

WATERLOO | MATHEMATICS

The CENTRE for EDUCATION in MATHEMATICS and COMPUTING



2015
*Beaver
Computing
Challenge
(Grade 9 & 10)*

*Questions,
Answers,
Explanations,
and
Connections*

Part A

Popularity

Story

Five beavers join an online social network resulting in the following activity:

- Ari follows Bob
- Ari follows Dmitri
- Bob follows Chio
- Bob follows Dmitri
- Chio follows Ari
- Chio follows Bob
- Dmitri follows Ehab
- Ehab follows Bob

Then, we track how popular various beavers are. We give n popularity points to beaver Y if beaver X follows beaver Y and there are n beavers that follow beaver X . We compute beaver Y 's total popularity points by adding up all such points from all the beavers that follow beaver Y . For example, Bob has 3 popularity points.

Question

Which beaver has the most popularity points?

- (A) Ari
- (B) Bob
- (C) Chio
- (D) Dmitri

Answer

(D) Dmitri

Explanation of Answer

The following calculations show that Dmitri has the most popularity points:

Beaver	Followers	Number of Followers	Popularity Points
Ari	Chio	1	1
Bob	Ari, Chio, Ehab	3	$1 + 1 + 1 = 3$
Chio	Bob	1	3
Dmitri	Ari, Bob	2	$1 + 3 = 4$
Ehab	Dmitri	1	2

Connections to Computer Science

If we think of the beavers as websites and “following” as “linking to”, then this is an example of a very small *internet*. *Search engines* typically *rank* internet results by some measure of popularity or importance.

Social networks themselves are incredibly powerful tools in today’s world. Computing statistics on their users and pages is useful to marketers and anyone else trying to understand a person or group of people.

Country of Original Author

Canada

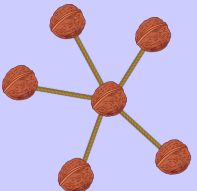
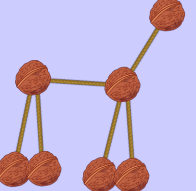
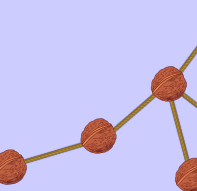
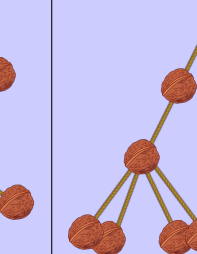


Chestnut Animals

Story

Tommy Beaver was inspired by the picture of an animal made from nuts (shown to the right), and created 4 animals by himself using chestnuts, strings and glue (shown below):

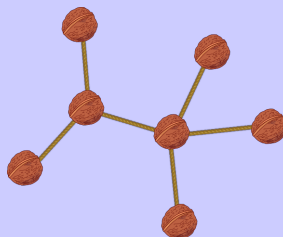


Starfish	Dog	Sea lion	Giraffe
			

His sister plays with these animals by moving the chestnuts around without breaking any connections. This makes it hard to recognize which shapes correspond to which animals.

Question

Which animal was the following shape before Tommy Beaver's sister played with it?



- (A) Starfish
- (B) Dog
- (C) Sea lion
- (D) Giraffe

Answer

(B) Dog

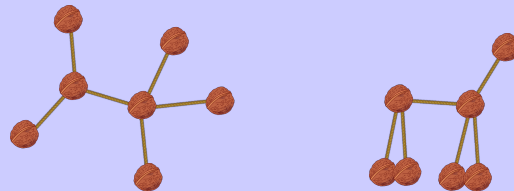
Explanation of Answer

Each animal is determined by the connections between its parts. The specific positions of chestnuts and the angles between connections may change when moved, but that does not change the connections themselves.

Therefore, we can look at the connection count of each of the chestnuts in his sister's shape:

- 5 chestnuts are connected to only one other chestnut;
- 1 chestnut is connected to 3 other chestnuts;
- 1 chestnut is connected to 4 other chestnuts.

There is only one animal that has these exact connections, which is the dog. Notice that the starfish and sea lion only have 6 chestnuts, and the giraffe has one chestnut with 5 connections.



Connections to Computer Science

With chestnut animals, we *abstract* from features like fur and size. We represent the animal only by the structure of its body; the rest is unimportant. This structure is preserved even when the animals are transformed. A computer scientist must recognize what is important, what can be left out, and how two structures are similar.

The representation of each animal is by way of a *graph*: a set of *vertices* (in this case, the chestnuts) and a set of *edges* between pairs of vertices (in this case, the connections between chestnuts). The underlying mathematical problem we are trying to solve in this task is the *graph isomorphism* problem: given two graphs, is their structure the same? This problem is very difficult to solve efficiently for very large graphs. However, since the graphs are very small (less than 8 vertices) in our task, we can look for particular *vertex degree* matches (i.e., the number of connections of each vertex) in order to determine which two graphs are isomorphic.

Country of Original Author

Czech Republic



Catch Up

Story

Allison Beaver has a pile of 10 trees. Beatrice Beaver has only 1 tree.



Allison Beaver and Beatrice Beaver start chewing trees in a forest at exactly the same time. They add every tree they chew down to their own pile of trees.

Allison Beaver chews down one tree per hour.

Beatrice Beaver chews down trees at a different rate. In the first hour, she will chew down one tree. In the second hour, she will chew down two trees. In the third hour, she will chew down three trees, and so on.

Question

After how many hours will Beatrice first have at least as many trees as Allison?

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

Answer

(B) 5

Explanation of Answer

Consider the following table:

Time	Alison Pile	Beatrice Pile
Start	10	1
After 1 hour	11	2
After 2 hours	12	4
After 3 hours	13	7
After 4 hours	14	11
After 5 hours	15	16

Connections to Computer Science

This question focuses on the *asymptotic analysis* of algorithms.

We would say that Allison chews trees at a *linear rate*, which we would represent as $O(n)$. That is, after n hours, she would have a total number of chewed trees that is proportional to n . Specifically, the number of trees Allison chews after h hours is $10 + h$.

