



UNIVERSITY OF  
**WATERLOO**



The CENTRE for EDUCATION in  
MATHEMATICS and COMPUTING



2017  
*Beaver  
Computing  
Challenge  
(Grade 7 & 8)*

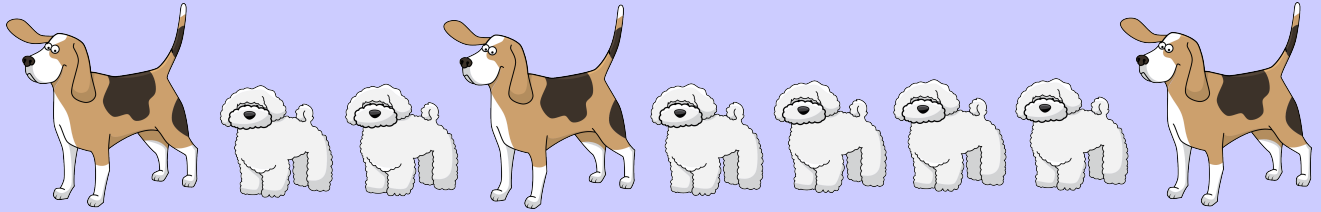
*Questions,  
Answers,  
Explanations,  
and  
Connections*

# Part A

## Swapping Dogs

### Story

Two types of dogs are standing as shown below.



A *swap* occurs when two dogs that are beside each other exchange positions. After some swaps, the three large dogs end up in three consecutive positions.

### Question

What is the fewest number of swaps that could have occurred?

- (A) 5
- (B) 6
- (C) 7
- (D) 8

### Answer

(B) 6

### Explanation of Answer

The three dogs can end up in three consecutive positions by

- first swapping the first large dog twice towards the right, and then
- swapping the last large dog four times towards the left.

Every small dog must be involved in a swap because each small dog is positioned between two large dogs. Swapping two small dogs does not have any effect so each useful swap of a small dog must be with a large dog. Since there are six small dogs, this means there must be at least six swaps.

Notice that trying to move all of the three large dogs towards the left or all three large dogs towards the right, requires more than six swaps.

### Connections to Computer Science

*Data* is stored in a computer's memory. Data includes both *internal memory* (often referred to as *RAM*) and *external memory* (which might be a hard drive or USB flash drive, for example). Computers can access data in internal memory very quickly but external memory is more cost-effective than internal memory. This trade-off means that modern computers usually have a lot more external memory but computer scientists try to find ways to use the internal memory as much as possible.

For this problem, the only operation we are concerned about is a *swap*. If we think of the dogs as data stored in the memory of a computer, then a swap involves changing the location of two pieces of data. If this data is in external memory, then having as few swaps as possible is exactly what we need to do in order to perform the overall task as quickly as possible. Swap operations are also used in some *sorting algorithms*, where we have to arrange data in ascending or descending order, since we would like some information listed in some ranking or priority order. An example of a sorting algorithm is *bubble sort*: in each round of the algorithm we swap two consecutive data that are not in the correct order.

### Country of Original Author

Canada



# School Newspaper

## Story

Ten students work on a school newspaper using a lab of identical computers. All the work is done in one day. In the table below, cells with a check mark show when each student works.

Student	Time						
	8:00	9:00	10:00	11:00	12:00	1:00	2:00
1		✓	✓				
2			✓	✓	✓	✓	
3	✓	✓					
4					✓	✓	✓
5		✓	✓				
6				✓	✓		
7			✓	✓	✓	✓	✓
8		✓					
9	✓	✓	✓				
10						✓	✓

During any of the one-hour time slots, a computer can be used by only one student.

## Question

What is the minimum possible number of computers so that every student can get their work done?

- (A) 4
- (B) 5
- (C) 6
- (D) 10

**Answer**

(B) 5

**Explanation of Answer**

At 09:00 and 10:00, 5 students need a computer – we can't solve the problem with fewer than 5 computers. If we organize the information properly, as in the following table, 5 computers are enough.

Student	Time						
	8:00	9:00	10:00	11:00	12:00	1:00	2:00
1		PC 3	PC 3				
2			PC 1	PC 1	PC 1	PC 1	
3	PC 1	PC 1					
4					PC 3	PC 3	PC 3
5		PC 4	PC 4				
6				PC 2	PC 2		
7			PC 5	PC 5	PC 5	PC 5	PC 5
8		PC 5					
9	PC 2	PC 2	PC 2				
10						PC 2	PC 2

When a student arrives, they sit on the first available computer. When they have finished their work, another student comes and sits at that computer.

## Connections to Computer Science

In order to understand large quantities of data and the relationship between different types of data, it is often best to create a method of *data representation*, for example a table, chart or diagram. Depending on the information of interest, we can represent the same data differently, and different relationships may be revealed.

For the current task, another way to represent this data is:

Computer	Time						
	8:00	9:00	10:00	11:00	12:00	1:00	2:00
PC 1	Student 3	Student 3	Student 2	Student 2	Student 2	Student 2	
PC 2	Student 9	Student 9	Student 9	Student 6	Student 6	Student 10	Student 10
PC 3		Student 1	Student 1		Student 4	Student 4	Student 4
PC 4		Student 5	Student 5				
PC 5		Student 8	Student 7	Student 7	Student 7	Student 7	Student 7

*Scheduling* and *optimizing resources* are important problems in real life. For example, in a room reservation system for a hotel, it is very important for two reservations not to overlap.

Another way to represent this data is to create a *graph*. A graph is a set of *nodes* along with a set of *edges* connecting pairs of nodes. In this task, the *nodes* are intervals and there is an *edge* between two nodes if those two intervals intersect. This graph is called an *interval graph*. If we can assign the fewest number of colours to each node so that no adjacent nodes are the same colour, then that will be the optimal allocation.

