



UNIVERSITY OF
WATERLOO



The CENTRE for EDUCATION in
MATHEMATICS and COMPUTING



2021
*Beaver
Computing
Challenge*
(Grade 9 & 10)

*Questions,
Answers,
Explanations,
and
Connections*

Part A

Messages

Story

Beaver Xavier creates a code by representing letters using only the digits 1 and 0 as shown.

Letter	T	E	A	K	C	R
Code	1	00	0010	0110	1010	1110



Xavier uses the letters to write a message for Yvonne. Then he replaces each letter with the corresponding code and sends her the result.

Question

Which of the following messages does Xavier write if he sends Yvonne 1001001100010100010111000 ?

- (A) TEACRATE
- (B) EATCAKE
- (C) TAKECARE
- (D) RETAKE

Answer

(C) TAKECARE

Explanation of Answer

By replacing each letter in TAKECARE with its corresponding code, we get 1001001100010100010111000 which is the message Xavier sent.

For Option A, Xavier would have sent 1000010101011100010100 which does not match.

For Option B, Xavier would have sent 0000101110100010011000 which does not match.

For Option D, Xavier would have sent 11100010010011000 which does not match.

You might have tried to solve this without considering the provided options and instead tried to decode 1001001100010100010111000 directly. This is a bit tricky to do when working from left to right. For example, what Xavier sends begins with 100, which matches messages that begin with TE, but also messages that begin with TA. The fundamental issue is that the code for E is the beginning of the code for A. Also note that the code for T is the start of the codes for C and R. In general, this makes decoding from left to right relatively hard for this particular code.

Interestingly, it is easier to decode by working from right to left. We don't run into the same fundamental issue because there isn't a code that ends with the code for another letter. For example, what Xavier sends ends in 00 and none of the codes for other letters end in 00 so the message Xavier wrote must end with E. Continuing to decode this way, we end up with the message TAKECARE.

Connections to Computer Science

When information is transmitted between computers, either through wires or through the air wirelessly, the information is sent as sequences of *signals*. Each signal can be viewed as describing one of two possibilities: either *on* or *off*. In this problem, each letter corresponds to a particular *binary sequence*, where instead of sending on/off signals, the binary digits *0* and *1* are used.

The specific way by which information is translated into signals is called a *binary code*. In this problem, a *variable length code* was used: the code length of a letter could be 1, 2, or 4 *bits*. Most computers use *fixed length codes* to encode information: for example, the most common encoding of textual information is (or is based on) *ASCII*, where there are 8 signals for each different character.

It is crucial that the receiver of a coded message can translate the signals back to the original message without making mistakes. Mistakes could include not having a corresponding original message or having two or more possible original messages. In other words, codes must be designed carefully, and ensure that they are *uniquely decodable codes*.

A special kind of uniquely decodable code is the *prefix-free code*. These code have the property that no code is the prefix (or starting sequence) of any other code. The code that Xavier creates is **not** prefix-free, since, for example, the code of 00 for character E is a prefix of the code 0010 for character A.

For example, the following is a prefix-free code for the same characters:

Letter	T	E	A	K	C	R
Code	11	00	0111	0110	1010	1001

Prefix-free codes tend to be quite short and they are easily decoded. They are not only used for communication purposes but also are used in several *compression algorithms*.

Country of Original Author

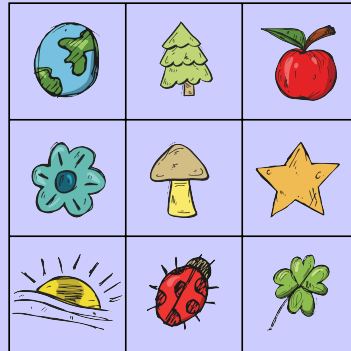
Switzerland









Arranging Objects

Story

A board is divided into squares and a different object is placed in each square as shown.

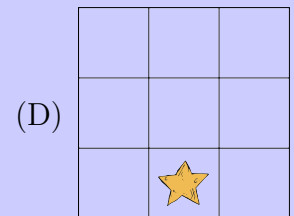
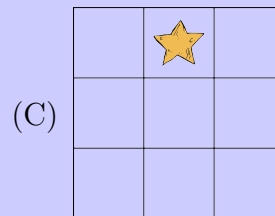
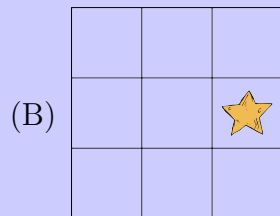
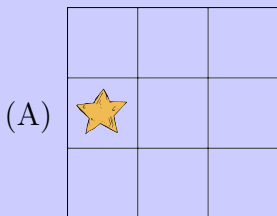


A *swap* exchanges the locations of two objects. Three swaps occur in this order:

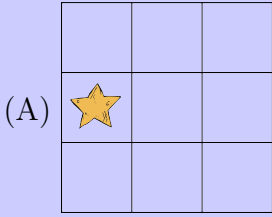
-  ↔ 
-  ↔ 
-  ↔ 

Question

What is the location of  after the last swap?

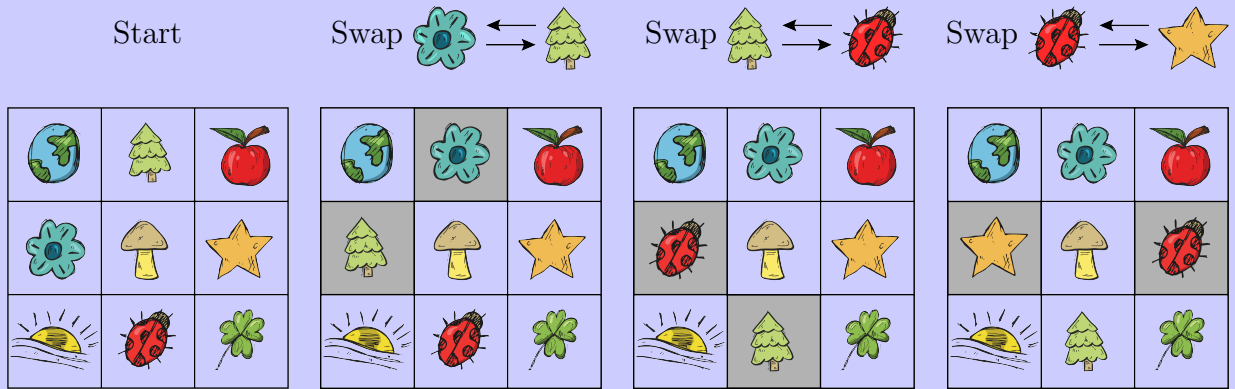


Answer



Explanation of Answer

Here is the state of the board after each swap:



If you compare the locations of the objects at the beginning, with the locations of the objects after the last swap, you can see that the star is now in the original location of the flower.

Another way to see this final result is to think about trading goods at a market. Suppose you bring a flower and you trade it for a tree. Then you trade your tree for a ladybug. Then you trade your ladybug for a star. The item you end up with is the same as if you traded your flower for a star directly.

Connections to Computer Science

This task focuses on the concept of *swapping* values between two *variables*. In computer programming, a *variable* is a memory location that can hold information. *Swapping* involves exchanging the values of two variables.

For example, suppose A is a variable that holds the value “18” and B is another variable that holds the value “42”. After a swap, variable A will hold “42” and variable and B will hold “18”.

In most *programming languages*, a *temporary variable* is needed to swap values between two variables. For example, if there was a temporary variable T, the following three steps would swap the values between A and B:

1. Give T the value of A.
2. Give A the value of B.
3. Give B the value of T.

One of the most common uses of swapping in computer science is in *sorting*, where a collection of data it put into either ascending or descending order.

Country of Original Author

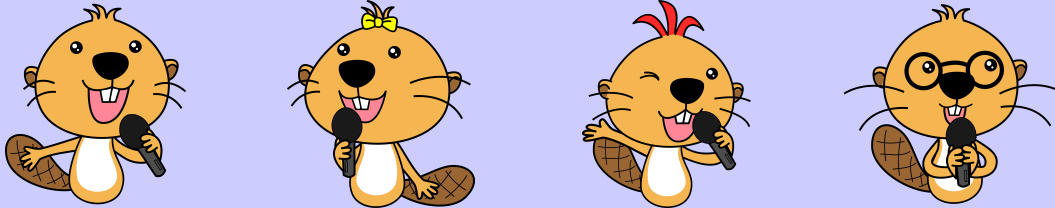
India



Singing Contest

Story

Three judges evaluate four singers in a singing contest.



Each judge uses their own scoring system, resulting in the following scores.

Singer	Judge 1	Judge 2	Judge 3
Ara	9	85	20
Benito	7	100	15
Chien	8	70	25
Dennis	10	60	45

The different scoring systems make it difficult to declare a winner, so the judges each rank the singers 1st, 2nd, 3rd, and 4th from highest to lowest score.

