of steps to accomplish some goal) is to try to simplify, abstract and reason about what the algorithm does. In this problem, we can either *abstract* away the details of the problems (by numbering the beavers in some reasonable way) or *simplify* the problem by noticing that two reverse operations leave the list unchanged.

This problem also highlights the concept of storing information for later retrieval. The white tunnel is an example of a *Last In First Out* (LIFO) structure, which is more commonly known as a *stack*. A stack of books (or plates, or papers) works by having the most-recently placed item being the top-most item, which will be the first to be removed. Another data structure that works differently than a stack is a *queue*, which is a *First In, First Out* (FIFO) structure. A line-up at the cashier in a store works as a queue.

Ornaments

Question:

Holidays are almost here and Katy Beaver is preparing four ornaments.
Katy plans to make them from wooden sticks (which will be horizontal pieces), threads (to connect balls to sticks) and small balls (to make the ornament look pretty). She has prepared four sketches. Unfortunately, one of them will not hang as nicely as she imagines.



We can ignore the mass of sticks and threads. All balls have equal mass. An ornament is *well-balanced* if the number of balls hanging on the left side and the right side of each stick is equal.
Which ornament is not well-balanced?

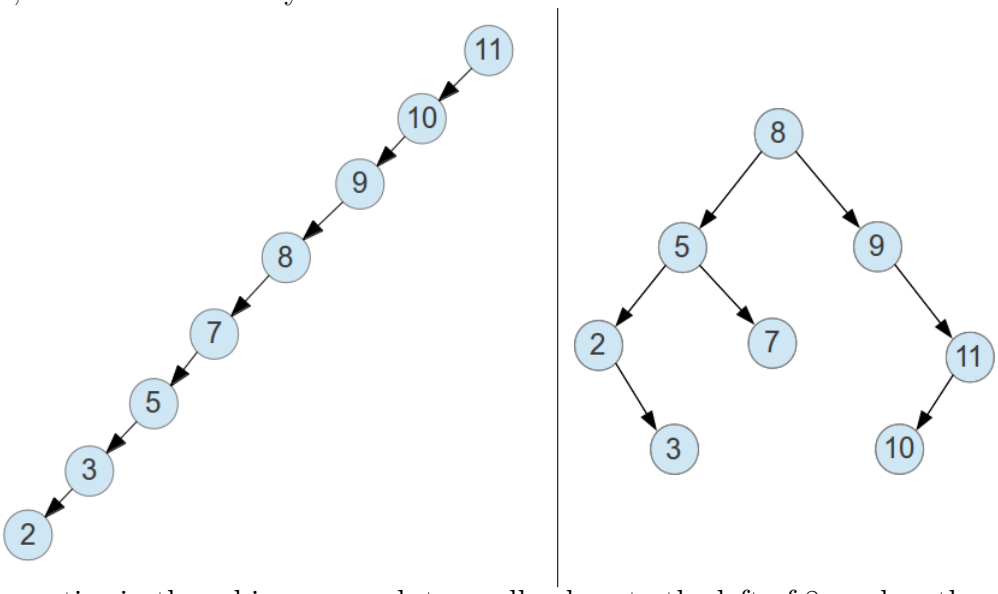
Possible Answers:

Ornament 1
Ornament 2
Ornament 3
Ornament 4

Correct Answer:
Ornament 1

Explanation of Correct Answer:
The left side of Ornament 1 is balanced; however, on the right side, there are two balls on the left side and four balls on the right.

Connections to Computer Science:
This question discusses *balanced trees*.
In computer science, information is often stored in a *binary search tree*, where we have a *root node* (at the "top" of the ornament) and two ornaments on the left and right. We add the restriction that for *every* value stored in the binary search tree, the values in the left tree are less than the node value, and the values in the right tree are greater than the node value.
For example, two different binary search trees with the same values are shown below:



For example, notice in these binary search tree, all values to the left of 8 are less than 8, and all values to the right of (if there are any) are larger than 8.
Binary search trees allow us to store a very large amount of information in an ordered manner.
However, the time to search for information is related to how many steps from the root to the node which contains the information. In other words, the height of the tree is the *worst case search time*.
Therefore, the more balanced a tree is, the more nodes that can be stored closer to the root node, making the tree shorter and the average search time to find a value smaller. In the pictures above, the binary search tree on the right is much more balanced than the binary search tree on the left, and the time to find elements starting from the root is much shorter in the balanced tree than in the unbalanced tree.

Deletion Game

Question:

Bob the Beaver plays a game against Sam the Squirrel.
At first Bob removes four numbers from the list 1, 2, 3, 4, 5, 6, 7, 8.
Then Sam deletes two of the remaining numbers.
Bob wants the positive difference between the remaining two numbers to be as large as possible. Sam wants this positive difference to be as small as possible. Both Bob and Sam know what the other player is trying to achieve.
What is the positive difference between the last two numbers if both Bob and Sam play as well as possible?