For Beatrice, we would say that she chews trees at a *quadratic rate*, which we would represent as $O(n^2)$, meaning that after n hours, Beatrice would have a total number of trees proportional to n^2 . Specifically, the number of trees Beatrice chews after h hours is

$$1 + (1 + 2 + 3 + \cdots + h) = 1 + \frac{h(h+1)}{2} = \frac{1}{2}h^2 + \frac{1}{2}h + 1.$$

The asymptotic analysis of algorithms is very important in computer science for several reasons. First, it allows us to compare algorithms using only the “most important term”. That is, whether the running time of a program is $n^2 + 10$ or $n^2 + 200$, the difference between these two running times becomes insignificant when n is large enough. Secondly, asymptotic analysis gives a good estimate of the running time as the size of the input grows. Throughout computational history, the size of problems has grown: more airline flights need organizing, more stock market transactions to reconcile, etc. Asymptotic analysis gives us a clear picture about how algorithms will perform when the input size gets very large. Finally, asymptotic analysis allows comparisons between different algorithms (as we have done between the two algorithms here) to be treated formally and mathematically.

Country of Original Author

Canada



QB-Code

Story

Beavers want to encode numbers for keeping track of how many trees they have chewed down. Therefore they developed the Quick-Beaver-Code (QB-Code). This is a graphical code consisting of nine 1×1 squares arranged into a 3×3 square. Every square has a certain value. The squares are filled line by line from the bottom to the top, from right to left. The next square has double the value of the square before. In the example, you see the values of the first five squares.

...
...	16	8
4	2	1

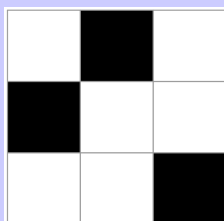
To encode a number, the beavers darken some squares. The number encoded is the sum of the values of the dark squares.

For example, the number encoded in this QB-Code is 17:



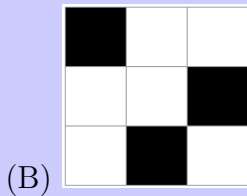
Question

Which of the following encodes the largest number?



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

Answer



Explanation of Answer

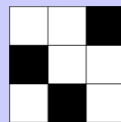
The answer can be obtained without doing any complicated calculation. The square in the top-left corner has the highest value (256). Note that the sum of all the other squares (i.e., $128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$) is less than 256. Hence, the highest possible number encoded is the QB-Code with the top-left corner darkened, and there is only one answer with the top-left corner darkened.


Connections to Computer Science

QB codes look much like *QR-codes* (short for “Quick-Response” codes). The QR-code for the mobile Wikipedia page, for instance is:



QB-codes can be thought of as a mapping from a binary number to a decimal number. Specifically, the QB-code



can be rearranged in a row  which can be written as 001100010, when reading the picture from left-to-right, top-to-bottom and treating 0 as “white” (“off”) and 1 as “black” (“on”). QR codes encode multiple numbers, rather than just one number. They are also unambiguous: three out of four corners are marked, so even if the QR code image is rotated, the software knows which rotation is correct. Additionally, information is repeated multiple times within the QR code to increase the robustness and allow *error correction* when the picture quality is poor.

Country of Original Author

Germany

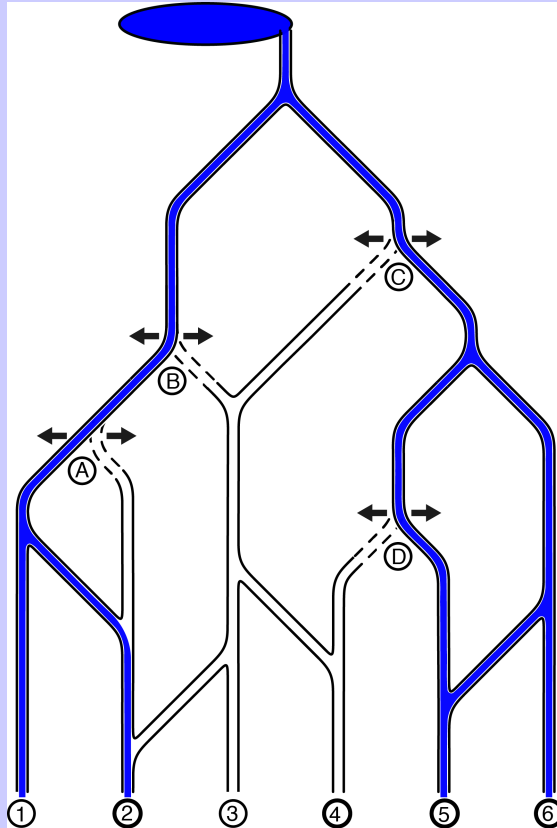


Irrigation System

Story

Beavers have created a nifty irrigation system for their fields. The water flows from a lake at the top of the hill all the way down to the fields numbered 1 to 6 at the bottom.

Along the water canals, the beavers have installed four water gates A to D, where the water can only flow either to the left (←) or to the right (→). An example showing how these may be set to have the water flow to fields 1, 2, 5 and 6 is shown below.



Question

What is the correct configuration for the water gates to irrigate only fields 2, 4, 5 and 6?

- (A) A: ← B: ← C: → D: ←
- (B) A: → B: ← C: ← D: →
- (C) A: → B: ← C: → D: ←
- (D) A: ← B: → C: → D: →

Answer

(C) A: \rightarrow B: \leftarrow C: \rightarrow D: \leftarrow

Explanation of Answer

Answer (A) is incorrect, because field 1 would be irrigated, although it shouldn't be. Answer (B) is incorrect, because field 5 and 6 would not be irrigated, although they should be. Answer (D) is incorrect, because field 3 would be irrigated, although it shouldn't be. We can verify that answer C is correct, since it irrigates exactly fields 2, 4, 5 and 6 and no other fields.