## Country of Original Author

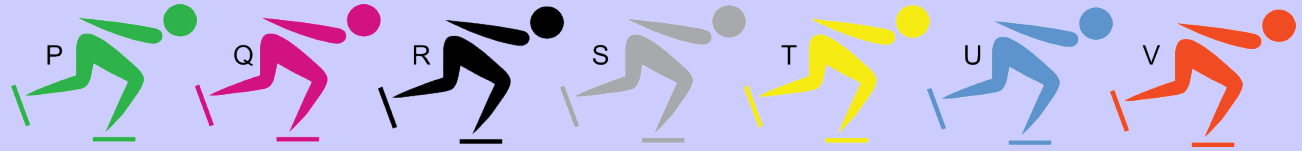
Hungary



# Skaters

## Story

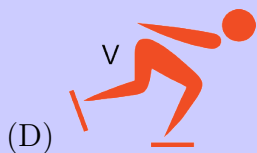
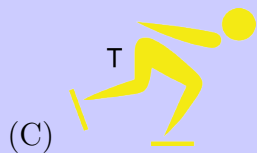
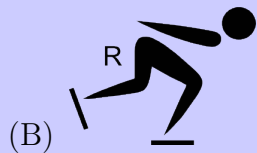
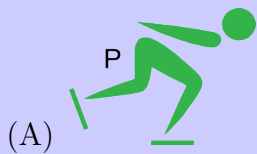
Seven people are skating in a line on a very long, frozen canal. They begin as shown below.



After every minute the person at the front of the line moves to the end of the line. For example, after one minute, U will be in front of the line, since V will move behind P.

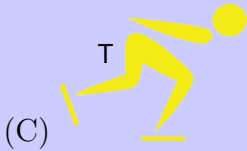
## Question

Which skater will be at the front of the line after 16 minutes?





### Answer



### Explanation of Answer

There are seven skaters. After every seven minutes, the skater originally at the front of the line returns to the front of the line. Thus, after 7 minutes, the skaters will be in the same position as they started. After 14 minutes, again, the skaters will be in the same positions. So, to get to 16 minutes, we need two more minutes, and so V then U move to the back of the line, and thus T is in front.

### Connections to Computer Science

The solution to this problem involved finding the quotient and remainder when dividing 16 by 7. Computing and working with remainders is called *modular arithmetic*. This mathematics is central to many areas of computer science. Modular arithmetic is related to how we *represent integers* in computer memory. It also plays a central role in *cryptography* which is the study of sending and receiving secret message.

A third use of modular arithmetic arises when arranging and manipulating data similar to the skaters in this problem. We can abstractly view a line of data by what computer scientists call a *queue*. One way to store the data in a queue in memory, is to use a “*circular*” *array-based queue*. This data structure allows data to be efficiently removed from the front of the queue and added to the rear of the queue. Modular arithmetic is usually used in the details of the implementation.

### Country of Original Author

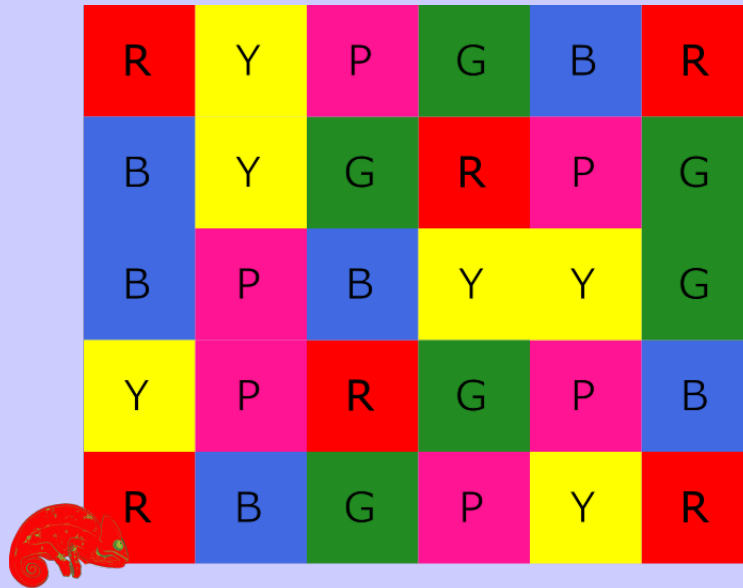
Canada



## Chameleon

### Story

A chameleon travels on the grid below. It moves between adjacent cells either horizontally, vertically or diagonally. In a cell, a chameleon has the same colour as the colour of the cell.



### Question

What is the minimum number of different colours that the chameleon has when traveling from the lower left of the grid to the upper right?

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



## Risk

### Story

Darren's computer is connected to the Internet but does not have any antivirus or firewall software. None of the accounts on his computer are protected by a password.



### Question

Which computers are at risk because of this?

- (A) only Darren's own computer
- (B) only the computers in the same room as Darren's computer
- (C) only the computers in the same country as Darren
- (D) all computers in the world which are connected to the Internet and set up like Darren's

### Answer

(D) all computers in the world which are connected to the Internet and set up like Darren's

### Explanation of Answer

Due to Darren's reckless behaviour, his own computer may get infected by malicious software. However, infected computers are often used as a platform for attacking other computers. This means that all computers connected to the Internet without proper protection, such as firewalls and antivirus software are at risk because of Darren's actions.