Possible Answers:

1
2
3
4

Correct Answer:
2

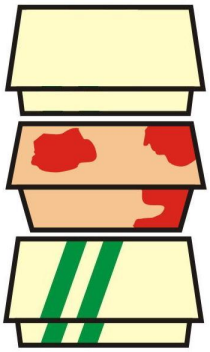
Explanation of Correct Answer:
Bob must not leave two adjacent numbers: if he does, Sam will leave those two adjacent numbers and the positive difference will be 1.
So, Bob can either delete all the odd or all the even numbers. Sam will then be forced to leave the smallest gap possible, which is 2.

Connections to Computer Science:
In this particular game, each *state* can be written down.
So, the initial state is the list (1,2,3,4,5,6,7,8).
There are 70 different states that can Bob can get to from this initial state (Bob must pick 4 values out of a set of 8 values).
For each of these 70 different states, Sam has 6 different choices (Sam must pick 2 values out of a set of 4 values).
In each of these $70 \cdot 6 = 420$ states, the positive difference could be computed. Then, both Sam and Bob would know exactly what the consequences of their choices would be.
Notice that each move/turn in the game moves from one state to another. So, we can picture the entire game as a connection of states. This connection of states is known as a *game tree*. For many games (such as chess and checkers), the number of states is massive, so we cannot store the entire game tree. Instead, we store partial information (i.e., our current state and some of the next possible states that are within several moves) and we can perform *heuristics* like the *minimax algorithm with* $\alpha - \beta$ *pruning* to make reasonable guesses about what the next best move is from the current state. A heuristic is a technique for solving a problem by finding an approximate solution when solving the problem exactly would require too much time. The minimax algorithm with $\alpha - \beta$ pruning is a technique to make the "best choice possible" in a game knowing only partial information.

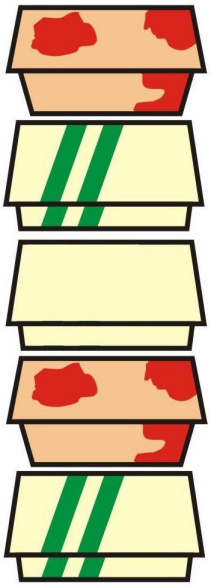
Burgers

Question:

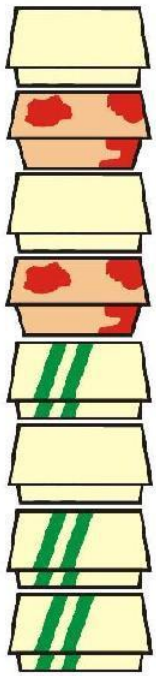
Tim and Tom are working at a burger restaurant. Tim cooks burgers one at a time. After cooking a burger, he places it into one of three different boxes: one with stripes, one with a pattern and one plain box. If he has cooked three burgers, he would have a stack as follows:



If he cooked two more burgers, he would have a stack like:



As Tim cooks a burger, he places that box on the top of the stack of not yet sold burgers, and continues to cycle through the three different boxes (stripe, pattern, plain, stripe, pattern, plain, ...) into which to place the burger.
Tom is selling the burgers one at a time and always takes the uppermost box from the stack. Tim is cooking faster than Tom can sell the burgers.
After some time, Tom has sold some burgers and Tim has cooked more burgers. Suppose the stack of unsold burgers looks like the following:



What is the fewest number of burgers sold by Tom?

Possible Answers:

4
5
6
7

Correct Answer:
4

Explanation of Correct Answer:
Consider labelling the boxes as S (for stripped), P (for patterned) and B (for blank, meaning no stripe or pattern).
So, we want to get a sequence that looks like:

S,S,B,S,P,B,P,B

where the leftmost boxes are at the bottom of the stack, and the rightmost boxes are at the top.
If no burgers were sold at all, we would have the sequence:

S,P,B,S,P,B,S,P,B,S,P,B,...

Notice that if we sell the burgers that are **bolded** as shown below:

S,**P**,**B**,S,**P**,B,S,P,B,**S**,P,B

we have the sequence

S,S,B,S,P,B,P,B.