Connections to Computer Science

The irrigation system behaves like a *directed graph* in *graph theory*. The graph shape is very similar to a *tree* with a *root node* (the lake at the top) and several *leaves* (the fields at the bottom); but in this graph there are directed connections between several vertices, which would not occur in a tree.

Notice that if a field is connected to the root by a directed path, passing through gates A, B, C or D in the specified direction (i.e., with the gate turned to the correct direction), water will flow there. Therefore, fields that need to be irrigated need to have at least one connection to the root node and fields that don't need to be irrigated must not have such a connection.

Country of Original Author

Switzerland



Part B

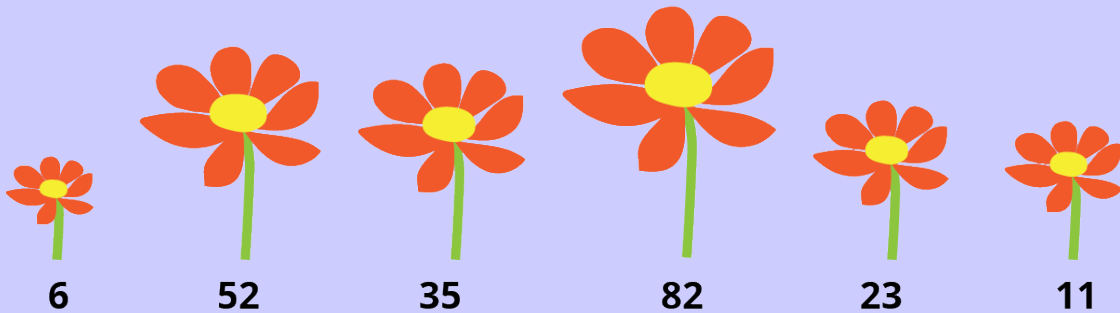
Collecting Pollen

Story

Beever the bee flies to a field of flowers to collect pollen. On each flight, he visits only one flower and can collect up to 10 mg of pollen. He may return to the same flower more than once.



The initial amount of pollen in each flower (in mg) is shown below.



Question

What is the maximum total amount of pollen that Beever can collect in 20 flights?

- (A) 179 mg
- (B) 195 mg
- (C) 196 mg
- (D) 200 mg

Answer

(C) 196 mg

Explanation of Answer

One approach Beaver could take is to collect as much pollen per flight as possible. This begins with Beaver collecting 10 mg of pollen per flight while he can. We use division to calculate how many times he can do this:

$$\begin{aligned}6 \text{ mg} &= 0 * 10 \text{ mg} + 6 \text{ mg} \\52 \text{ mg} &= 5 * 10 \text{ mg} + 2 \text{ mg} \\35 \text{ mg} &= 3 * 10 \text{ mg} + 5 \text{ mg} \\82 \text{ mg} &= 8 * 10 \text{ mg} + 2 \text{ mg} \\23 \text{ mg} &= 2 * 10 \text{ mg} + 3 \text{ mg} \\11 \text{ mg} &= 1 * 10 \text{ mg} + 1 \text{ mg}\end{aligned}$$

After $(0 + 5 + 3 + 8 + 2 + 1) = 19$ flights, he collects $19 * 10 \text{ mg} = 190 \text{ mg}$ of pollen. In his 20th and final flight, Beaver collects the largest amount left over, which is 6 mg. In total, Beaver collects $19 * 10 \text{ mg} + 6 \text{ mg} = 196 \text{ mg}$ of pollen. Notice that making any trip without taking the maximum will yield a total of less than 196 mg.

Notice that once Beaver decides how much pollen to collect on each flight, the order in which the flights happen does not matter. That is, we may take 6 mg from the flower with 6 mg of pollen on any trip, so long as we take 10 mg from each of the other flights.

Connections to Computer Science

To solve this task, we use a *greedy algorithm*. A greedy algorithm is a sequence of steps that makes a choice which is optimal or “the best” for each step. Greedy algorithms work only when an optimal answer to a smaller problem is part of an optimal answer for the overall problem. In this task, Beaver can take as much pollen for each trip for as long as possible.

Greedy algorithms are usually simpler than other approaches, so even in situations where a greedy algorithm does not give an optimal answer, it might be used to give an answer that is “good enough”.