For example, Judge 1 ranks Benito 4th since Benito received the lowest score by Judge 1. Judge 2 ranks Benito 1st since Benito received the highest score by Judge 2.

Each singer's ranks are then added together to get their *rank sum*. The singer with the *lowest* rank sum is declared the winner.

Question

Who won the singing contest?

- (A) Ara
- (B) Benito
- (C) Chien
- (D) Dennis

Answer

D) Dennis

Explanation of Answer

In the table below, the judges' ranks are given in parentheses and used to calculate the rank sum for each singer.

Singer	Judge 1	Judge 2	Judge 3	Rank Sum
Ara	9 (2)	85 (2)	20 (3)	7
Benito	7 (4)	100 (1)	15 (4)	9
Chien	8 (3)	70 (3)	25 (2)	8
Dennis	10 (1)	60 (4)	45 (1)	6

Since Dennis has the lowest rank sum, he is the winner.

Connections to Computer Science

The field of *machine learning*, which is an area of *data science*, is concerned with using large amount of data with various characteristics to identify, categorize, and make decisions about that data. The key idea is that a *model* is constructed, or *trained*, based on a set of examples, and this model is refined with more data.

For example, suppose a machine learning model is trained using data with characteristics of a house: total floor area, number of rooms, distance from the closest school, age, distance from nearest grocery store, etc. The goal of the model is to predict the price of a house. The units of each characteristic such as area, age, and distance are different, and the range of price will be quite different.

In this case, *normalization*, a process of converting values so that all features have a similar influence, is required. Therefore, in machine learning, the normalization process is important to ensure that all data injected into a model are reflected at the same scale (level of importance).

In this task, each judge can be viewed as a different characteristic being measured. Ranking each judges score, rather than using the actual score, is the normalization process.

Country of Original Author

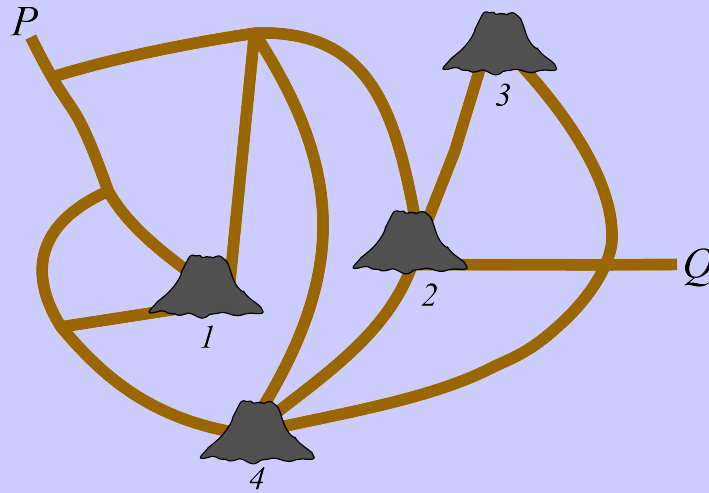
South Korea



Volcanoes

Story

In the map shown, Dino can follow roads and can climb up and over volcanoes unless they are erupting.



Because two volcanoes are erupting, Dino cannot get from point P to point Q .

Question

Which two volcanoes are erupting?

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

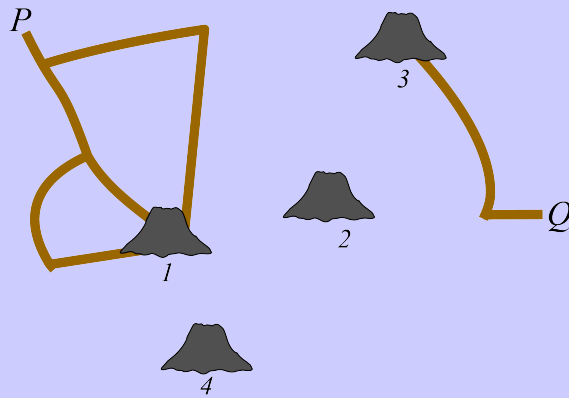
Answer

(D) Volcanoes 2 and 4

Explanation of Answer

Since Dino cannot climb up and over an erupting volcano, a road to or a road from an erupting volcano is not helpful (at least up to the point where it intersects another road).

In the following image all roads to and from volcanoes 2 and 4 have been removed. Notice that in this situation it is not possible for Dino to get to point Q from point P .



Using the same approach we can see that a route from P to Q does exist for the other three options.

<p>Volcanoes 1 and 2</p> <p>Route over volcano 4 exists</p>	<p>Volcanoes 3 and 4</p> <p>Route over volcano 2 exists</p>	<p>Volcanoes 1 and 4</p> <p>Route over volcano 2 exists</p>
--	--	--

Connections to Computer Science

In computer science, this type of diagram is called a *graph*. The volcanoes are the *vertices* and the roads connecting the volcanoes are the *edges* of the graph.

In the initial graph, points P and Q are *connected*: there is a sequence of edges, known as a *path*, that leads between P and Q.

This task is concerned with finding a set of *cut vertices*, which are also known as *articulation points*. A set of cut vertices has the property that when those vertices and all of the edges attached to those vertices are removed, the graph will become disconnected.

In this problem, no single vertex is a cut vertex: removing just one of the four vertices will not disconnect the graph. However, removing Volcanoes 2 and 4 will disconnect the graph.

One crucial use of finding cut vertices is for determining the *fault tolerance of a network system*: finding the number of routers or servers that would need to go down/offline before some computers in the network can no longer connect to each other.

Country of Original Author

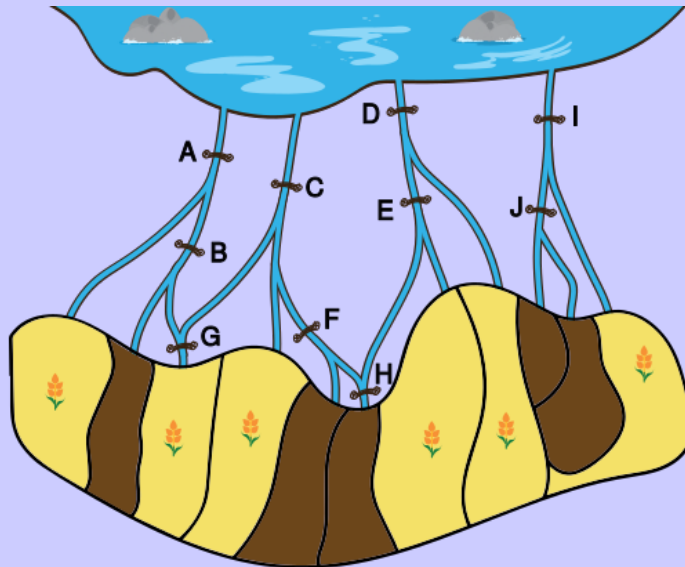
Romania



Wheat Field Irrigation

Story

Water flows from a lake along irrigation channels towards some fields. The water only flows downwards in the diagram. Valves at the spots marked A to J can each be open or closed. When a valve is closed, water stops flowing further towards the fields through that channel. Water can flow past an open valve.



A farmer configures the valves so that water is supplied to the six yellow wheat fields marked with 🌾, but no water is wasted on the other five fields of weeds shown in brown.

Question

How many valves are closed?

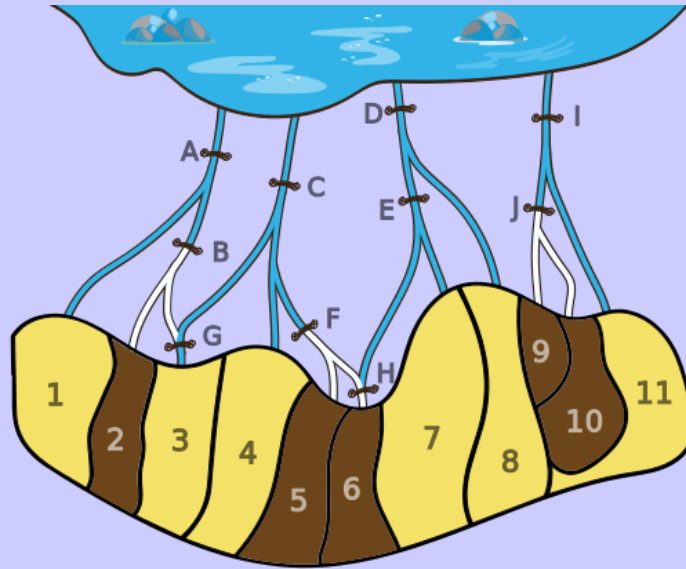
- (A) 2
- (B) 3
- (C) 4
- (D) 5

Answer

(C) 4

Explanation of Answer

The four valves at B, F, H, and J must be closed.