Connections to Computer Science:
Computer scientists are concerned about how to efficiently store information. For certain problems, the best way to store information is in a data structure called a *stack*.
A stack is a data structure that imposes the following rule about accessing data:

- new items can be put on the top of the stack (to become the new top of the stack): this is called *pushing* the element onto the stack
- items that are to be removed are removed from the top of the stack (making the element just below the top the new top of the stack): this is called *popping* the stack

Stacks are used for a variety of problem solving techniques, and perhaps the easiest one to visualize is the *balanced parentheses* problem.
You would like to verify that some mathematical expression involving parentheses is valid. So (1+1), is valid, ((2+3)*(1+1)) is valid and so on. Ignoring any of the numbers or operators, we can ensure that we have a valid sequence of parentheses by the following simple process:

- read the mathematical expression from left-to-right;
- if we see (, push (onto the stack;
- if we see), pop the top (symbol from the stack;

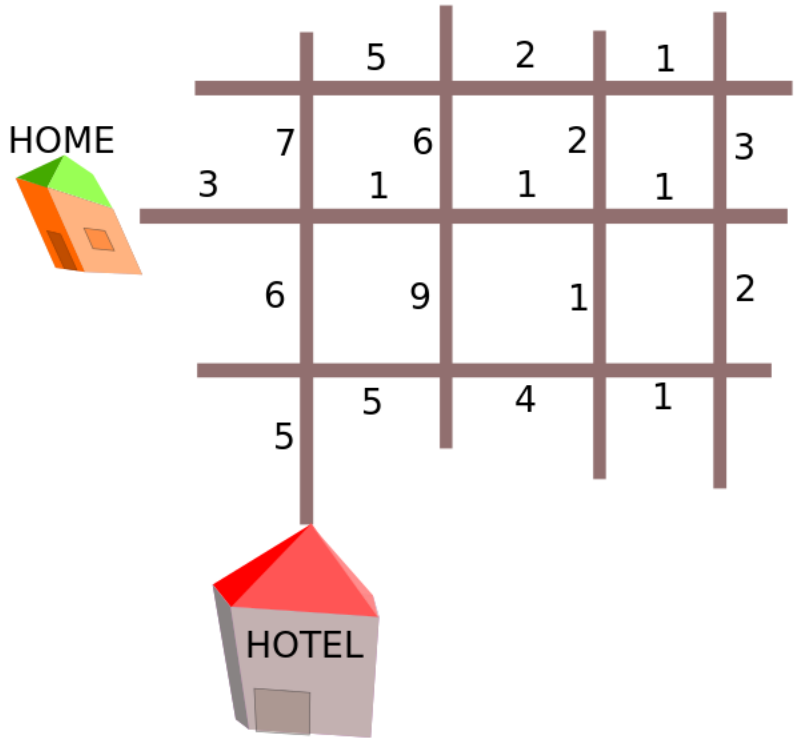
- if we try to pop an empty stack, i.e., a stack without anything on it, the sequence is invalid;
- if we read the entire mathematical expression and the stack is not empty, the sequence is invalid;
- otherwise, the sequence is valid.

You can verify that the sequences above are verified by this algorithm, and that sequences like (((1+1) and)()()(would be correctly determined to be invalid by this algorithm.

No Turning Left!

Question:

Betty the Beaver uses the roads in Beaver City to get from the Hotel to Home as fast as possible. However, her car cannot turn left!
The travel time (in minutes) between intersections in Beaver City is shown below.



How long does Betty take to get from Hotel to Home?

Possible Answers:

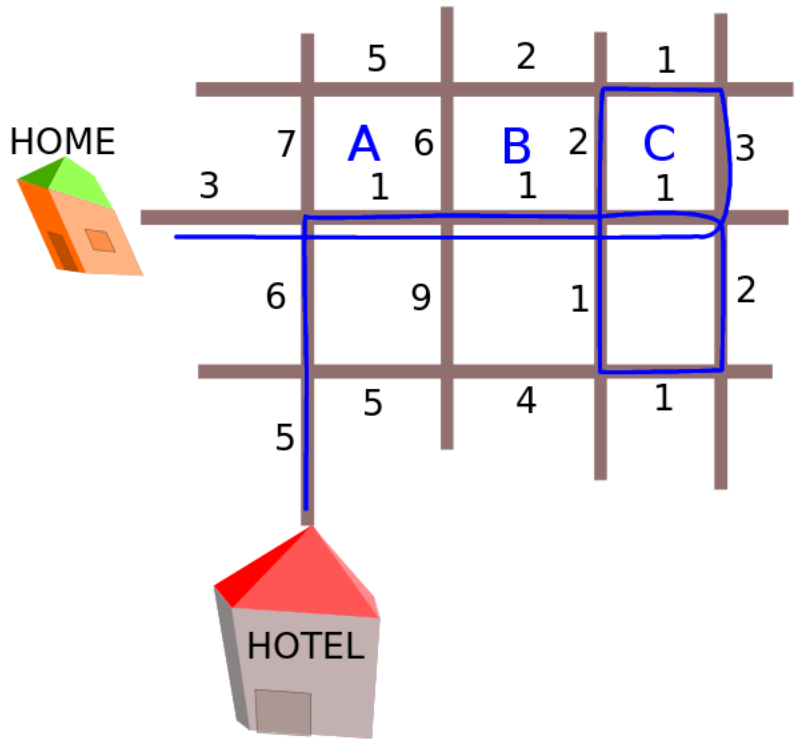
35 minutes
33 minutes
32 minutes
30 minutes