Country of Original Author

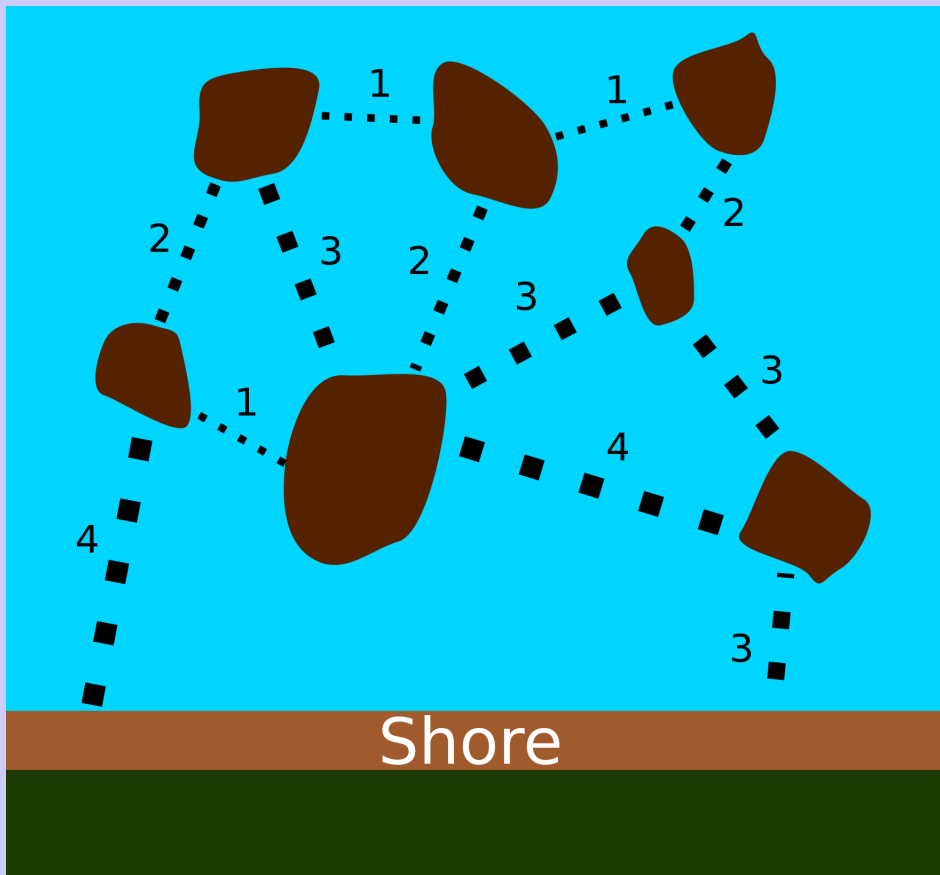
Malaysia



Connecting Beaver Dens

Story

There are seven dens in a pond just off a shore as shown below. Dotted lines show where bridges can be built. The numbers show how many trees are needed to build each possible bridge. A beaver needs to decide which bridges to build so that any den can be reached from the shore without swimming.



Question

What is the fewest number of trees needed to build the bridges?

- (A) 12
- (B) 13
- (C) 17
- (D) 18

Answer

(B) 13

Explanation of Answer

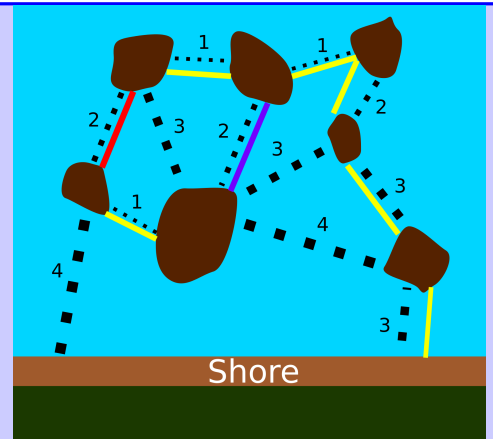
To use as few trees as possible, the yellow bridges and either the red bridge or purple bridge shown to the right should be built. Building these seven bridges requires a total of $1 + 1 + 1 + 2 + 2 + 3 + 3 = 13$ trees.

Why is 13 the best we can do?

Informally, exactly seven bridges are needed. There are seven dens and one shore, making a total of 8 things to be connected. If fewer than seven bridges are built, then at least one den or the shore will not be reachable. More than seven bridges simply requires more trees.

The seven bridges needing the fewest trees require a total of $1 + 1 + 1 + 2 + 2 + 2 + 3 = 12$ trees. However, if only these bridges are built, then it is easy to check that at least one den will not be reachable.

More formally, if we use either of the 4 log bridges along with the next 6 smallest bridges, we will have a total bridge length of 13 and we already have a solution with 13. So we can choose to not take a 4 log bridge. This means we need to include the two 3 log bridges on the bottom right. Taking the next 5 minimum length bridges gives us a bridge length of 13 and hence this must be minimal.



Connections to Computer Science

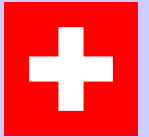
In general, this problem involves finding the least costly way to fully connect a set of objects. The design of circuits and networks is a practical application of *minimum spanning trees*. There are also surprising applications to computer vision, understanding financial markets and hand-writing recognition.

One way to build bridges is found by always building the cheapest (one requiring the fewest trees) unbuilt bridge among those that do not connect two dens that are already connected through the previously built bridges. This is the same as choosing the cheapest unbuilt bridge that does not lead to a circular path along bridges and/or the shore. For example, only one of the red or purple bridge will be added and the other ignored, since if both were added, a circular path (called a *cycle*) would be formed, and we only require a path, and thus, a cycle has an unnecessary edge. This procedure is known as *Kruskal's algorithm*.

Another way of finding the answer begins by choosing the cheapest bridge joining the shore to a den. Then, we repeatedly build the cheapest unbuilt bridge that connects a den that can only be reached by swimming to one that can already be reached using bridges. We stop when every den can be reached without swimming. This procedure is sometimes called the *Prim-Jarník algorithm*.

Country of Original Author

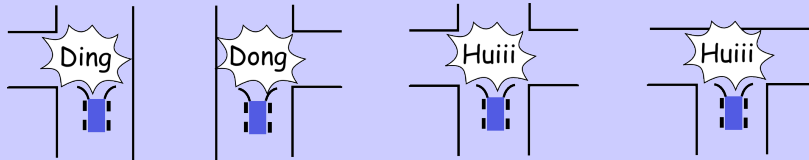
Switzerland



Robotic Car

Story

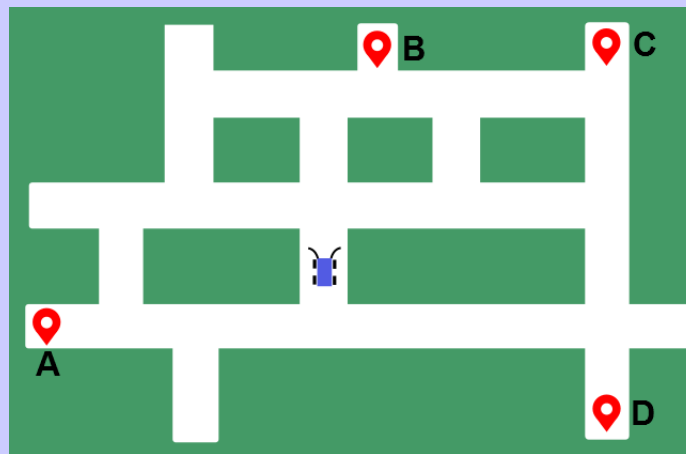
Beavers have developed a robotic car. It has sensors that detect intersections. It produces the sounds shown below, when it is possible to turn left, right or both directions. The robotic car can go straight through an intersection (when possible), turn right (when possible) or turn left (when possible). The robotic car cannot make U-turns and cannot reverse.