If the valves at these points are not closed, the water will reach at least one field with weeds. If more valves are closed, then water will not reach all the fields with wheat. We can determine this by examining the role of each valve in the system:

- A must be open to supply water to field 1.
- B must be closed to avoid supplying water to field 2, since the valve at A is open.
- C must be open to supply water to field 4.
- D must be open to supply water to field 8.
- E must be open to supply water to field 7.
- F must be closed to prevent supplying water to field 5, since the valve at C is open.
- G must be open to supply water to field 3.
- H must be closed to prevent supplying water to field 6. Although F is closed, water will still flow through valves at D and E, so it must be stopped at H.
- I must be open to supply water to field 11.
- J must be closed to prevent supplying water to fields 9 and 10, since the valve at I is open.

Connections to Computer Science

In this task, water flows to the fields based on a number of *conditions*. For instance, water flows to field 7 if both valves D and E are open, and water flows to field 3 if G is open and either of these two conditions hold:

1. C is open, or
2. both A and B are open.

These types of compound conditions are formed with the *boolean operators*, specifically *AND* and *OR*. If there are two valves on the same channel one after other, such as C and G, that would be represented as an AND expression: in order for water to flow, both C AND G must be open. If water can flow to the same field from two separate channel segments, such as field 3, that would be represented as an OR expression: in order for water to reach field 3, either it goes along the path with values A, B, and G, or it goes along the path with values C and G.

Whether a valve is open or closed can be represented by a *boolean value*, which is either *true*, representing open, and *false*, representing closed.

In programming languages, boolean conditions are used in *conditional expressions* such as *if-statements*, which are statements that ensure a certain condition is true before executing a series of instructions.

Country of Original Author



Turkey

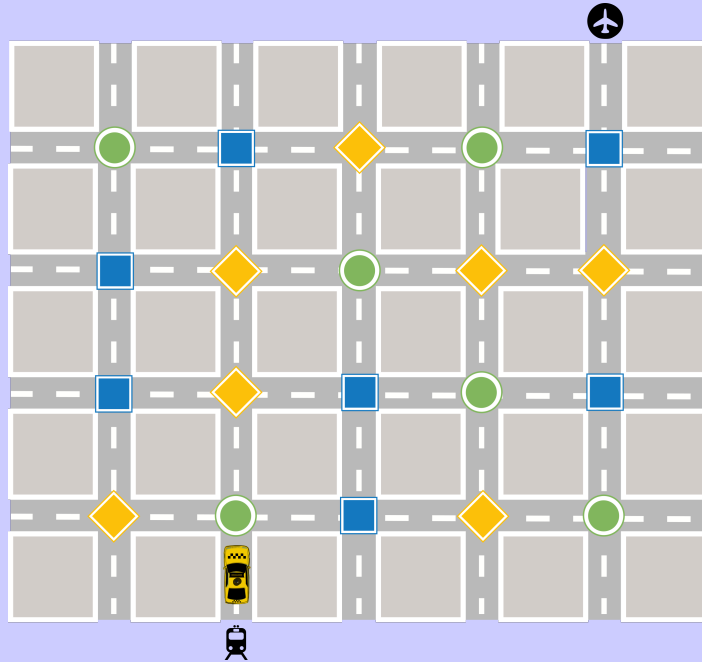


Part B

Taxi













Story

A taxi travels from the train station  to the airport  along the city streets shown in the map below. At each intersection, the taxi travels one block in the direction indicated by the symbol at that intersection.



Question

Which of the following gives the correct instruction for each symbol?

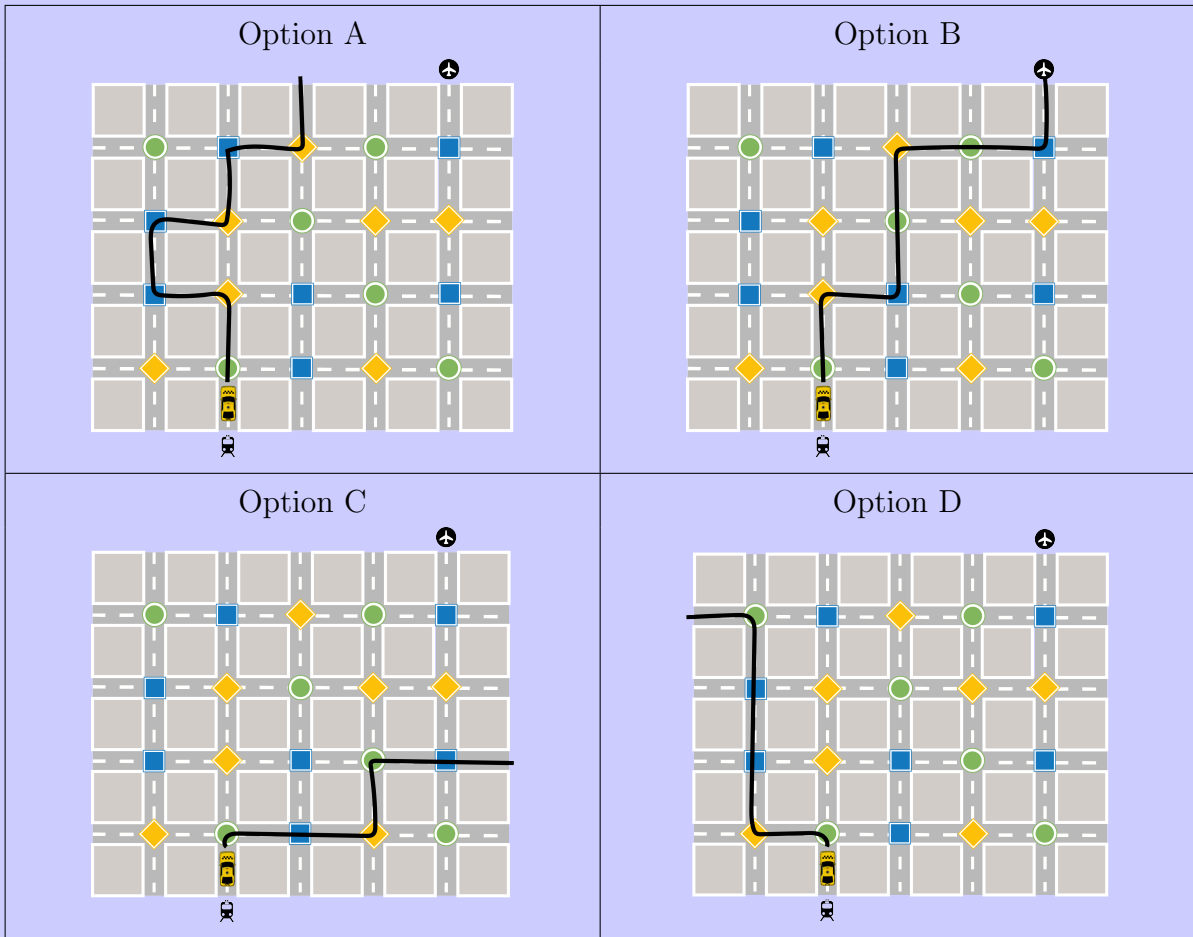
- | | | | |
|-----|---|-----|---|
| (A) |  continue in the same direction | (B) |  continue in the same direction |
| |  turn the taxi to the driver's left | |  turn the taxi to the driver's right |
| |  turn the taxi to the driver's right | |  turn the taxi to the driver's left |
| (C) |  turn the taxi to the driver's right | (D) |  turn the taxi to the driver's left |
| |  turn the taxi to the driver's left | |  turn the taxi to the driver's right |
| |  continue in the same direction | |  continue in the same direction |

Answer

- continue in the same direction
- (B) ◆ turn the taxi to the driver's right
- turn the taxi to the driver's left

Explanation of Answer

The taxi's route corresponding to each of the four options is shown below. Option B gets the taxi from the train station to the airport.



Connections to Computer Science

This task is primarily concerned with *tracing an algorithm*. Specifically, each configuration of symbols corresponds to a different *sequence of instructions*. The key idea is to *trace* the algorithm: at each intersection, determine whether the taxi turns right, left, or continues going straight. In order to assist with the tracing of the algorithm, we need to keep track of the direction of the taxi as it comes into the intersection: this can be viewed as the *state* that is changing in this algorithm.

Computer programmers use tracing to find errors in their programs, which is called *debugging*, or to *verify* that their code is correct.



Country of Original Author

Austria



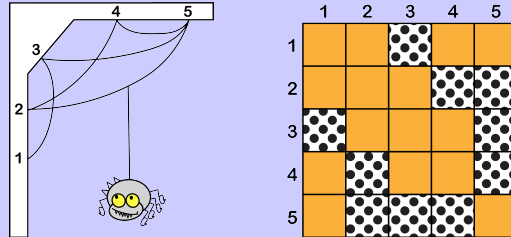
Spider Quilts

Story

When Wanda sees an interesting spider web, it inspires her to design a new quilt. She first numbers the places where the web is anchored to the wall from 1 to n . Then she arranges dotted  and solid  fabric squares into an n by n grid as follows:

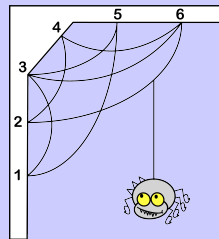
- For every piece of web silk, if its anchors are numbered x and y , she places:
 - one dotted fabric square where row x and column y meet, and
 - another dotted fabric square where row y and column x meet.
- Wanda fills the rest of the grid using solid fabric squares.