Correct Answer:
30 minutes

Explanation of Correct Answer:
First, notice that it is necessary to travel along the top-most horizontal street in order to get home. Therefore, there are 6 possibilities for our trip along the top-most horizontal street:

- go around block A only;
- go around block B only;
- go around block C only;
- go around blocks A and B;
- go around blocks B and C;
- go around blocks A, B and C.

Consider this path, shown in blue:



The total time is $5+6+\mathbf{1}+\mathbf{1}+\mathbf{1}+2+1+1+2+1+3+\mathbf{1}+\mathbf{1}+\mathbf{1}+3=30$, where the **bolded** numbers are the roads which are travelled on twice.
All other answers are longer than 30 minutes. Specifically:

- the shortest path around the block labelled A is 33 minutes;

- the shortest path around blocks labelled A,B is 32 minutes;
- the shortest path around blocks labelled A,B,C is 35 minutes.

The shortest path around the block labelled B is 42 minutes and the shortest path around the blocks labelled B and C is 45 minutes.

Connections to Computer Science:

In computer science, we often try to determine a *minimum path* based on given weights for each step in the path. Solving this type of problem is useful for all sorts of applications: transporting material, minimizing a computer circuit in order to make it run more efficiently, etc.

In this problem, we have added an additional constraint in terms of finding the minimum path. In many problems, the number of paths is very large (exponential in the number of "intersection" points) and so more sophisticated searching techniques (like *breadth-first search*) are required. In this particular problem, the constraint of no left turns reducing the number of possible paths to something that could be examined by hand.

Measuring Cups

Question:

You have a set of measuring cups. The sizes of the cups are: 8000 mL, 4000 mL, 2000 mL, 1000 mL, 500 mL, 250 mL and 125 mL. You have filled the 4000 mL, 1000 mL, 500 mL, 250 mL and 125 mL cups with flour.

Your sister also has an identical set of measuring cups of exactly the same sizes. She has filled the 125 mL, 500 mL and 2000 mL cups with flour.

Your father has a third identical set of measuring cups of exactly the same sizes. His cups are initially all empty.

All the flour from your cups and your sister's cups are put into your father's cups such that each of his cups are either full or empty.

What is the fewest number of your father's cups that are full?

Possible Answers:

1
2
3
4

Correct Answer:
2

Explanation of Correct Answer:
Notice that in total there is 8500 mL of flour. So, we can use the 8000 mL cup and the 500 mL cup.

Connections to Computer Science:
This question deals with *binary number representation*. The cups form a geometric sequence: each cup is twice the size of the next smaller one. As such, we can think of the cups 10101 ("your sister's measuring cups" are the 125 mL, not the 250 mL, the 500 mL, not the 1000 mL and the 2000 mL) and 101111 ("your measuring cups": all cups but the 2000 mL) being full.
To add two binary digits, there are 3 simple rules:

- $0 + 0 = 0$
- $1 + 0 = 0 + 1 = 1$
- $1 + 1 = 10$

The last rule can be thought of "carrying" the one to the next largest column, much in the way that adding $5 + 5 = 10$.
Using these rules, we can then add these two binary numbers using the same technique that would be used in elementary school addition:

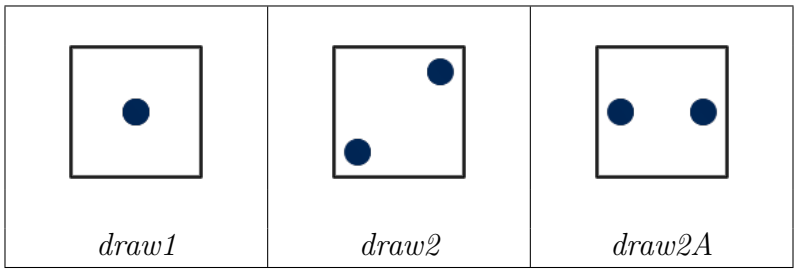
$$\begin{array}{rr} & 10101 \\ + & 101111 \\ \hline & 1000100 \end{array}$$

which means that two cups (the 8000mL and 500mL) must be full.