It automatically stops when it senses an obstacle in front of it.

Question

The car drives around the map shown below, starting at the indicated position. As it drives around the map, it produces the sounds **Huiii Ding Huiii Dong**, in that order.



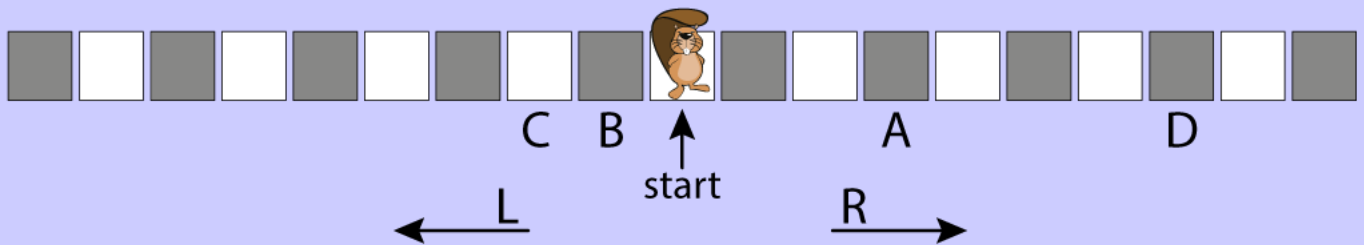
At which location  does the car stop?

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

Jumping

Story

A beaver moves in strange ways. He starts at the middle position, as shown below. He will make five moves, alternating between right (R) and left (L): he first moves right, then left, then right, then left, and finally, right.



On each move, he can jump 1, 2, 3, 4 or 5 positions from his current position. He picks each distance exactly once. For example, he can move R by 2, L by 1, R by 5, L by 4 and R by 3, ending at $2 - 1 + 5 - 4 + 3 = 5$ positions to the right from where he started.

For your convenience, every second position is shaded.

Question

Out of the four positions marked by a letter, there is one that he cannot end up on. Which one?

- (A) Position marked A.
- (B) Position marked B.
- (C) Position marked C.
- (D) Position marked D.

Answer

(C) Position marked (C)

Explanation of Answer

Consider going right as positive and going left as negative.

Position (A) can be reached by $5 - 4 + 3 - 2 + 1 = 3$ steps to the right.

Position (B) can be reached by $1 - 5 + 2 - 3 + 4 = -1$ steps to the right, which is really one step to the left.

Position (D) can be reached by $5 - 3 + 4 - 1 + 2 = 7$ steps to the right.

Notice that he cannot reach (C) since there are 3 odd numbers and 2 even numbers, and when these are combined as additions and/or subtractions, the result will always be odd. Specifically, notice that two odd numbers added or subtracted will always be even, and three odd numbers added or subtracted will always be odd. When two even numbers are combined by addition or subtraction, they form an even number, and an even number and an odd number combined by addition or subtraction is always odd. However, (C) is at position -2 , which is unreachable.

Connections to Computer Science

This task focuses on *abstraction* and following an *algorithm*. In particular, the sequence of moves is an algorithm, in that a decision on the first step influences what the next step(s) can be, and these moves are executed in a particular order.

Abstraction is used by expressing the problem mathematically. We use positive numbers to represent moves to the right and negative numbers to represent moves to the left. In this context we can then determine feasible and infeasible solutions.

This problem is not as simple as it first appears. Imagine a slightly different problem in which a beaver is given a set of numbers and it can use them in an arbitrary order, not necessarily alternating left and right, and he wishes to end up on a particular position. Solving this problem is known to be really difficult (computer scientists call it *NP-hard*) and practically unsolvable with today's technology as soon as the number of moves becomes large.

Country of Original Author




Canada



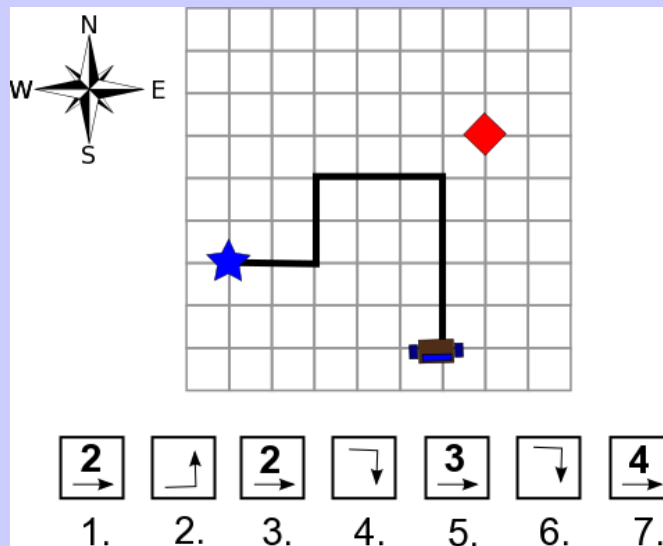
Mistakes

Story

Three kinds of buttons control a robot:

Button	Description
	robot turns left
	robot turns right
	robot moves X units in the direction it is facing

The robot starts at the blue star facing east. John presses the seven buttons shown (from left to right) to try and move the robot to the red diamond. Unfortunately, he presses two extra buttons by mistake.



Question

Which two button presses should be removed so that the robot ends at the correct location?

- (A) the 1st and the 2nd
- (B) the 1st and the 4th
- (C) the 3rd and the 4th
- (D) the 2nd and the 6th

Answer

(C) the 3rd and the 4th

Explanation of Answer

The robot needs to go vertical 3 units which can only occur when button 5 is pressed. This must happen while the robot is facing north which can only be after button 2 is pressed and before turning again. Therefore pressing button 4 must be a mistake. This then also means that pressing button 3 is a mistake because otherwise the robot moves too far north without any way of heading south later. We can check that pressing buttons 1, 2, 5, 6 and 7 (in that order) does indeed bring the robot from the blue star to the red diamond.

