



2015
*Beaver
Computing
Challenge
(Grade 7 & 8)*

*Questions,
Answers,
Explanations,
and
Connections*

Part A

Favourite Numbers

Story

Billy Beaver writes down his favourite numbers, from smallest to largest when read from left to right.

Question

Which of the following orderings of numbers is the one that Billy Beaver wrote down?

- (A) 2 3 4 5 10 31 29
- (B) 5123 5148 5171 5149 5189
- (C) 3 10 19 24 99 101
- (D) 1 100 1000 100000 10000

Answer

(C) 3 10 19 24 99 101

Explanation of Answer

The **bolded** element in the list that is smaller than elements earlier in the list:

(A) 2 3 4 5 10 31 **29**

(B) 5123 5148 5171 **5149** 5189

(D) 1 100 1000 100000 **10000**

The list in 3 10 19 24 99 101 has no such element.

Connections to Computer Science

Following specifications exactly is part of understanding how to communicate and understanding how computers work. Computers require very specific and precise directions.

Country of Original Author

Canada



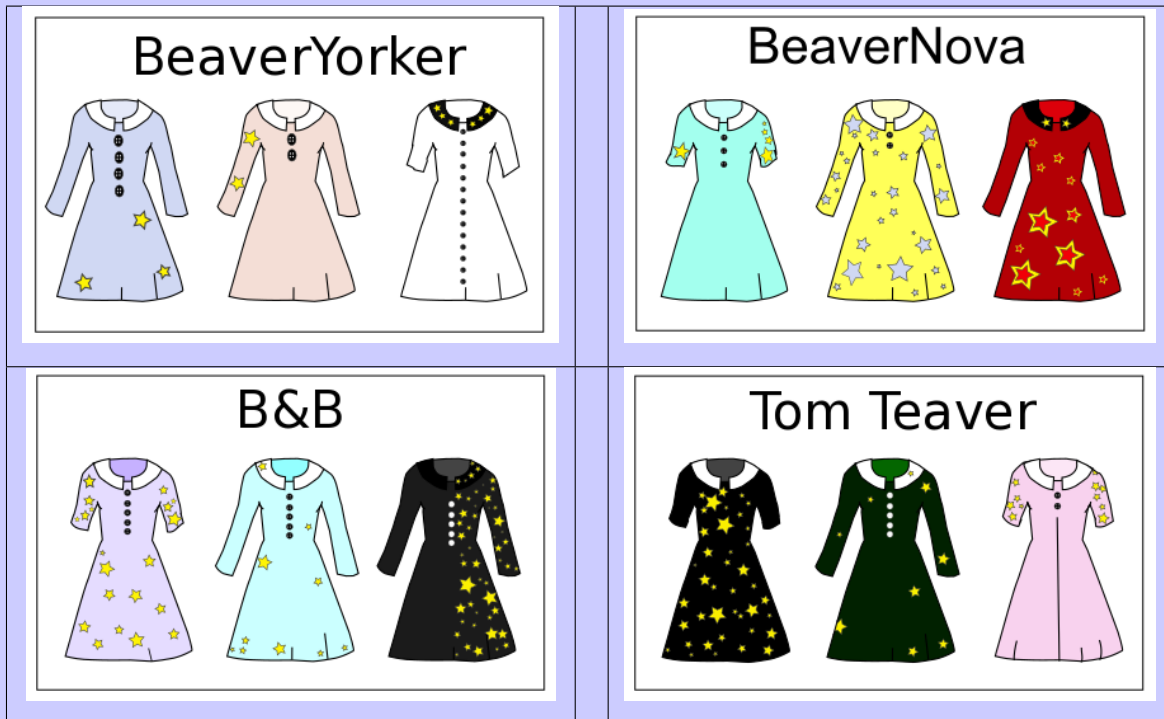
Dream Dress

Story

Kate wants to buy her dream dress. It must

- have short sleeves, and
- have more than 3 buttons, and
- have stars on its sleeves.

Four shops sell only the dresses shown:



Question

Which of these shops sells Kate's dream dress?

- (A) BeaverYorker
- (B) Beaver Nova
- (C) B & B
- (D) Tom Teaver

Answer

(C) B & B

Explanation of Answer

To solve this task, we must simultaneously satisfy three requirements. This can be done by discarding dresses that do not meet any one of the requirements. After doing this, we can see that the dress on the bottom left sold by B & B is Kate's dream dress.

The other answers are incorrect because

- The only dress sold by BeaverYorker with stars on the sleeves has long sleeves;
- Beaver Nova does not sell any dresses with more than three buttons;
- The only dress sold by Tom Teaver that has more than three buttons also has long sleeves.