Dice

Question:

You have a machine that draws black circles on a white square to create the face of a die. Your machine is quite simple; instead of using 6 different patterns (one for each possible face of a 6-sided die), it only uses four commands. The result of three of these commands (*draw1*, *draw2* and *draw2A*) are shown below:



The fourth command, *turn90*, turns the face by 90 degrees clockwise. For example, we can perform the two commands *draw2A, turn90* to produce the following face:

As another example, we can perform the sequence of commands *draw1, draw2, turn90* to produce this face:

Which sequence of commands could have been used to draw this face *exactly* as shown below?

Possible Answers:

draw2A, turn90, draw2, draw1
draw2A, draw2, turn90, draw2
draw2, draw2A, turn90, draw2
draw2, turn90, draw2, draw2A

Correct Answer:
draw2, turn90, draw2, draw2A

Explanation of Correct Answer:
In the sequences of commands which are not correct, the command *draw2A* draws two dots horizontally. After drawing them the command *turn90* (which is used later in the sequence) rotates them. So each of the images drawn by these three incorrect sequences contain:

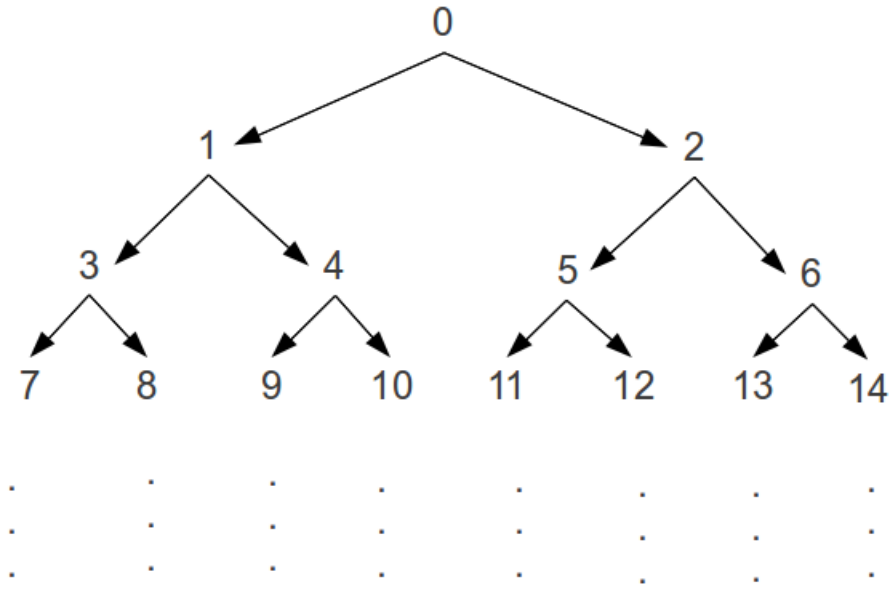
We see that these dots are not part of the questioned picture and will not be rotated to the correct orientation (since there is only one *turn90* command in each sequence).

Connections to Computer Science:
The sequence of commands in this problem can be thought of as an *algorithm*, which requires carefully keeping track of the *current state* of the picture. Here the state of the picture is the placement of the dots and the orientation of the entire face.
We can think of the commands as individual instructions in a *programming language*. Even with this limited set of instructions, we can still create a wide variety of patterns of dots on the face of the die. In computer science, even a small set of instructions in a programming language can accomplish a very wide variety of programs if the instructions are combined together in various ways.

Descend the Tree

Question:

Your friend writes down all of the integers starting from 0 to 2046 in the following way:



Specifically, below every number there are two numbers: one on the left and one on the right. For example, below 3, the number 7 is on the left, and the number 8 is on the right.
The numbers can be read in increasing order from the top row to bottom row and from left-to-right within a row.
Notice that we can get from 0 to 11 by going right (R) (from 0 to 2), left (L) (from 2 to 5) then left (L) (from 5 to 11).
Starting at 0, what sequence of left (L) and right (R) moves will end up at 100?

Possible Answers:

LRRLRR
RLLRLR
RRLLRL
LLRRLL

Correct Answer:
RLLRLR

Explanation of Correct Answer:
Notice that all even numbers are arrived at by a (R) move. Therefore, 100 must end with an (R). The parent of 100 is $\frac{\lfloor 100-1 \rfloor}{2} = 49$.
Using a similar argument, 49 (which is odd) must be arrived by an (L) move, and the parent of 49 is 24.
Similarly, 11, 5, 2, 0 is sequence of steps to the root of the tree, which causes moves to be RLLRLR.
No other path can lead to the number 100, since paths are unique.