For example, the spider web on the left inspired Wanda to design the quilt on the right.

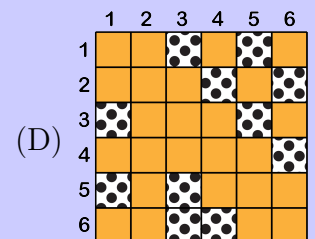
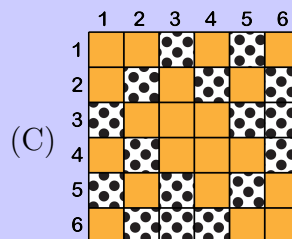
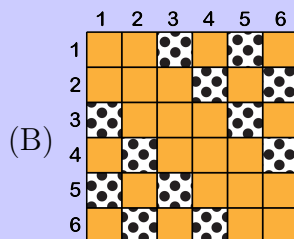
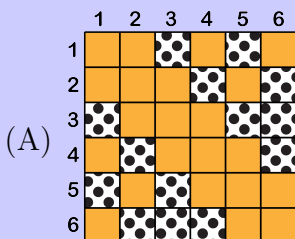


Question

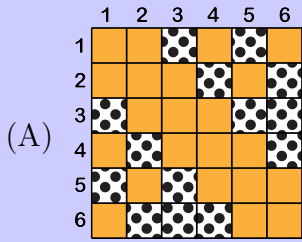
Wanda now sees the following spider web.



What new quilt will this inspire her to design?



Answer



Explanation of Answer

Silk joins anchor 1 with anchors 3 and 5, so row 1 will have dotted fabric in columns 3 and 5.
Silk joins anchor 2 with anchors 4 and 6, so row 2 will have dotted fabric in columns 4 and 6.
Silk joins anchor 3 with anchors 1, 5, and 6, so row 3 will have dotted fabric in columns 1, 5, and 6.
Silk joins anchor 4 with anchors 2 and 6, so row 4 will have dotted fabric in columns 2 and 6.
Silk joins anchor 5 with anchors 1 and 3, so row 5 will have dotted fabric in columns 1 and 3.
Silk joins anchor 6 with anchors 2, 3, and 4, so row 6 will have dotted fabric in columns 2, 3, and 4.

The rest of the quilt will have solid fabric squares.

Option B is missing dotted fabric in row 3 column 6 and in row 6 column 3. Option C has extra dotted fabric placed in row 2 column 2 and in row 5 column 5. Option D cannot be correct since it is not symmetrical. That is, if we draw a line that divides the web in half from the top left to the bottom right, these two halves are symmetrical about the line and so this must also be the case for the quilt.

Connections to Computer Science

The spider web can be considered as a *graph*, a concept that is often used in computer science.

A graph is composed of *vertices* (the anchor points of the web) and *edges* (the pieces of silk between two anchor points). Graphs are used to represent objects and the relationships between objects. For example, a graph could show people who are friends on social media, or flights between countries.

In this task, Wanda's quilt demonstrates a way to represent a graph, known as an *adjacency matrix*. Adjacency matrices are useful representations since they provide an efficient way to answer questions about the structure of a graph. In particular, a question like "Does a particular edge exist in the graph?" is quick and easy to answer with an adjacency matrix.

Country of Original Author

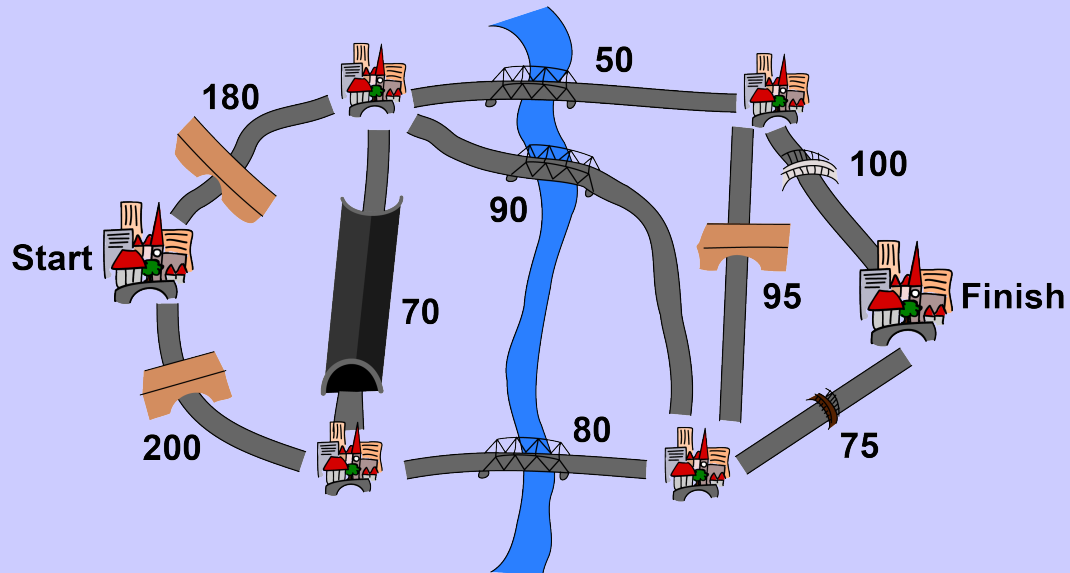
Canada



Travelling Trucks

Story

Trucks travel between six cities using the roads shown in the diagram. Each road has a bridge or tunnel that limits the height of a truck that can travel along it. The maximum truck height for each road is indicated in the diagram.



Question

What is the maximum height of a truck that can travel from Start to Finish?

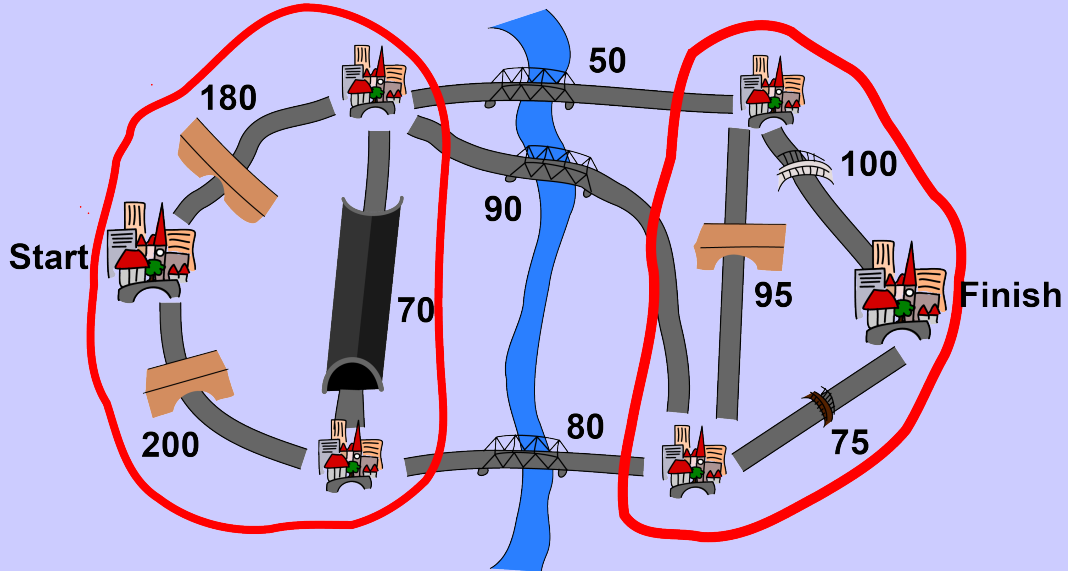
- (A) 70
- (B) 80
- (C) 90
- (D) 95

Answer

(C) 90

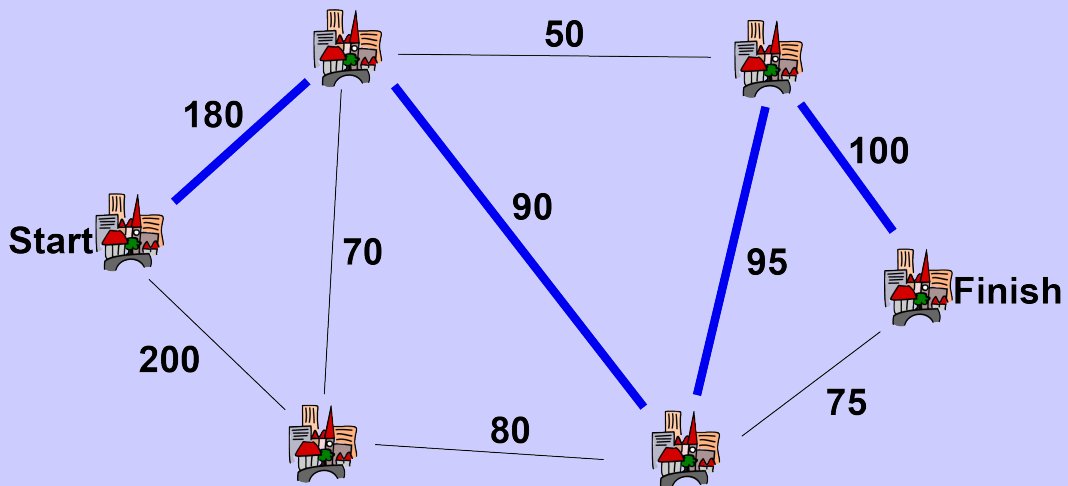
Explanation of Answer

In order to travel from Start to Finish, a truck must pass from the group of three cities on the left to the group of three cities on the right.