### Connections to Computer Science

Owners of other computers can reduce some of the risks by following good practices. For example, installing *security updates* and running *antivirus software* reduces the risk of malware spreading to other computers. Unfortunately, this does not eliminate all risks. *Malware*, which is software designed to do malicious things, if it is on Darren's computer, may be used to send spam or overload network connections (these are called *DDoS attacks*), and there is very little other computer users can do to protect themselves against these larger scale attacks.

*Cybersecurity* (or *Information Technology security*) helps protect computer systems from theft or damage to the hardware, software or the information on them, as well as from disruption or misdirection of the services they provide.

### Country of Original Author

Estonia



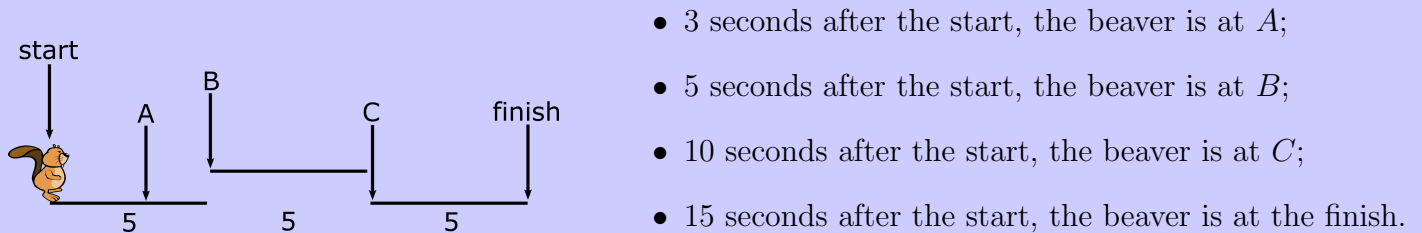
## Part B

# Jumpers

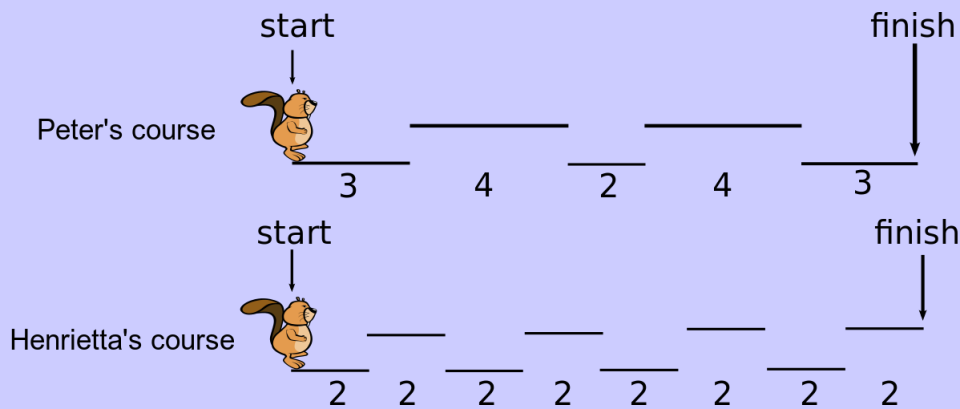
## Story

Peter and Henrietta are playing a video game. They move a beaver at a constant speed from the start of a course to the finish. The course consists of platforms on two levels. At the end of each platform before the finish, the beaver jumps instantaneously up or down to the next platform. The amount of time to move over each platform of the game is shown below each platform.

Here is an example course:



Peter and Henrietta start playing the following two different courses at exactly the same time.



## Question

For how long are both beavers moving along the top level at the same time?

- (A) 2 seconds
- (B) 4 seconds
- (C) 6 seconds
- (D) 8 seconds

### Answer

(B) 4 seconds

### Explanation of Answer

We can see this answer by writing down "being at the bottom level for one second" as a 0, and "being at the top level for one second" as 1. Then, Peter's game can be described as:

0001111001111000

and Henrietta's game can be described as:

0011001100110011

To find the times that they are both on the upper level, we need to find the times when both games have a 1 value simultaneously. This can be done by applying the boolean logic function AND between these two sequences to get:

$$\begin{array}{r} 0001111001111000 \\ 0011001100110011 \\ \hline 0001001000110000 \end{array}$$

### Connections to Computer Science

One of the main philosophies in computer science is how to represent information. In this task, we can *abstract* (or hide) away some of the details of the game to focus on what matters, which in this case is when a player is on the upper level. Then, once the *information* is represented in a precise way, we can *transform* or *combine* the information into new and meaningful ways. Specifically, for this problem, we treat the information as sequences of *binary numbers* and perform a *bit-wise* AND operation to find where both sequences have a 1 value simultaneously.

### Country of Original Author

Canada





## Bread

### Story

Alice, Bob, Charles, and Dorothy share two baguettes, two rolls, two croissants, and two slices of toast.



Each person has two different types of bread. Also:

- Alice and Bob do not have any type of bread in common.
- Charles has a baguette.
- Dorothy has a roll, but Alice does not have a roll.
- Bob has a croissant.

### Question

Which types of bread does Alice have?

- (A) A baguette and a croissant
- (B) A roll and a slice of toast
- (C) A baguette and a slice of toast
- (D) A roll and a croissant

### Answer

(C) A baguette and a slice of toast

### Explanation of Answer

The *truth table* expresses relationships between beavers and types of bread. Number the four bullet items from top to bottom, and then we can record facts from each bullet item.

	Alice	Bob	Charles	Dorothy
Baguette			Yes (1)	
Roll	No (3)			Yes (3)
Croissant	No (1),(4)	Yes (4)		
Toast				

Since Alice does not have a roll or croissant and she had two different type of bread, she must have a baguette and a slice of toast.