Connections to Computer Science:
This question deals with two aspects of information.
First, numbers can be thought of as *binary numbers*, and there is a connection between binary numbers, powers of 2 and the rows in the diagram. In particular, notice that each row starts with a value of the form $2^k - 1$. Then, going left or right can be thought of as using 0 or 1 to encode some information.
The second aspect of information in this problem is that there is a formal relationship between the children and parents in this structure. This same structure is used in storing data in a *heap* data structure. Finding the parent or child of a given node can be done without *pointing* at them, but simply by determining their position relative to our current position (i.e., the parent of a value at position X is at position $\frac{\lfloor X-1 \rfloor}{2}$). It follows that binary trees of this form can be stored in a contiguous array without needing to store extra pointers, which saves memory.

Putting People in Line

Question:

You are arranging people in order based on the numbers on their shirts. The order to start is:

7 3 2 9 8 5 1 4 6

You will arrange individuals using the following technique:

- Look at two consecutive people at a time, starting from the left.
- If the person on the left has a number which is larger than that of the person on the right, switch the positions of those two people; otherwise, leave them in the order they are in.
- Move to the right one position, so that you are comparing one new person with one of the people just compared, and repeat the above comparison and potential swap.

Once you have compared the right-most two people in the list, we call this one "pass" over the list.

How many passes over the list are required until the list is in the order

1 2 3 4 5 6 7 8 9?

Possible Answers:

2
4
6
9

Correct Answer:
6

Explanation of Correct Answer:
Here is what the sequence looks like after each pass, starting from the initial sequence:

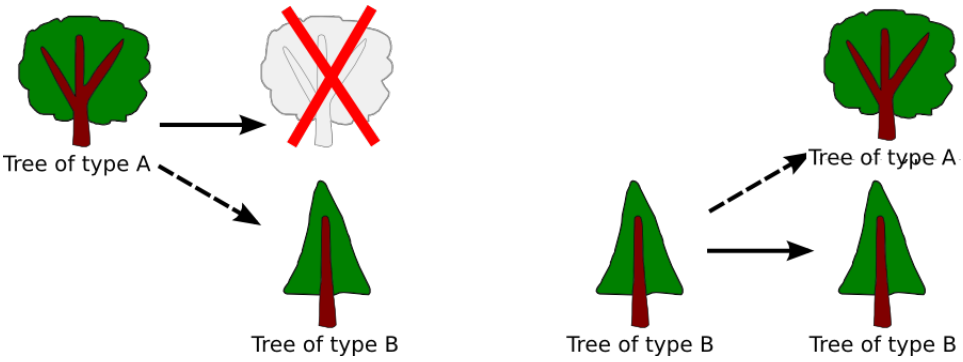
7 3 2 9 8 5 1 4 6
3 2 7 8 5 1 4 6 9
2 3 7 5 1 4 6 8 9
2 3 5 1 4 6 7 8 9
2 3 1 4 5 6 7 8 9
2 1 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9

Connections to Computer Science:
The process described above is an algorithm known as *bubble-sort*. The algorithm will sort any sequence into increasing order, but the worst-case running time is quadratic in the size of the input (i.e., if the input has size n, the algorithm will need roughly n^2 steps).
There are many other sorting algorithms, each of which highlights different aspects of algorithm analysis, data structures and algorithms. For example, *mergesort*, and *quicksort* use a *recursive* algorithm technique called *divide-and-conquer* to sort smaller lists of numbers and then combine those sorted smaller lists into larger sorted lists. The sorting technique known as *heapsort* relies on a data structure known as a *heap* to perform sorting.

Trees in the Forest

Question:

In a forest, there are two types of trees.
Type A trees live for only one year, but after this year, they transform into a tree of type B.
Type B trees live forever and produce a new tree of type A at the end of every year.
These two scenarios can be illustrated as follows, with each arrow representing the transformation at the end of one year.



For example, if we start with one type A tree, after one year there will be one type B tree in the forest. Similarly, if we start with one B tree, there will be one type A tree and one type B tree in the forest after one year.
If we start with just one type A tree in the forest, how many type A trees and type B trees will there be after 10 years?

Possible Answers:

34 A trees, 20 B trees
54 A trees, 144 B trees
34 A trees, 55 B trees
121 A trees, 55 B trees

Correct Answer:
34 A trees, 55 B trees

Explanation of Correct Answer:
There are two important observations:

- the number of A trees in a given year is the number of B trees in the previous year;
- the number of B trees in a given year is the number of B trees in the previous year plus the number of A trees in the previous year.