To do so, the truck must take one of the three middle roads, with maximum heights of 50, 90, and 80. This tells us that a truck with a height greater than 90 cannot make the desired trip.

Now it remains to verify that a truck of height 90 can make the desired trip. One option is for such a truck to take the path indicated in the following diagram.



Notice that each road on the indicated path has a maximum height of at least 90.

Connections to Computer Science

This task is a representation of how data flows through a *computer network*. In a computer network, whether it is connected via wifi or a wires, there are capacity limits on how much data can flow between two points, which is known as the *bandwidth* between two points. These bandwidth can be on the wireless connection, the physical wires, or the *routers* which direct *data packets* through the network.

In this task, each bridge or tunnel represents the bandwidth through a given point, and the key problem to solve is what route has enough bandwidth to support the amount of information that needs to flow.

Country of Original Author

Cyprus



Picket Painting

Story

A beaver wants to paint as many pickets of a fence as possible using the following cans of paint.



The amount of paint in one can is exactly the amount needed to paint one picket.

Two half cans of different colours can be mixed to paint one picket but the paint cannot be mixed in any other way. Mixing yellow and blue makes green. Mixing red and yellow makes orange. Mixing red and blue makes violet. This means that there are six possible colours for the pickets.

The fence must be as colourful as possible. Specifically:

- One colour cannot be used to paint two pickets unless there is at least one picket of every other colour.
- One colour cannot be used to paint three pickets unless there are at least two pickets of every other colour.

Question

How many pickets can be painted in total?

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

Answer

(B) 8

Explanation of Answer

Before painting two pickets the same colour, there must be one picket of each colour. This requires the beaver to use one can of blue paint for the blue picket, a half can of blue paint for the violet picket, and a half can of blue paint for the green picket. In total, it requires two cans of blue paint. Similarly, two cans of each of red and yellow paint are required. The beaver is then left with two cans of red paint, one can of blue paint, and no cans of yellow paint.

At this point, the beaver can only use the remaining blue paint one time – to paint a second picket blue or a second picket violet. Regardless, the beaver will then only be able to paint a second picket red giving a total of eight painted pickets.

Connections to Computer Science

As shown in this task, two colours can be combined to form new colours. There are various ways that colours are represented, stored, and displayed on a computer.

RGB colour coding is probably the most common method of colour coding. To define a colour, a number between 0 and 255 is set for each of the colours red (R), green (G), and blue (B). For these three numbers, 0 means “none” and 255 means “all”. Note that we are coding light, not ink or paint. A higher figure means more light. The higher the RGB values, the lighter the colour. The lower the RGB values, the darker the colour. For example, we can create colours such as:

- white for which $(R,G,B) = (255, 255, 255)$
- light grey for which $(R,G,B) = (20,20,20)$
- yellow for which $(R,G,B) = (255,255,0)$
- black for which $(R,G,B) = (0,0,0)$

The main purpose of the RGB colour model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers.

The RGB colour model is an *additive* colour model in which red, green, and blue light are added together in various ways to reproduce a broad array of colour. The RGB colour model is essentially opposite to the *subtractive* colour model, given by the *RYB colour model*, using colours red, yellow, and blue, which this task illustrates.

Country of Original Author

Lithuania





Acorns and Mushrooms

Story

A squirrel enjoys acorns and mushrooms as a treat. It creates four piles of treats with seven treats in each pile. Then it adds an eighth treat to each pile as follows:

- If there is an even number of acorns , then it adds a mushroom .
- If there is an odd number of acorns , then it adds another acorn .

Later, a rival squirrel changes two piles by swapping a random  from one pile with a random  from another pile. Now the piles of treats look like this:



Pile 1



Pile 2



Pile 3



Pile 4

Question

Which two piles did the rival squirrel change?

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

Answer

(B) Piles 2 and 3

Explanation of Answer

We focus on the effect of the last treat added by the squirrel on the number of acorns in the pile. The motivation for this comes from the observation that the rule itself depends on the number of acorns:

- If there is an even number of acorns among the first 7 treats, the addition of a mushroom does not change the number of acorns. In particular, it does not change the fact that there is an even number of acorns in the pile.
- If there is an odd number of acorns among the first 7 treats, the addition of another acorn increases the number of acorns by 1. Thus, the number of acorns now becomes even.

The key insight here is that every pile of treats should have an even number of acorns after a treat is added to each pile. If the number of acorns is odd, then the pile has been changed by the rival squirrel.

- Pile 1 has 4 acorns.
- Pile 2 has 3 acorns.
- Pile 3 has 7 acorns.
- Pile 4 has 2 acorns.

Piles 2 and 3 have an odd number of acorns. Since only two piles are changed, the correct answer is Option B.

Connections to Computer Science

The addition of either an acorn or a mushroom depending on the number of acorns is the key to identifying which piles are affected by the rival squirrel's swapping. In computer science, particularly in the fields of *information theory* and *coding theory*, this task focusses on *error detection*.

Errors can occur whenever data is transmitted over a network, and both error detection and *error correction* are important processes that address these errors. When data is stored or transmitted on a computer, it is represented with *binary digits*, or *bits*, 0 and 1. In this task, we can view each acorn as a 1, and each mushroom as a 0.

One of the simplest error detection schemes is *parity checking*. *Parity* refers to whether a number is odd or even. There are two variants of this technique: *even parity* and *odd parity*. In both variants, an extra bit (either a 1 or a 0) – referred to as the *check bit* – is appended to the data. In the case of even parity, the check bit is set to 0 if there is already an even number of 1's; otherwise, it is set to 1. The opposite happens in the case of odd parity. In this task, the last treat added by the squirrel acts as an even parity check bit.

Consequently, if even parity is employed and the data (together with the check bit) contains an odd number of 1's, then an error is detected. For instance, sending 1000101 requires that an even parity bit of 1 is appended: 10001011. If, for some reason, the second bit becomes corrupted, transforming the data to 11001011, then we are able to recognize the presence of an error since 11001011 has an odd number of 1's – a violation of the parity rule.

Parity checking only works if the number of corrupted bits is odd. This is the reason why this task specifies that the rival squirrel swaps only a *single* acorn from one bag with a *single* mushroom from another. Although this error detection technique does not tell us which bits are corrupted (that is, which treats are swapped) nor does it provide us with any way to repair the data, it is still widely used in some hardware components, such as *PCI buses*, inside the computer.

Country of Original Author

Philippines

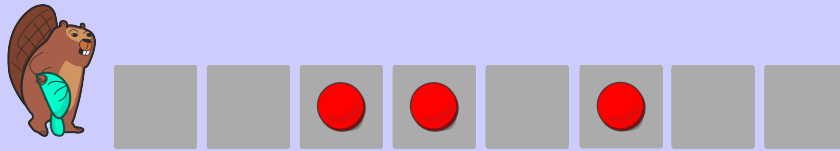


Part C

Hat Game

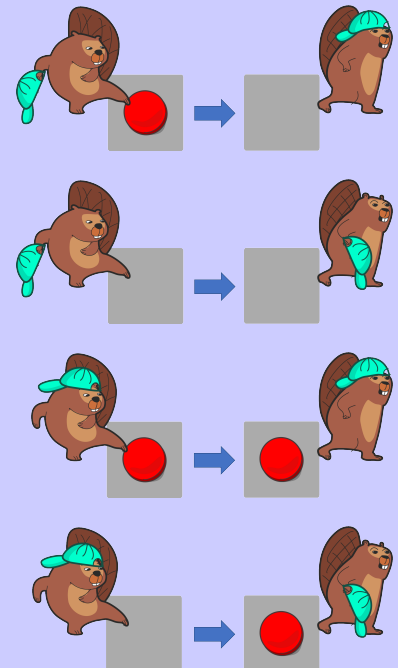
Story

A beaver plays a game with chips, a hat, and the gameboard shown.