### Connections to Computer Science

The core idea of this problem is on *logic*, which is the study of truth, implications and conclusions. More formally, this problem relies on *boolean algebra*. Algebra deals with solving equations to find an unknown value; for example, the equation  $x + 9 = 14$  has a solution of  $x = 5$ . Boolean algebra does not give numeric values to variables, but instead gives *boolean values* of either *true* or *false* to each variable. In this problem, there would be 16 “variables”, one of each of the combinations of people and bread.

Computers use boolean logic in *conditional evaluation*, such as *if-statements*, as well as *iteration*, such as *loops*, which are two of the main programming concepts in many *programming languages*.

### Country of Original Author

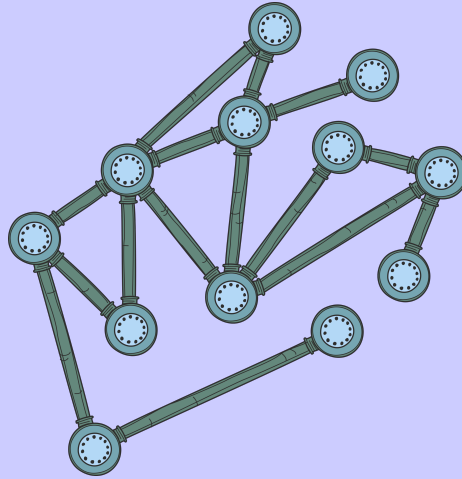
Japan



## Pipe Network

### Story

A network of 12 nodes connected by pipes is shown below. Exactly one node is clogged. However, even with this clog, water can flow between any pair of connected unclogged nodes in the network.



### Question

How many possibilities are there for the clogged node?

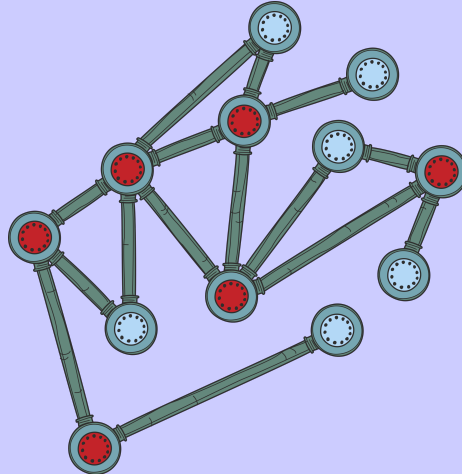
- (A) 5
- (B) 6
- (C) 7
- (D) 8

### Answer

(B) 6

### Explanation of Answer

In the diagram below, the six red nodes are the only nodes which would disconnect the pipe network if they were removed.



### Connections to Computer Science

Many structures, from *hardware chipset* to city maps, can be modeled by *graphs*. A graph is a set of *nodes* with a set of *edges* which connect pairs of nodes. In most of these structures, it is crucial to keep all components *connected* to allow communication/transportation between all nodes. One method to check the connectedness of the structure is to find and limit the number of *cut nodes* in the graph. A *cut node* is any node whose removal increases the number of connected components. This problem deals with the analysis and identification of all cut nodes in the graph.

### Country of Original Author

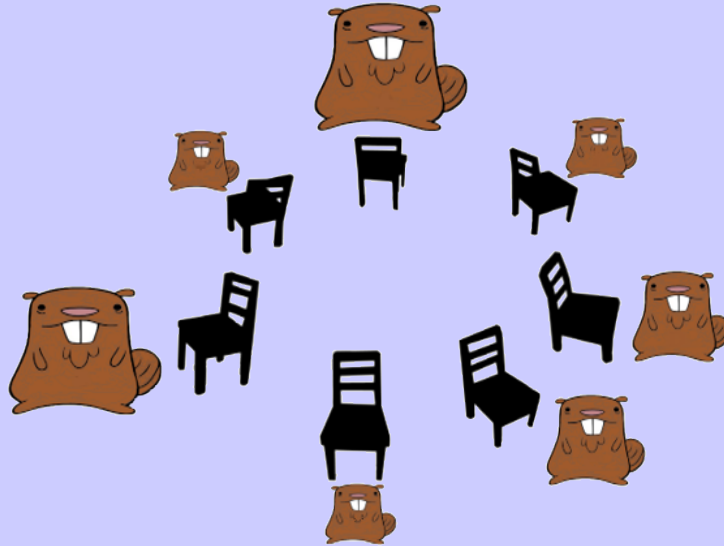
Iran



## Circle Chairs Game

### Story

Two large beavers, two medium beavers and three small beavers are playing a game around of circle of chairs. Seven chairs are placed at seven fixed positions. At the start of the game, each chair has one beaver in front of it, as shown below.



In one round of the game,

- each large beaver moves three positions counterclockwise,
- each medium-sized beaver moves two positions counterclockwise,
- and each small beaver moves one position clockwise.

### Question

After three rounds, how many chairs do not have a beaver in front of them?

- (A) 0
- (B) 1
- (C) 2
- (D) 3

### Answer

(C) 2

### Explanation of Answer

Label the chairs from 1 to 7, clockwise, starting with 1 being the top-most chair with a large beaver in front of it. Consider the movement after three rounds, which multiplies the movement by 3.

Big beavers move 9 steps counterclockwise. Thus, one big beaver moves from chair 1 to 6, and the other big beaver moves from chair 6 to 4.

Medium beavers move 6 steps counterclockwise. Thus, one medium beaver moves from chair 4 to 5, and the other medium beaver moves from chair 3 to 4.

Small beavers move 3 steps clockwise. One small beaver moves from chair 2 to 5, another moves from chair 7 to 3, and third moves from chair 5 to 1.