Connections to Computer Science

Computers are *programmed* much like the robot is controlled but with a larger and more complicated set of possible instructions. This means that even the most skilled computer programmers make mistakes. So it is important to understand how to find and correct mistakes. An error in a computer program is called a bug and the process of finding and fixing bugs is called *debugging*. Everyone has experienced the frustration of software (e.g., an app) crashing. A crash usually happens because of a bug. Unfortunately, bugs can cause much more than frustration. For example, critical software is used to administer medicine to hospital patients and to send rockets into space. Debugging and *testing* are especially important in these life-or-death situations.

Country of Original Author

France



Part C

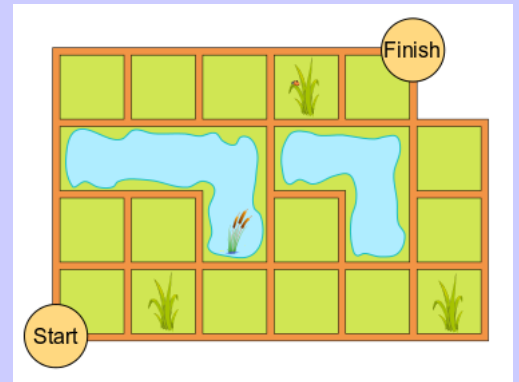
You Must Turn

Story

A king loves long travels in his coach, so he orders his coachman to never go straight when reaching a new road. That is, the coachman must turn either right or left if he comes to any intersection. This applies even for intersections with three roads.

On the image to the right you see a road map of the country. Roads are shown as red vertical or horizontal lines. All roads connecting two adjacent intersections are 1 km long.

The king needs to go from the bottom left corner, marked Start, to the top right corner, marked Finish. The coachman however, wants to get to the destination as quickly as possible.



Question

Find a shortest path for the coachmen from start to the finish that does not break the rule that the coach can never go straight. What length does it have?

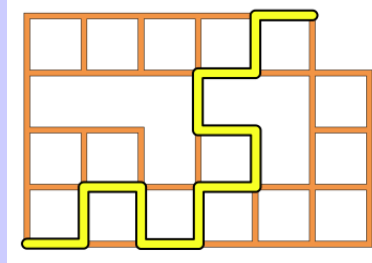
- (A) 11 km
- (B) 13 km
- (C) 15 km
- (D) It is not possible to get to the destination without breaking the rule.

Answer

(B) 13 km

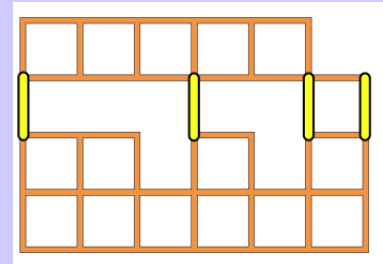
Explanation of Answer

Here is the 13 km path:

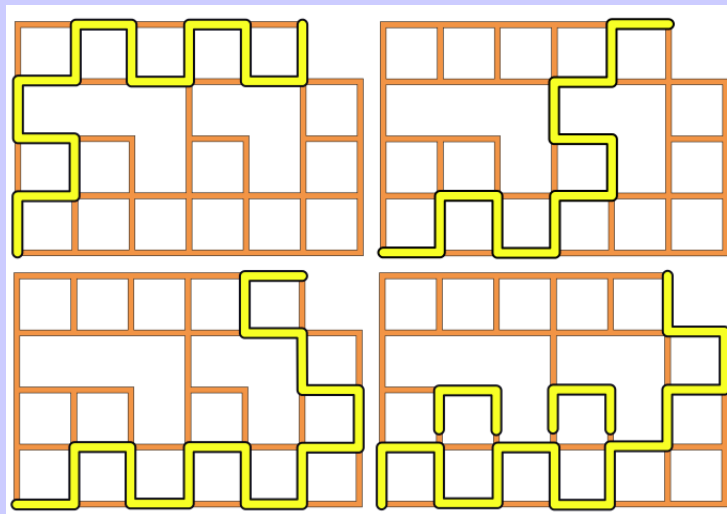


Let us now show that this path is optimal.

On the image to the right you see four marked roads. The king on his journey must use at least one of these roads. Let us now consider them one by one. It is possible to reconstruct the path starting from these roads, i.e. we continue the path first forwards, then backwards. Almost always there is only one way to continue a path; sometimes we may turn in both directions, but in these cases we do not turn to obviously non-optimal directions.



Finally we select the optimal path from four variants:



Connections to Computer Science

This is a problem about *searching* for the *optimal path*. An optimal path is one which goes between two or more points with the minimal distance or cost.

This task is a more complicated version of the simple *path search* problem, because paths must satisfy certain conditions. A real world example of such a restriction is that some intersections do not allow drivers to make left turns. Thus, finding optimal paths in a city network requires much care in order to not send drivers down one-way streets or turning left when it is illegal to do so.

Country of Original Author

Russia



Weather

Story

Beaver John plans to go to the beach tomorrow, but he will go only if there will be at least three sunny hours between 13:00 and 19:00 (which is 24-hour time for 1:00pm and 7:00pm).

He has a file containing the hourly weather forecasts, made up of 24 lines corresponding to each hour of the day, from 00:00-01:00 to 23:00-24:00; each line contains one of the words *sunny*, *cloudy*, *rainy*, or *snowy*. He can use the following commands

- **ONLY** w selects only the lines containing the word w
- **FIRST** n selects the first n lines (or all the lines if there are fewer than n lines)
- **LAST** m selects the last m lines (or all the lines if there are fewer than m lines)
- **COUNT** counts the number of lines in its input

Using `|` as a separator, John can combine these commands in sequence as he likes: the output of any command in the sequence will be the input of the following command. For example, the sequence

A | B | C

where A, B and C are commands from the list, indicates that the output of A is used as the input for B; the output of B is then used as the input for C. The input to the first command is always the content of the file of forecasts.

Question

How can John arrange the previous commands in order to decide whether or not he will go to the beach?