Each square on the gameboard either contains one chip or no chip. The beaver starts with a hat in its hand. It then steps on the squares one at a time from left to right, and acts according to the following rules until it moves off the gameboard:

- If the beaver has the **hat in its hand** and steps on a square that **contains a chip**, then the beaver removes the chip from the square, puts the hat on its head, and then moves to the next square.
- If the beaver has the **hat in its hand** and steps on a square that **does not contain a chip**, then the beaver simply moves to the next square.
- If the beaver has the **hat on its head** and steps on a square that **contains a chip**, then the beaver simply moves to the next square.
- If the beaver has the **hat on its head** and steps on a square that **does not contain a chip**, then the beaver puts a chip on the square, puts the hat in its hand, and then moves to the next square.



Question

What does the final gameboard look like once the beaver has moved through all eight squares?

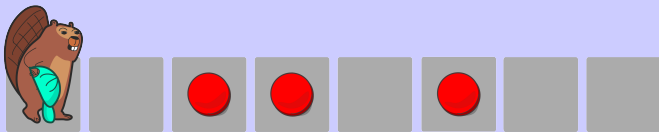
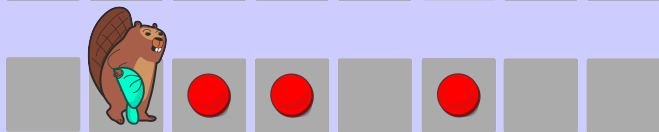
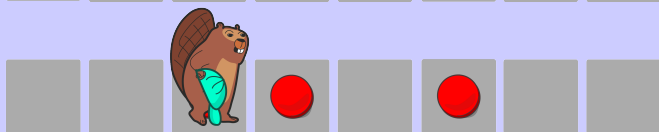


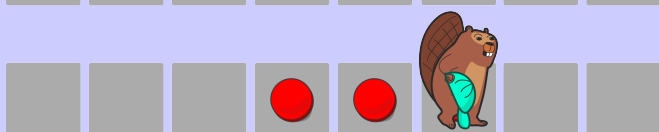
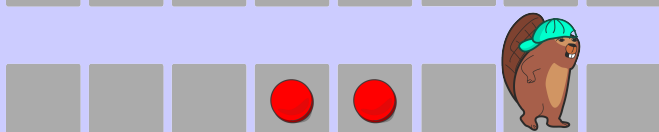
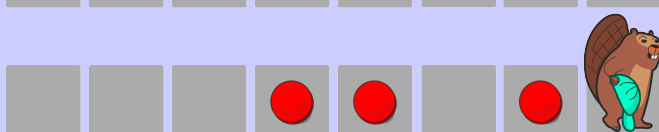
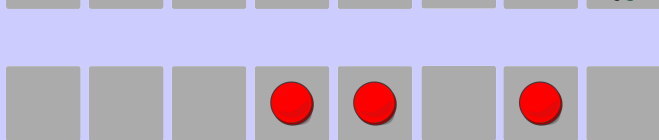
- (A) (B) (C) (D)

Answer



Explanation of Answer

The following diagram illustrates the movements and actions of the beaver as it passes through each of the eight squares in turn. The ninth image shows the final gameboard.

①		Square 1: Hat in hand and no chip. Move to the next square.
②		Square 2: Hat in hand and no chip. Move to the next square.
③		Square 3: Hat in hand and chip. Remove chip and put hat on head. Move to the next square.
④		Square 4: Hat on head and chip. Move to the next square.
⑤		Square 5: Hat on head and no chip. Add chip and put hat in hand. Move to the next square.
⑥		Square 6: Hat in hand and chip. Remove chip and put hat on head. Move to the next square.
⑦		Square 7: Hat on head and no chip. Add chip and put hat in hand. Move to the next square.
⑧		Square 8: Hat in hand and no chip. Move to the next square.
⑨		Final gameboard

Connections to Computer Science

This task is a simple representation of a *Turing machine*, which is a *model of computation* first described by the English mathematician and computer scientist Alan Turing in 1936.

A Turing machine is composed of a *control unit* and a *memory tape*.

The memory tape is a collection of squares, viewed as infinitely long in either direction, with each square holding one symbol, either 0 or 1. In this task, the tape is finite in length, and the symbols 0 and 1 are represented by an empty square or a chip on the square, respectively.

The control unit has three key components:

- A *read/write head*, which looks at a square and reads the symbol on that square, possibly writing a new symbol, and then moving left or right one square or staying on the same square. In this task, the beaver represents the read/write head.
- A *finite set of states* that the control unit can be in. In this task, there are two states, but in general there can be any finite number of states. The two states in this task are “hat in hand” and “hat on head.”
- A set of *transition rules*, which specify how the machine operates. That is, each transition rule will use the current state and the symbol on the current square on the tape to determine what the next state is, what symbol to write on the current square, and what direction to move on the tape. In this task, there are four transition rules stated in the task.

Although a Turing machine seems very simple, it is as powerful and efficient as any program running on any computer: we can convert any program into a Turing machine and, conversely, any Turing machine into a program. Notice that computers we use today have two key components: the *central processing unit (CPU)* and *random access memory (RAM)*, which are equivalent to the control unit and tape in a Turing machine.

Country of Original Author

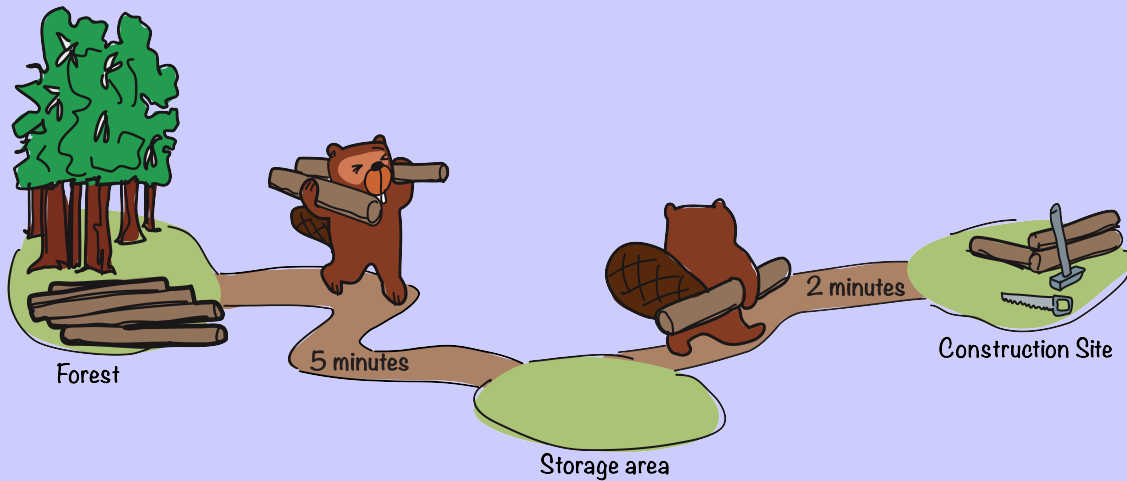
Switzerland



Logs

Story

Nila and Sam are building a log house. Nila delivers logs from the forest to the storage area. She carries 2 logs per trip and it takes her 5 minutes to make the trip in either direction. Sam delivers logs from the storage area to the construction site. She carries 1 log per trip and it takes her 2 minutes to make the trip in either direction.



When Nila arrives at the storage area, she immediately returns to the forest and vice versa.

As soon as there is 1 log at the storage area and she is no longer carrying logs, Sam immediately heads to the storage area, and then immediately back to the construction site. Otherwise, she remains at the construction site.

When work begins, Nila is at the forest, Sam is at the construction site, and all logs are in the forest.

Question

How many logs will be at the construction site 30 minutes after work begins?

- (A) 3
- (B) 4
- (C) 5
- (D) 6

Answer

(C) 5

Explanation of Answer

The earliest that Nila can deliver logs to the storage area is 5 minutes after beginning to work. It will take 10 minutes to return to the forest, obtain 2 logs, and bring them to the storage area, and another 10 minutes to get another 2 logs to the storage area. To summarize, Nila can deliver 2 logs to the storage area 5 minutes, 15 minutes, and 25 minutes after beginning work. These are the only logs that she can deliver in the first 30 minutes since it would take at least $25 + 10 = 35$ minutes to deliver 8 logs.

It takes Sam 4 minutes to leave the construction site, get 1 log from the storage area, and bring it back to the construction site. At 5 minutes after beginning work, Sam will head to the storage area and make 2 round trips to collect the 2 logs. This will take a total of $4 + 4 = 8$ minutes, so at $5 + 8 = 13$ minutes after beginning work, she will remain at the construction site and 2 logs will be there.

Starting at 15 minutes after beginning work, Sam will spend another 8 minutes bringing the 2 new logs from the storage area to the construction site. Thus, at $15 + 8 = 23$ minutes after beginning work, there will be 4 logs at the construction site.

Starting at 25 minutes after beginning work, Sam will begin collecting logs again. By 29 minutes after beginning work, she will have collected 1 log from the storage area and delivered it to the construction site. However, she will not have time to retrieve the sixth log. Therefore, a total of 5 logs can get to the construction site in the first 30 minutes.