So after 3 rounds, chairs 1, 3, 6 are occupied with 1 beaver each, while chairs 4 and 5 are occupied with 2 beavers each. The only unoccupied chairs are 2 and 7.

### Connections to Computer Science

One way to solve this problem is remembering the *state* of each beaver, in terms of its position after each round. To determine the correctness of a program, typically we need to determine the state of all variables at each step of the program. This process of *debugging* or *tracing* of a program is an important part of designing and reasoning about the correctness of computer programs.

Computers, in particular the *central processing unit* or *CPU*, can be modelled as a *finite state machine* which keeps track of its current state and will change its state based on input. This problem can be simulated with a finite state machine with 7 states (one for each chair), and transitions for each type of beaver to its corresponding next state.

### Country of Original Author

Malaysia



# Beavers and Trees

## Story

Samantha is asked to record sequences of beavers and trees. Here is an example:

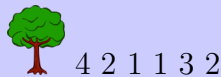


Samantha has a brilliant idea. For this example, she would only record this:







That is, she begins by recording the first image in the sequence (a beaver or a tree) and this is the only image she draws. After it, she writes down the number of times this image appears consecutively before the other image appears. Following this number, she writes down the number of times the other image appears consecutively, and so on. She continues writing down numbers in this way for the entire sequence.

Samantha looks back at her notes and finds this record of a sequence:



## Question


What was the original sequence of beavers and trees?

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

### Answer



### Explanation of Answer

We know it starts with 4 trees, since the initial picture is a tree  and it is followed by number 4, and then we alternate between beavers and trees.

### Connections to Computer Science

Compressing information saves time (in sending information across networks), space (in storing information on disk or RAM) and money (since time and space require money). Computer scientists have devised various *compression* techniques that exploit *patterns* in sequences to *encode* them using less information. In this task, the compressions technique is known as *run-length encoding*: notice that if the sequence of beavers and trees has lots of longer runs, we can write down the information more succinctly, and this encoding can be easily *decoded* to the original text. Usually, in computer science this applies to sequences of symbols like 1 and 0, instead of sequences of pictures of beavers and trees.

Many types of information, such as video files (.mp4) or audio files (.mp3), image files (.png) or even text messages (.txt), can be compressed and decompressed by various algorithms.

### Country of Original Author

Canada



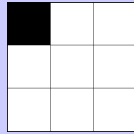


## Part C

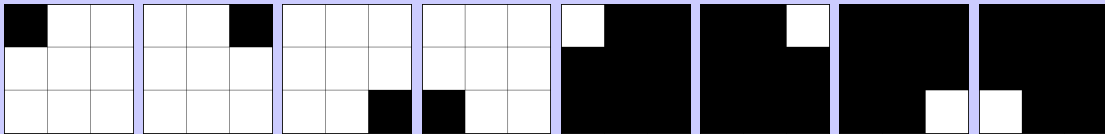
## What is THIS?

### Story

Beatrice Beaver is playing around with her simple 3-by-3 computer screen. She can paint some squares black. For example, if she painted only the top-left square, the screen would look like this:



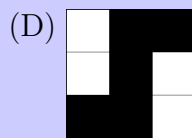
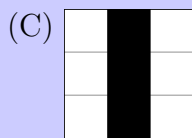
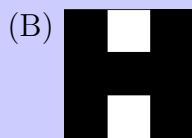
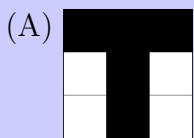
Her computer also has “rotate” and “invert” buttons. The “rotate” button rotates the screen clockwise by 90 degrees. The “invert” button changes all white squares to black and all black squares to white. For example, when Beatrice presses the “rotate” and “invert” buttons after painting only the top-left square, she can create a total of eight different patterns:



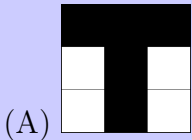
Beatrice begins with different images on the screen. She uses the two types of buttons any number of times and in any order trying to make different patterns.

### Question

Which of the following starting images allows Beatrice to make the largest number of different patterns?

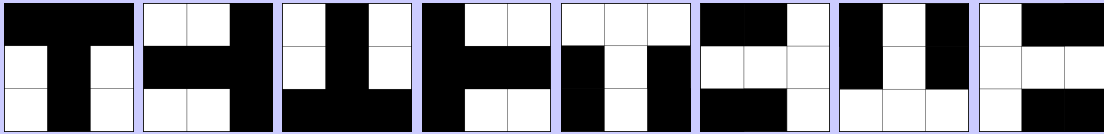


### Answer

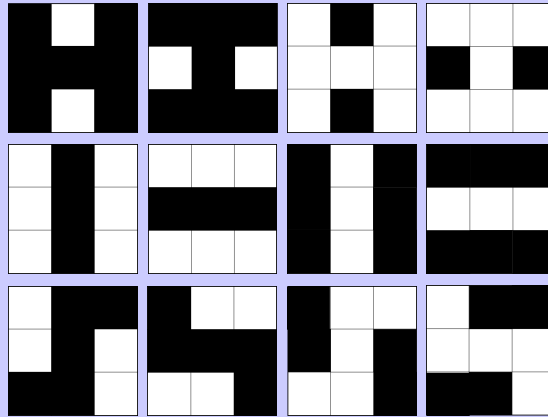


### Explanation of Answer

Notice that there are 8 different patterns for the “T” image:



Comparing this to the other three images, we see that each of those has only 4 different patterns:



### Connections to Computer Science

This problem is about *searching* in a tree-like data structure: we can think of beginning with the starting image, and applying the four rotations to this image to get (at most) four “new” patterns. From each of these patterns, we can search by applying the “invert” operation to get two possible patterns from these patterns. This searching method is one way to examine all possibilities looking in a *breadth-first* manner: we look at every pattern which is “one move” away from the current pattern before we consider “two moves”, “three moves”, etc.