- (A) `FIRST 19 | LAST 6 | ONLY sunny | COUNT`
- (B) `ONLY sunny | FIRST 19 | LAST 6 | COUNT`
- (C) `FIRST 20 | LAST 7 | ONLY sunny | COUNT`
- (D) `LAST 20 | FIRST 6 | ONLY sunny | COUNT`

Answer

(A) FIRST 19 | LAST 6 | ONLY sunny | COUNT

Explanation of Answer

Notice that (A) will take the first 19 hours (ending at 19:00); we can then take the last 6 of these (from 13:00 to 19:00), keep only those that are sunny, and then produce a count of those. If this number is at least 3, Beaver John will go to the beach; otherwise, he will not go to the beach.

Notice that answer (B) will sometimes produce a count of at least 3 even if there is no sun between 13:00 and 19:00. For example, if it is sunny in every other time period except 13:00 to 19:00 (that is, for 18 hours), answer (B) will produce a value of 6.

For answer (C), the count is incorrect. This sequence of commands will go up to 20:00 (not 19:00), and thus could give an incorrect answer (if there is sun from 19:00-20:00).

For answer (D), similar to answer (B), we may incorrectly filter out incorrect times: the last 20 hours begin at 04:00 and thus the window of hours examined in this case is from 04:00 to 10:00, rather than 13:00 to 19:00.

Connections to Computer Science

Several computer science activities can be efficiently carried out through a set of processing steps, each filtering a part of its input, executed in sequence. Examples of such an approach include:

- *compilation*: converting a source code program into machine-readable binary;
- *data conversion*: converting an HTML file into a PDF file;
- *data compression*: converting a plain-text file into a smaller zip-file format.

This approach allows us to focus on several simple tasks (the ones carried out in a specific processing step), rather than on the (more complex) general problem.

The operating system Unix (and its Linux descendants) allow this sort of “piping” and “filtering” of commands.

Country of Original Author












Italy



Fireworks

Story

Two beavers live in lodges separated by a large forest. They decide to send messages to each other by shooting fireworks into the sky above the trees. Each message is a sequence of words, but the beavers only know five words. They shoot two types of fireworks one after the other according to the following code:

Word	Code
log	 
tree	  
rock	  
den	 
food	

For example, to send the (strange) message “food, log, food”, a beaver would shoot:



Question

How many different meanings does the following sequence of fireworks have?



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

Answer

(D) 4

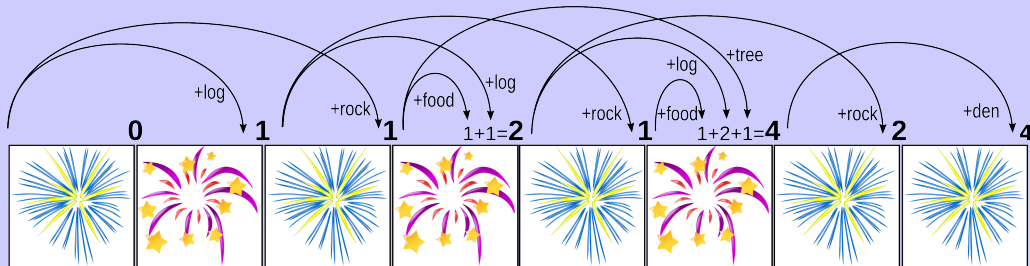
Explanation of Answer

The message could mean any of the following (notice that the last word is always “den”, since the last two fireworks can only be used in the word “den”):

- log, rock, food, den
- log, log, log, den
- rock, tree, den
- rock, food, log, den

To convince yourself that there are no more possibilities, you can systematically count them:

- Start with the first firework. It is not a message, so count it as zero.
- The first two fireworks can only mean log. Count the first two fireworks as one message.
- Looking at the third firework, it cannot be a new word on its own, but it can form a word (rock), and thus it counts as one message.
- The fourth firework is more interesting. It can either add the word log to the first two fireworks, or food to the first three fireworks, as shown by the arrows below. So we sum the two numbers at the 2nd and 3rd firework and write it to the 4th ($1+1=2$).
- We proceed applying the same idea to each firework to the right. We look one, two and three fireworks back. If those shorter messages can be extended with a correct word, we mark this fact with an arrow. Then we just sum the numbers “brought” by the arrows to the currently examined firework.
- At the last firework we will have the number of all possible meanings.



Connections to Computer Science

All digital information is represented using *binary*. That is, it consists of only the bits 0 and 1. Only longer combinations of 0 and 1 (“words” in this task) allow the use of more than two different meanings. But we also want to avoid ambiguity in our messages.

Most standard codes use the same number of bits per word, so there is only one meaning to each message. But if some word is used very often and some rarely, such codes generate needlessly long messages.

It is then useful to have shorter codes for frequent words (like “food”) and longer codes for less frequent words (like “rock”). Of course you have to be smarter than the beavers in our task. If you generate a *prefix code*, the messages will only have one meaning. This trick (shortening frequent data chunks without introducing ambiguity) is used in *data compression*.

The systematic approach when we build our solution step-by-step using the previous steps is called *dynamic programming*. It makes the process much easier—just imagine trying to find all possible meanings of the message right away!

Country of Original Author

Canada

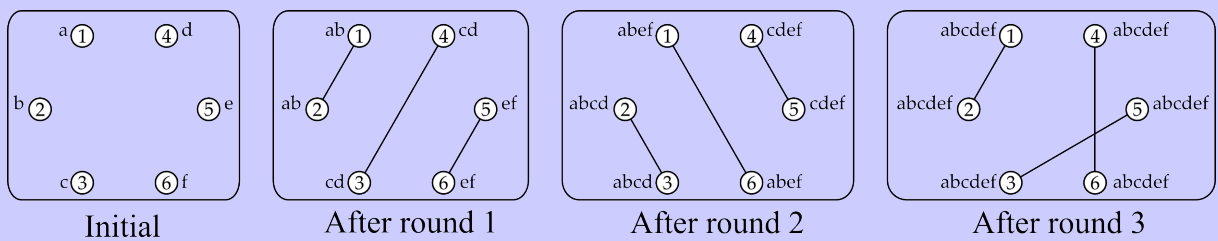


Spies

Story

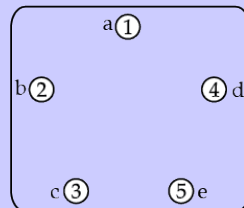
Every Friday, six spies share all the information they have gathered that week. A spy can never be seen with more than one other spy at the same time. They have to conduct several rounds of meetings where they meet up in pairs and share all information they have at that point.