Connections to Computer Science

The way the two friends are working together can be viewed as the *producer-consumer model* of *parallel processing* in computers. Nila is the producer of logs for Sam, and Sam is the consumer of the logs that Nila has produced.

The storage area acts as a *buffer* so that Nila does not have to wait until Sam comes to collect the logs; instead, Nila can return to forest for the next pair of logs immediately and be more productive. We might imagine Nila calling out to Sam when she adds new logs to the empty storage area. This would be similar to how *signals* are used in computer systems to allow one program/process to alert another. This lets Sam do other work instead of just waiting at the storage area. However, if Nila did call to Sam, it would take some time for Sam to go from the construction site to the storage area, causing *latency* in the movement of logs.

The concept of buffers, signals, and latency are all key concepts in how the *central processing unit* manages shared resources for multiple programs or processes.


Country of Original Author

Estonia





Unlock the Crown

Story

A crown  is locked in one of 15 drawers as shown.



There is a keyhole at the top of each drawer. To open the drawer, you must insert an object with the same shape as the keyhole. For example, for the keyhole  on the top left drawer, you must insert an object shaped like a diamond.

Each drawer contains one object as indicated on the front of the drawer below the keyhole. For example, the top left drawer contains an object shaped like a heart .

Question

Bella has an object shaped like a circle. What is the minimum number of drawers that Bella needs to open in order to retrieve the crown?

- (A) 3
- (B) 4
- (C) 5
- (D) 6

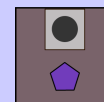
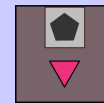
Answer

(C) 5

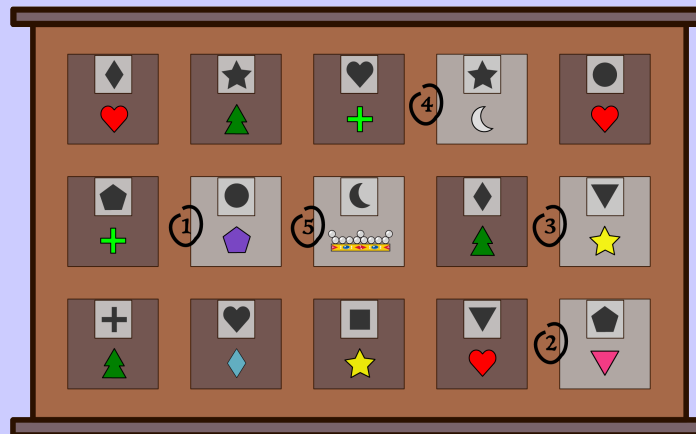
Explanation of Answer

Working backwards from the drawer with the crown, we can determine which drawers Bella must open to retrieve the crown.

1. To retrieve the crown, Bella must open the middle drawer. This means she needs to retrieve a moon.
2. From #1, we know Bella must open the drawer containing a moon. This means she needs to retrieve a star.
3. From #2, we know Bella must open one of the two drawers containing a star. This means she needs to retrieve either a square or a triangle. Since there are no drawers containing a square, she needs to retrieve a triangle.
4. From #3, we know Bella must open the drawer containing a triangle. This means she needs to retrieve a pentagon.
5. From #4, we know Bella must open the drawer containing a pentagon. She can do this using the circle she began with.



This work tells us that Bella must open at least 5 drawers in order to retrieve the crown. It also tells us that the 5 drawers highlighted in the following diagram can be used to retrieve the crown in the order indicated. Therefore, 5 is the minimum number of drawers that Bella needs to open.



Connections to Computer Science

The ways in which each drawer can be opened can be modelled by a *directed graph*. We can represent each drawer as a *vertex* of the directed graph. Each (*directed*) *edge* will go from a drawer with a certain symbol to a drawer with the keyhole matching that symbol. In the diagram, the edges are *implicit*, since they are not actually part of the picture, but are rather inferred from the information available on each drawer.

The reason we say the graph is “directed” is because the opening of a drawer is only valid in one direction. For example, if drawer A contains a symbol that opens the keyhole of drawer B, we would have a directed edge from drawer A to drawer B, and it is not necessarily the case that the symbol in drawer B opens the keyhole for drawer A. In this case, we call B a *neighbour* of A.

In this problem, we want to find a *directed path* to the crown. One way to solve this is to perform a *breadth-first search*, where all of the neighbours of the crown are determined, and then repeatedly determine the neighbours of each of those neighbours, and so on.

The idea of searching for a path in a directed graph has many applications, such as mapping out a driving route, determining how to send information through the internet, and determining recommendations for who you may want to connect with on social media platforms.

Country of Original Author

Iceland

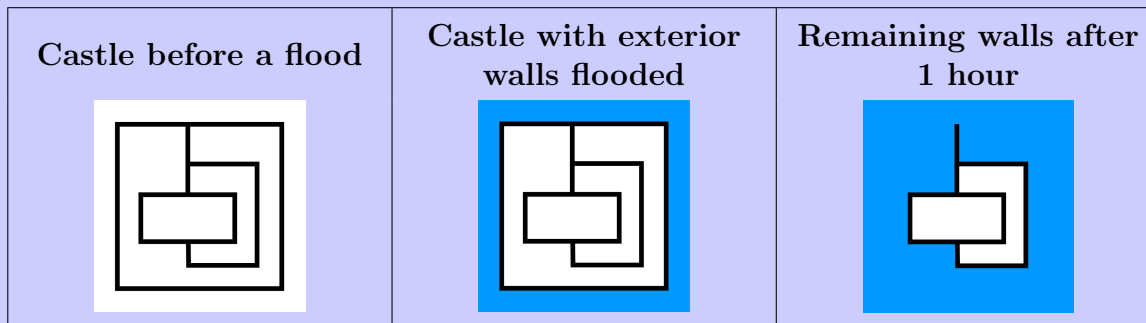


Flooding

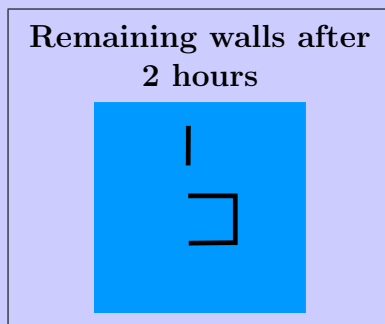
Story

A large flood in the land of Bevaria destroys many of its castles' walls.

First, water floods the exterior of a castle. Then after 1 hour, every wall that has water on one side but not the other breaks under the pressure of the water, and is destroyed. Walls with water on both sides, or water on neither side, remain intact. The water then floods any new exterior walls. For example:



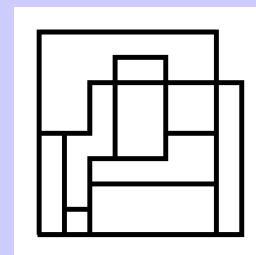
This process repeats until water has flooded the entire area of the castle. In our example, it takes a total of 2 hours to flood the entire area after the water floods the exterior walls. Notice that some walls remain after all the flooding.



Question

After water floods the exterior of the castle shown, how many hours will it take to flood the entire area?

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

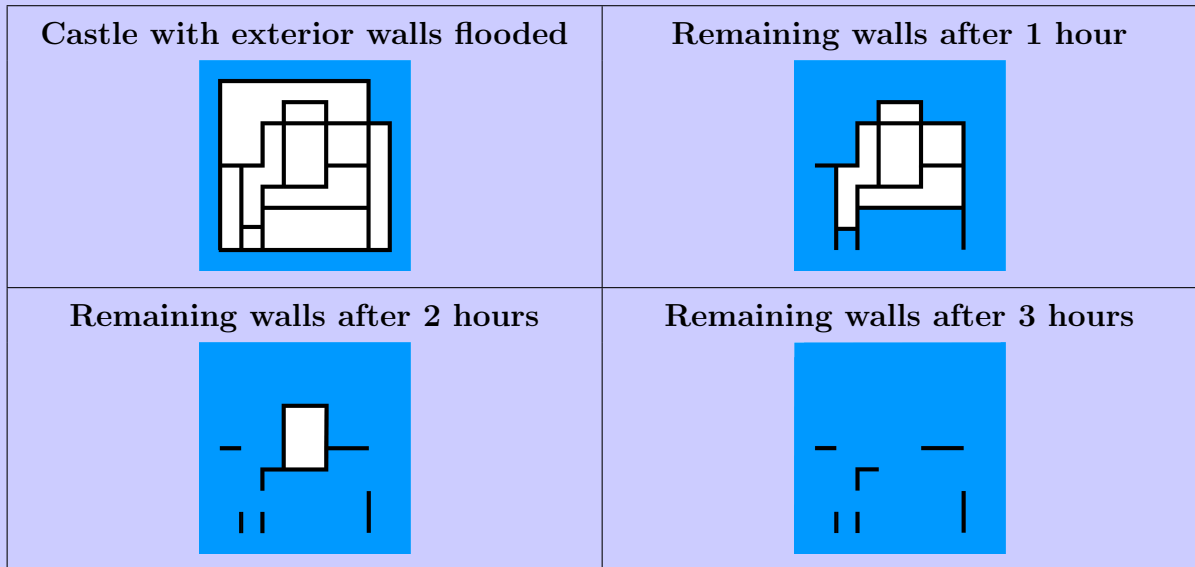


Answer

(B) 3

Explanation of Answer

The process of flooding the castle is illustrated in the following diagrams.