One application of this searching process is that configurations of board games can be modelled this way: there is one starting configuration for most board games, and then a limited set of possible moves, which produces a set of new configurations, from which there are another set of possible moves, and so on. To strategize about the game involves considering which “winning” boards are *reachable*, and this involves searching this *game tree*. In a game, several different sequences of moves may lead to the same position. To avoid extra work, it’s useful to recognize such duplicates and analyze each different pattern only once. Many games, such as chess and Go, have been modelled and analyzed using computer simulations and represented using game trees.

Country of Original Author

Canada

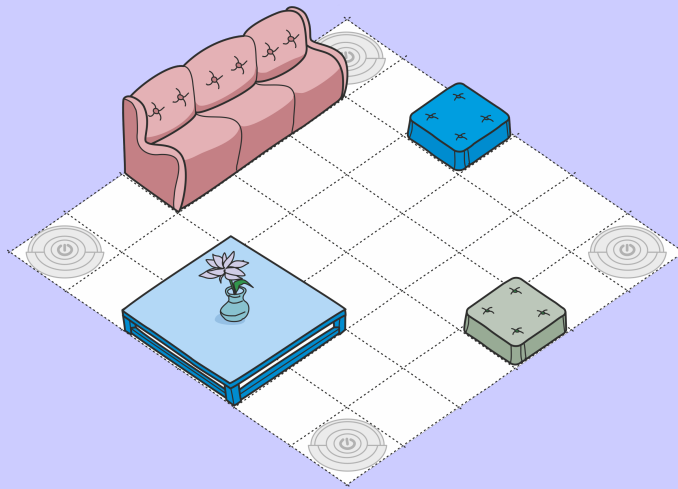


## Robot Cleaner

### Story

A robot washes the square tiled floor shown below by using the following commands:

- F – move forward one tile (which takes 1 minute)
- W – wash a tile (which takes 1 minute)
- R – turn  $90^\circ$  right (which is performed instantly)
- L – turn  $90^\circ$  left (which is performed instantly)



The robot can start at any corner facing any direction and can end at any corner. It never goes on a tile occupied by one of the four pieces of furniture and washes all the other tiles, including the 4 corner tiles, exactly once. The robot may travel over a tile more than once.

### Question

What is the minimum possible number of minutes the robot needs to wash the entire floor?

- (A) 53
- (B) 54
- (C) 55
- (D) 56

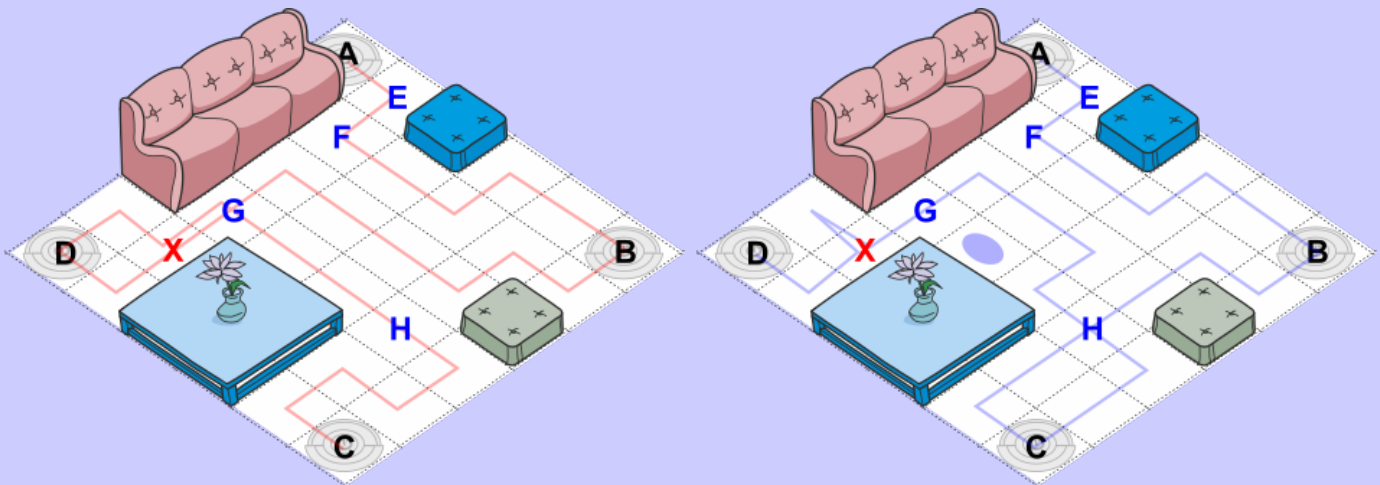
**Answer**

(C) 55

**Explanation of Answer**

There are  $36 - 9 = 27$  floor tiles, so washing takes 27 minutes. If every tile was visited once, moving would take 26 minutes (as we don't have to move to the first tile). So we have to minimize number of tiles which are passed twice.

Look at the pictures below. There is a tile X which must be passed twice every chosen way. Some of the tiles lie a "narrow passage" to corner tiles A B C D: these tiles, labelled by the letters E, F, G, and H, must be passed over twice if the nearest corner to E, F, G, or H, is not a starting or ending tile. Tiles E and F are on the A narrow passage, tile G is on the D narrow passage, and H is on the C narrow passage. Corner B has no narrow passage tile. If robot starts or finishes at some corner, the narrow passage tile(s) on this corner can be passed through only once. Thus we can try to arrange the path so that robot went through as few narrow passage tiles as possible.