After a bit of analysis, we can form the following table:

	Initial	1	2	3	4	5	6	7	8	9	10
A	1	0	1	1	2	3	5	8	13	21	34
B	0	1	1	2	3	5	8	13	21	34	55

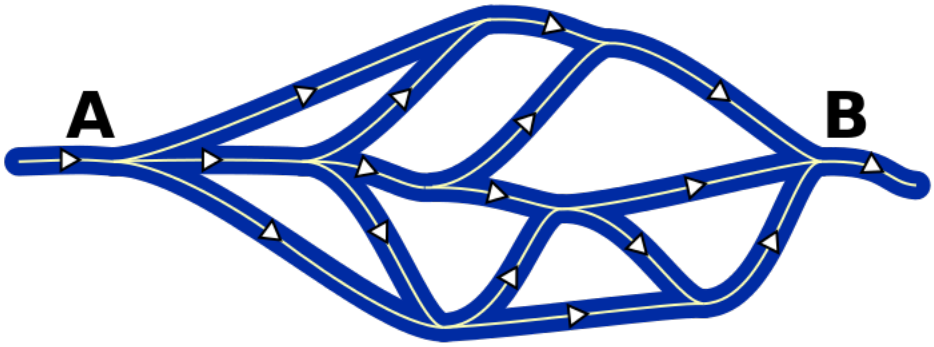
We do not actually need to build the whole table, because after several steps, it is possible to observe that the numbers form the well-known Fibonacci sequence, and there is only one answer of only Fibonacci numbers.

Connections to Computer Science:
This problem is solved by a *recursive formula*: we have a sequence of numbers (amounts of trees in this task) where each number is evaluated based on previous numbers in a sequence. The description of the problem can be thought of as a *Lindenmayer system* (also called an *L-system*). It describes the process of rewriting symbols with a small set of rules. Even from very basic rules, complex systems can emerge. The *Game of Life* is one famous 2-dimensional version of an *L-system*. L-systems can also be simplified to *context-free grammars*, which are used to verify the syntax of programming languages during compilation. L-systems are also used to create realistic looking fractals of plants, seashells or other natural phenomena.
In the solution to this problem we rely on building up larger solutions (i.e., for the state of the forest after the 10th year) from smaller subproblems. This technique of solving problems using smaller subproblems (i.e., the state of the forest after years 1, 2, 3, ...) is called *dynamic programming*.

River Inspection

Question:

Beavers want to explore the system of rivers below.
At least one beaver has to swim along each river.
Due to the heavy current, beavers can only swim downstream and they can only do one trip from A to B. So the beavers start at A, and meet at B.



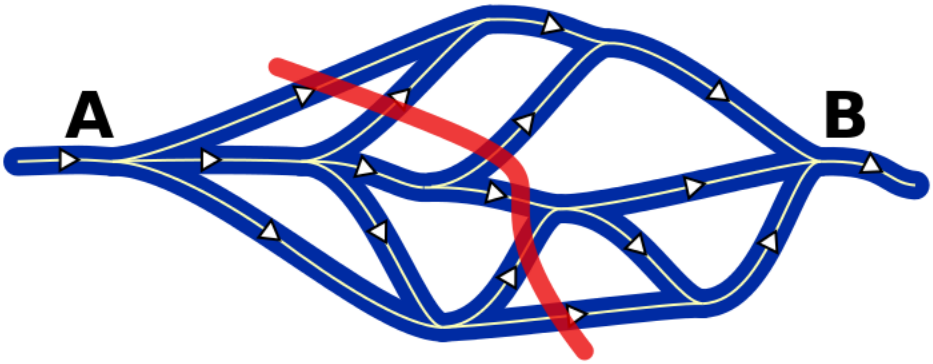
What is the minimum number of beavers needed to explore the system of rivers?

Possible Answers:

3
4
5
6

Correct Answer:
6

Explanation of Correct Answer:
As shown in the diagram below, the red line crosses six different streams. No beaver will be able to take more than one of these streams.