After 2 hours, one central region of the castle has not yet flooded. After 3 hours, the entire area of the castle has flooded.

Connections to Computer Science

In order to solve this task, we can use a technique called *breadth first search*, and specifically, a variant called the *flood fill* algorithm.

Breadth first search is an *algorithm* or procedure that searches by looking at all immediate “next locations” based on the “current location”. In this task, the next locations are the interior spaces behind an exterior wall which is being flooded. This process repeats, with the most recently added “next locations” becoming the “current location”. When observing the pictures used in the Explanation of Answer, the entire area is increasingly filled as if it were being flooded, and thus, the algorithm is known as *flood fill*.

One very common use of breadth-first search is in finding the *shortest path* between two given points, such as when finding the best sequence of directions to drive from one location to another. In this task, we are finding the shortest path from the outside to the most interior space.

Flood fill algorithms can be found in many graphical drawing programs which contain a “bucket fill” tool that will shade an entire region with a certain colour: the algorithm will colour adjacent *pixels* the same colour until it locates a boundary pixel of a different colour.

Country of Original Author

Portugal

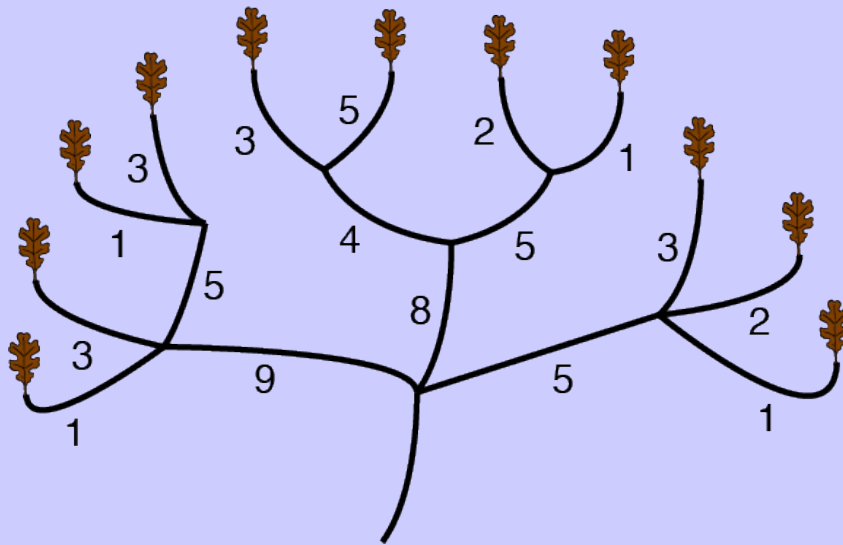


Cutting Branches

Story

Removing dead leaves from a tree can encourage new growth. Dead leaves are removed by cutting branches.

The following tree has 11 dead leaves. The time needed (in minutes) to cut each branch is shown.



When a branch is cut, all branches and leaves attached to it are removed from the tree. For example, if you cut the branch that takes 9 minutes, the four leftmost leaves are removed.

Question

What is the shortest amount of time needed to remove all 11 dead leaves from this tree?

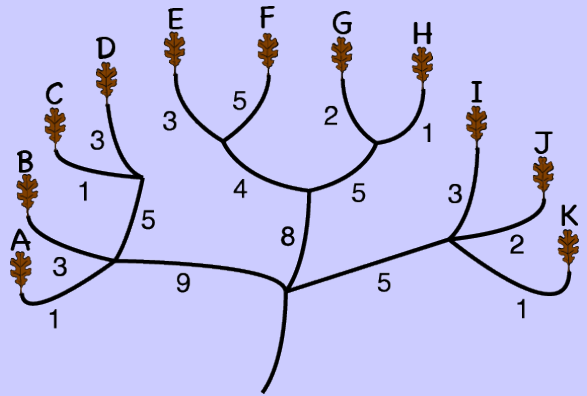
- (A) 19 minutes
- (B) 20 minutes
- (C) 22 minutes
- (D) 25 minutes

Answer

(B) 20 minutes

Explanation of Answer

The leaves have been labelled A through K in the diagram. Notice that the four leftmost leaves (A, B, C, and D) do not share any branches with the rest of the leaves (E through K). Therefore, removing these leaves will have no impact on the rest. The same is true for the four middle leaves (E, F, G, and H) and the three rightmost leaves (I, J, and K). This means that the shortest amount of time needed to remove all 11 dead leaves is the sum of the shortest amount of time needed to remove the leftmost leaves, the middle leaves, and the rightmost leaves.

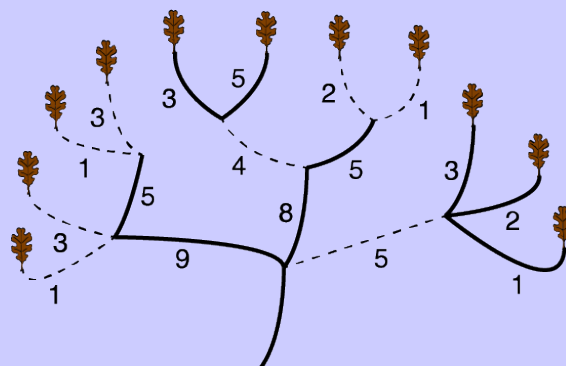


To remove the leftmost leaves we have three options: remove them all at once (9 minutes), remove them individually ($1 + 3 + 1 + 3 = 8$ minutes), or remove A and B individually and C and D together ($1 + 3 + 5 = 9$ minutes). The shortest amount of time needed to remove the leftmost leaves is thus 8 minutes.

To remove the rightmost leaves we have two options: remove them all at once (5 minutes), or remove them individually ($3 + 2 + 1 = 6$ minutes). The shortest amount of time needed to remove the rightmost leaves is thus 5 minutes.

To remove the middle leaves we have five options: remove them all at once (8 minutes), remove them individually ($3 + 5 + 2 + 1 = 11$ minutes), remove E and F individually and G and H together ($3 + 5 + 5 = 13$ minutes), remove E and F together and G and H individually ($4 + 2 + 1 = 7$ minutes), or remove E and F together and G and H together ($4 + 5 = 9$ minutes). The shortest amount of time needed to remove the middle leaves is thus 7 minutes.

Therefore, the shortest amount of time needed to remove all 11 dead leaves is $8 + 5 + 7 = 20$ minutes and this is achieved when the following branches are cut:

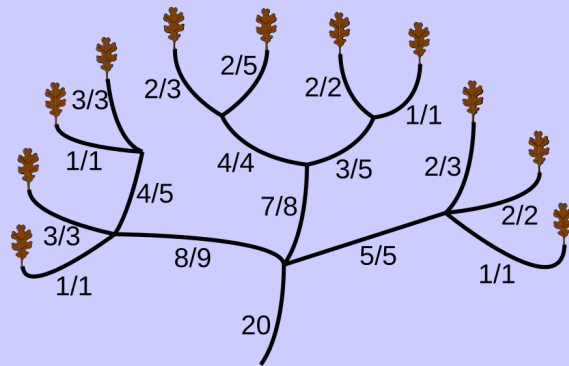


Connections to Computer Science

This task focusses on *trees*, which are one of the most important *data structures* in computer science. Trees are composed of *nodes* that are connected to each other by *edges* such that there are no *cycles* in any connection: that is, no node can take a path back to itself. In this problem, the nodes consist of the leaves (called *leaf nodes* and the points where different branches meet. The bottom-most node in the diagram is considered the *root node*.

This task is also focussed on finding the *minimal cut* in the tree so that every leaf node is disconnect from the root node: that is, what is the least amount of cutting required to ensure there is no *path* from the root node to any leaf node.

In order to find the minimal cut, we can solve the *dual problem* of finding the *maximum flow* that can go from the root to the leaf nodes. That is, if we think of each edge as having a maximum capacity for the amount that can flow on it, what is the maximum we can put on the tree, starting from the root. A solution for maximum flow on this tree is as follows:



Notice that each edge has a label of the form “flow/capacity”, where each flow is less than or equal to the capacity on that edge.

Finding the maximum flow is widely used in logistics. For example, it can be used to determine the maximum weight of goods that can be transported from the factories to other countries taking into account all means of transportation between major cities and their maximum capacities.

Country of Original Author

Uzbekistan