If the robot starts and finishes at the same corner tile it must pass all of narrow passage tiles twice. Because corner A has two narrow passage tiles, E and F, a good strategy is to choose the corner A as a starting (or finishing) point. We then have to decide between finishing corners C and D (in both cases we spare another one narrow passage tiles of G or H).

Tracing the picture on the left shows that to move from A to C needs 28 moves forward (it contains points X and G). Tracing from A to D needs odd number of moves so after using 28 moves forward at least one tile is not passed yet (in the right picture, one such tile which is not passed through yet is the one with the blue ellipse) so it is longer than from A to C.

Minimal time for cleaning is  $27$  (washing) +  $28$  (moving) minutes = 55 minutes.

### Connections to Computer Science

This task is a slight modification of the *travelling salesperson problem*, which is the problem of finding the shortest route around a collection of cities, with the restriction that we visit every city at least once. In our task, we may visit a “city” (i.e., a tile) more than once, but the underlying problem is equivalent. More formally, this shortest route is called a *Hamiltonian path*. Computer scientists have not yet found an efficient algorithm to solve the Hamiltonian path problem, and most computer scientists believe that there is no efficient solution. That is, any possible algorithm to solve Hamiltonian path with require an *exponential* amount of time in terms of the size of the problem, and therefore, even for “small” problems of 100 cities to be visited (or 100 tiles to be cleaned), even the most powerful computers would take thousands of years to find a solution.

The study of whether or not there exist efficient algorithms for these types of problems deals with the fundamental problem of determining whether  $P = NP$  or  $P \neq NP$ . This problem of *P versus NP* has remained unanswered for decades and is one of the most well-known problems in computer science.

### Country of Original Author

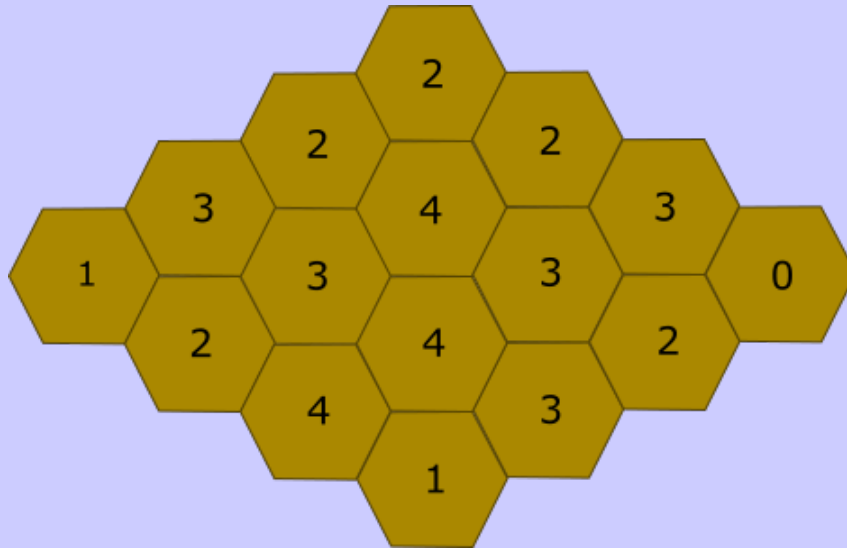
Lithuania



## Beehive

### Story

A bear studies how many hexagons in a honeycomb contain honey. For each hexagon, the bear records how many *other* hexagons touching this hexagon contain honey. So this number could be 0, 1, 2, 3, 4, 5 or 6. The results of the bear's study are below.



### Question

How many hexagons contain honey?

- (A) 7
- (B) 8
- (C) 9
- (D) 10



Answer

(C) 9

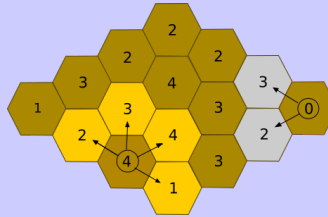
Explanation of Answer

One way to solve this task is to start from the cells for which the content of their adjacent cells is known. It is the case for two cells, that have a circle in the picture below:

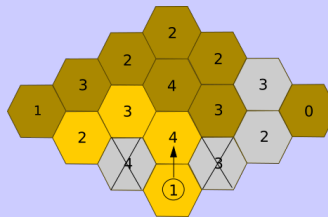
All the adjacent cells of the cells with a 0 are empty

All the adjacent cells of the cells with a 4 are full

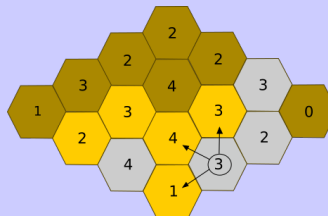
We will use yellow to represent honey in a hex, grey to represent no honey, and brown for undetermined.



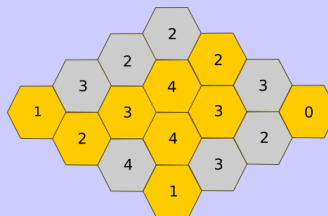
After that, we can easily find some others cells, near the zone that we have changed, for which the content of their adjacent cells is now known. For instance the cell with a circle in the picture below, has only 1 full adjacent cell. It is the one already colored in yellow, just above it. So its other adjacent cell is empty.



Then, the cell with a 3 on it, which we just coloured grey, must have exactly 3 full adjacent cells. We have already two of them on its left, the one on the right is empty, so the one above it must be full.



We can go on from one adjacent cell to the others, in order to cover all the beehive.



### Connections to Computer Science

This task concerns *logic* and *inference*.

The logic in this problem is to understand that, for example, a cell marked as 0 means that none of its neighbours contain honey. From this fact, we can *infer* further facts, and build up a solution to the larger problem.