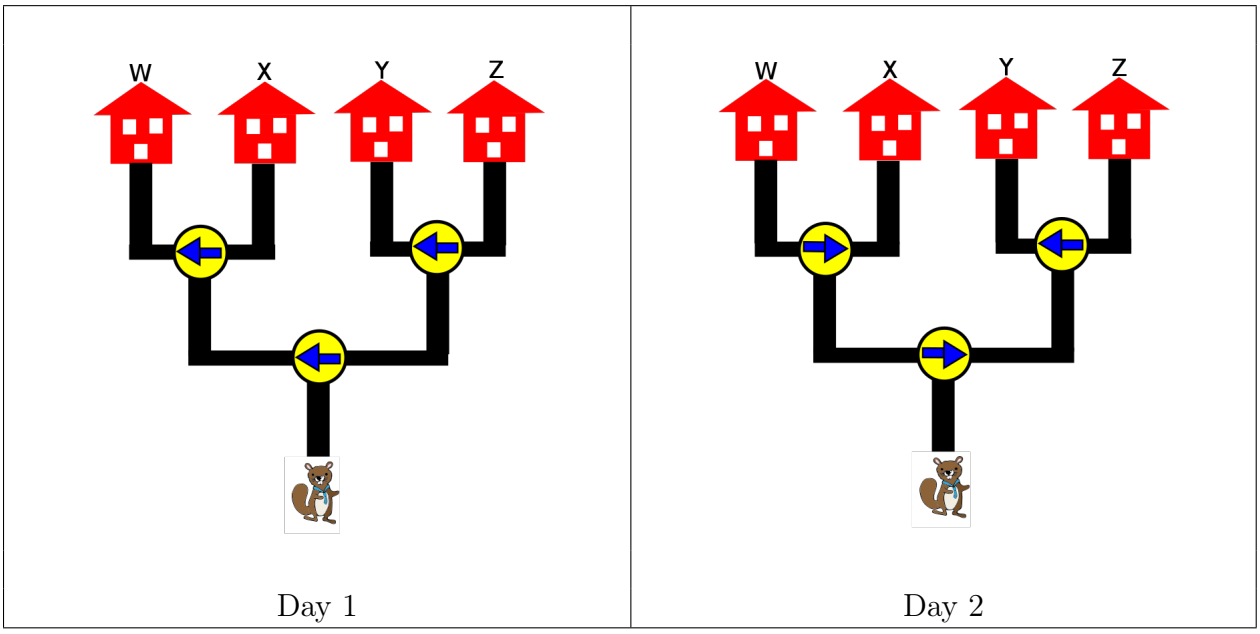
When you place one beaver on each of these streams, it is possible to cover all streams. The line is not unique, but has to cross as many lines of the graph as possible, and no other line can cut more lines.

Connections to Computer Science:
The system of rivers is an example of a *planar directed graph*. A *graph* is a collection of points (which we can think of as where the rivers split or join, which are formally called *vertices*) and a collection of connections between these points (the rivers themselves, which are formally called *edges*). A graph is *planar* and *directed* if the edges have directions (in this problem, the flow of the river) and these edges only meet at a vertex (i.e., rivers do not cross other than when they join or split).
Analyzing the flow in such a directed graph is often needed, for instance, when you want to investigate the flow in a computer program. The technique used to solve this problem (making a cut through as many streams as possible) is suited for these flow problems. Studying these kinds of algorithms is a large topic in computer science, specifically *computational graph theory*, *optimization* and *maximum-cut/minimum-flow* problems.

Visiting Friends

Questions:

Mr. Beaver has 4 friends living in different villages, and he plans to visit one of these friends every afternoon.
Mr. Beaver will follow the direction of the arrow on signs at each intersection.
Initially, all arrows point to the left road. When passing an intersection, Mr. Beaver switches the arrow to the opposite direction.
For example, on Day 1, Mr. Beaver takes the road on the left at the first intersection, takes the left road on the second intersection, and reaches Village W. On Day 2, Mr. Beaver turns right at the first intersection, then left at the second intersection, and arrives Village Y.



Day 1 | Day 2

Which village will Mr. Beaver visit on day 30?

Possible Answers:

Village W
Village X
Village Y
Village Z

Correct Answer:
Village Y

Explanation of Correct Answer:
When encountering the first intersection, Mr. Beaver takes the road on the left for odd number encounters, and the road on the right on even number encounters. Day 30 is an even number encounter at the first intersection, so Mr. Beaver will take the road on the right.
The intersection between Village Y and Village Z (which he will encounter on Day 30) will direct Mr. Beaver to Village Y on Day 2, Village Z on Day 4, Village Y on Day 6, and so on. Generally, if the Day is a multiple of 4 (i.e., has remainder 0 when divided by 4), then Mr. Beaver will visit Village Z; otherwise, if the Day is a multiple of 2 but not a multiple of 4 (i.e., has remainder 2 when divided by 4), then Mr. Beaver will visit Village Y. Since 30 has remainder 2 when divided by 4, Mr. Beaver will visit Village Y.
Another way to consider the solution: 4 days later, the state of all the signs will be the same as is was on Day 1. So Day 1 is the same as Day 5 and Day 9 and so on, and Day 30 is the same as Day 2.

Connections to Computer Science:
This task can be solved with *top-down analysis* which determines which road to take for each intersection. Top-down analysis means we need to understand the broad problem first (i.e., what does Mr. Beaver do at the first intersection) and then delving deeper (i.e., what does Mr. Beaver do at the second intersection which he encounters).
This problem can be solved by observing the *periodicity* from the *simulation*. Sometimes, an algorithm is best understood by simulating it for several steps, and patterns in how the algorithm proceeds can be observed. In this problem, the pattern of repeating the same process after 4 steps is a key observation of the simulation.