The group of 6 spies needs only three rounds to distribute all information. Before the meetings, each spy holds a single piece of information. (spy 1 knows “a”, spy 2 knows “b”, etc.). In the first round, spies 1 and 2 meet and share information so now both know “ab”. The diagram shows the initial information as well as the three rounds of meetings, with lines indicating which spies meet in each round. It also shows which pieces of information they all have. After three rounds all information has been distributed.



Question

After an international incident one spy has stopped attending the meetings. What is the minimum number of rounds needed for the five remaining spies to share all information?



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

Answer

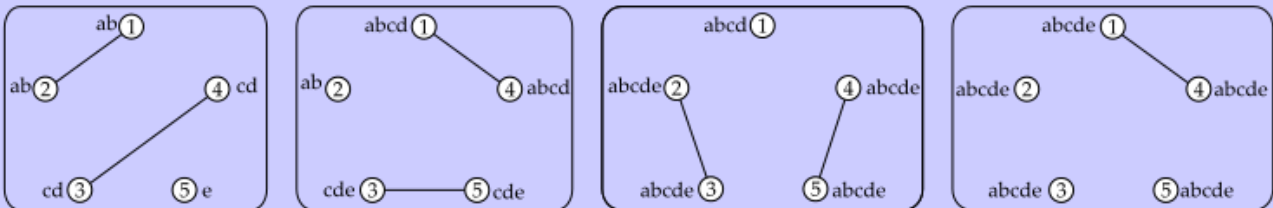
(C) 4

Explanation of Answer

This answer is probably unexpected: the obvious answer is three (or less?) since we have one spy less. This answer is even more strange if we consider that if there were only four spies they would quite obviously share all the information in two rounds.

However, unsuccessful attempts at solving the task soon show us the root of the problem: since the number of spies is odd, one of them is “inactive” in every round. Say that spy number 5 does not participate in the first round, but he/she participates in the second round. Thus after the second round, only two spies will know his/her piece of information (e). In the third round, these two spies will meet two other spies, so after three rounds (only) four spies will know e. The fourth round is needed to spread this information to the fifth spy.

Therefore, we have proved that at least four rounds are needed. To show that they are enough, we construct a schema with four rounds.



Connections to Computer Science

When computers share information, the sharing usually happens in pairs. A related problem is how to share information along a whole network in as short a time as possible. Thus, computer scientists need to solve problems similar to this task: how to share information as *efficiently* as possible.

This task is also known as the *gossip problem*. It is a worthwhile exercise to try to solve it for different numbers of spies and you might discover an interesting, general rule.

The solution of this problem was first solved and the general rule described in 1975. This and many similar problems can be applied to different areas in computer science such as *data sharing*, *communication networks* and *cryptology*.

Country of Original Author

Slovenia



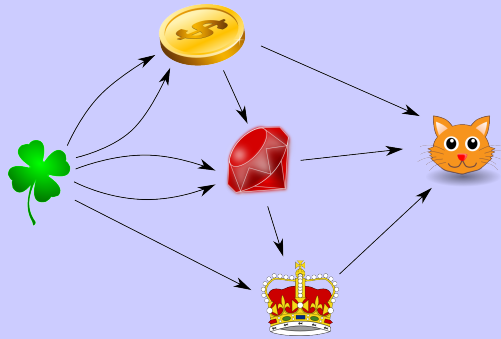
Beaver the Alchemist

Story

Beaver the Alchemist can convert objects into other objects. He can convert:

- two clovers into a coin;
- a coin and two clovers into a ruby;
- a ruby and a clover into a crown;
- a coin, a ruby, and a crown into a kitten.

After objects have been converted to another object, they disappear.



Initially Beaver the Alchemist has lots of clovers, but no coins, rubies, crowns or kittens.

Question

How many clovers does Beaver the Alchemist need to create one kitten?

- (A) 5
- (B) 10
- (C) 11
- (D) 12

Answer

(C) 11

Explanation of Answer

We can see the conversion as follows:

coin = 2 clovers
ruby = 2 clovers + 1 coin = 4 clovers
crown = 1 ruby + 1 clover = 4 clovers + 1 clover = 5 clovers
kitten = 1 coin + 1 ruby + 1 crown = 2 clovers + 4 clovers + 5 clovers = 11 clovers

Connections to Computer Science

We can think of the conversion as part of a (*context-free*) *grammar*. We can write the rules in the following way, where C is coins, R is rubies, O is crowns, K is kittens and L is clovers:

$$\begin{aligned}C &\rightarrow LL \\R &\rightarrow LLC \\O &\rightarrow RL \\K &\rightarrow CRO\end{aligned}$$

Here we can think of the \rightarrow as meaning “requires”: for instance, to make a coin, we require two clovers. We start with K and form a *derivation* of all the needed clovers (L). A derivation applies the rules in a certain order: to differentiate a rule from a derivation step, we use the symbol \Rightarrow to apply a rule:

$$\begin{aligned}K &\Rightarrow CRO \\&\Rightarrow LLRO \\&\Rightarrow LLLLCO \\&\Rightarrow LLLLLLO \\&\Rightarrow LLLLLLRL \\&\Rightarrow LLLLLLLLCL \\&\Rightarrow LLLLLLLLLLL\end{aligned}$$

Context-free grammars are used for language-processing (both *natural language* and *formal languages*). When we *derive* words, as we have done above, we call this *parsing*.

Country of Original Author

Russia