The method of building a solution using inference is what is called a *bottom-up* algorithm: we start with a solution to a very small part of problem (the “bottom” of the problem) and then build up larger and larger solutions until we have solved the entire problem (the “top” of the problem).

### Country of Original Author

Iran



## Building a Dam

### Story

To build a dam, a beaver needs to cut 10 metre logs into smaller logs of lengths three and four. The table below shows how many of these smaller logs are needed.

<b>Length of Log</b>	<b>Number of Logs</b>
4 metre	7
3 metre	7

The beaver cannot combine together smaller logs to form larger logs.

### Question

What is the least possible number of 10 metre logs that the beaver must cut?

- (A) 4
- (B) 5
- (C) 6
- (D) 7

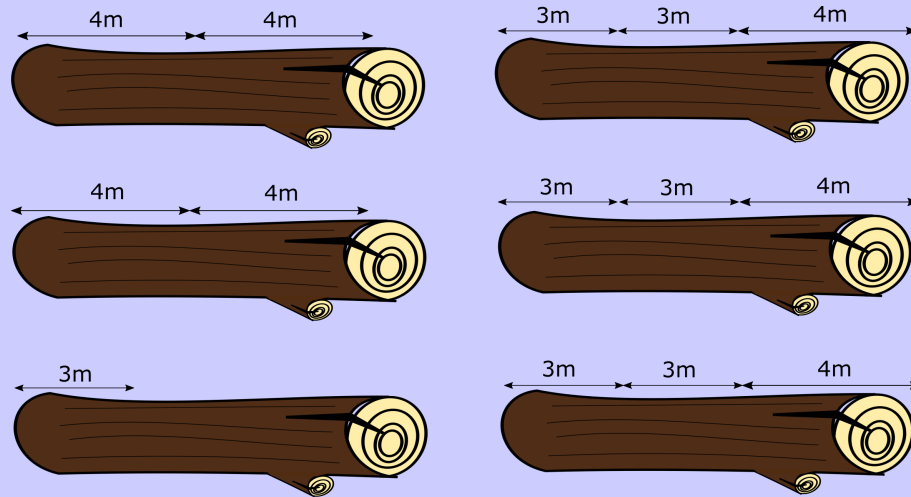
## Answer

(C) 6

## Explanation of Answer

Notice that to create the desired log lengths we need to create a total length of 49 logs ( $4 \cdot 7 + 3 \cdot 7 = 49$ ), and thus we need at least 5 logs. However, if we used only 5 10 metre logs, only one of those logs could “waste” 1 metre. Thus, none of the logs can be cut into two 4 metre logs (since that would waste 2 metres of log), which means we can create at most 5 logs of length 4 metres if we only have 5 logs of length 10 metres. Therefore, there must be at least 6 logs used.

The illustration below shows how 6 logs of length 10 metres can be cut to produce the required log lengths:



## Connections to Computer Science

This problem is an instance of the *stock cutting problem*. The practical application of this problem is to cut a standard length (or size) of material, such as log, into specified lengths (or sizes) while minimizing the total material used. This problem is an *optimization problem*, since we would like to maximize or minimize some *cost function* (for example, the waste) in order to meet the specification (for example, the required lengths of material that must be produced).

The problem can be formulated as an *integer linear programming problem*, but there is no known efficient algorithm for solving these types of problems. However, computer scientists have developed *heuristic algorithms*, such as *hill climbing* and *simulated annealing*, which attempt to find approximate, possibly non-optimal, solutions to these integer linear programming problems.

## Country of Original Author

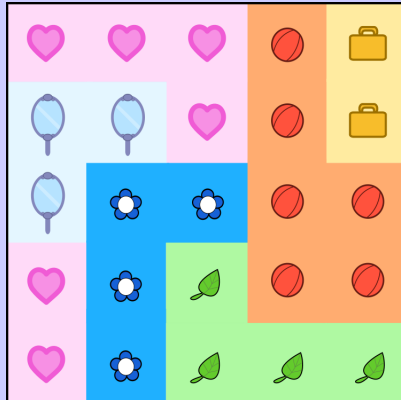
Iran



# Wallpaper

























## Story

Robyn covers a wall with six overlapping rectangular sheets of wallpaper as shown. Each sheet of wallpaper is designed using a different image in a repeating pattern.



## Question

What is the order of the wallpaper pieces from the one placed first to the one placed last?

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

### Answer

(A) 

### Explanation of Answer

The yellow wallpaper with the briefcases is the only wallpaper that isn't cut off by another one, so that must be the last one. You can see the suitcase cuts off the basketball wallpaper, that the basketball wallpaper cuts off the leaf wallpaper, the leaf wallpaper cuts off the flower wallpaper, the flower wallpaper cuts off the mirrors and the mirrors cut off the hearts.

### Connections to Computer Science

Using the same wallpaper rectangles in a different order would have given a totally different image: the *ordering of operations* are very important in computer science.

The *order* in which you execute statements is crucially important in programming or using a computer: an incorrect order of statements will yield incorrect results. Thinking of a mathematical expression like  $3 + 2 \times 7$ , we “know” from our BEDMAS rules that we must evaluate the  $2 \times 7$  first before adding it to 3. Similarly in *computer algorithms*, the order of evaluating steps influences what the end result will be.

This task is also an example of using *layers*: digital image editing uses layering to alter parts of an image. For example, images can be *composed* so that foreground images, such as a person, can be placed in a background image, such as a beach or forest. The software to enable this requires algorithms such as *boundary detection* in order to know where images overlap or intersect.

### Country of Original Author

United States