Connections to Computer Science

The task involves statements (conditions/requirements) that must be evaluated (determined to be true or false) for a set of objects (dresses). Conditions and their evaluation is an important part of *programming* and *algorithmic thinking*.

Conditions can be simple statements. However, more complex statements can be formed using *logical operators* such as AND, OR, NOT. This task uses the AND operator forming a *conjunction*.

Country of Original Author

Slovakia

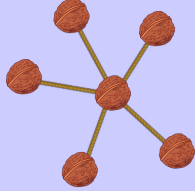
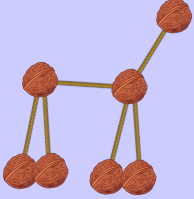
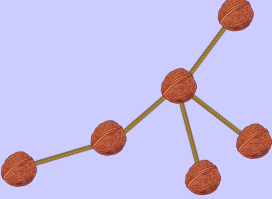
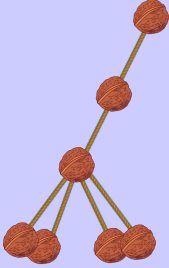


Chestnut Animals

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):

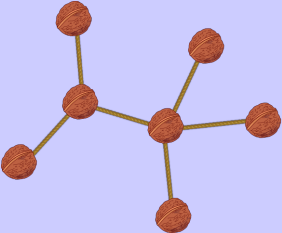


Starfish	Dog	Sea lion	Giraffe
			

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

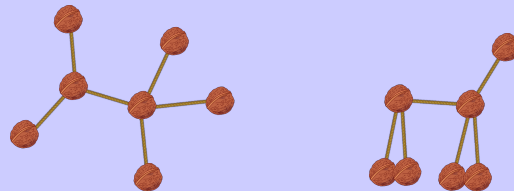
Explanation of Answer

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



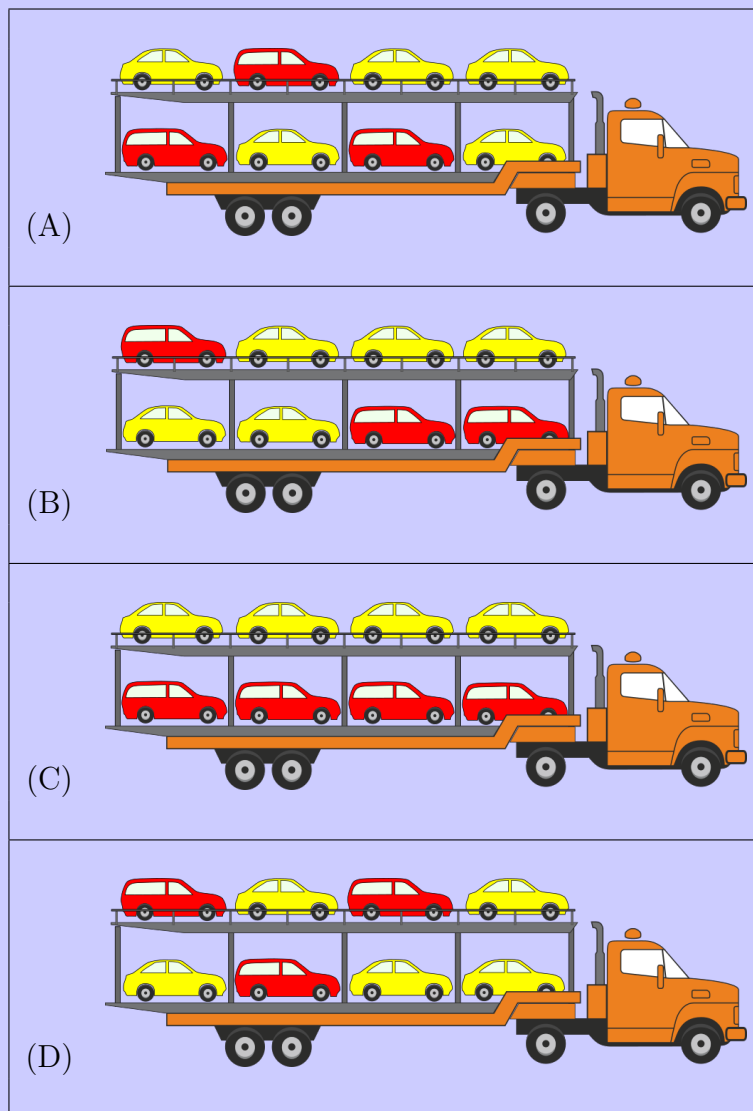
Car Transportation

Story

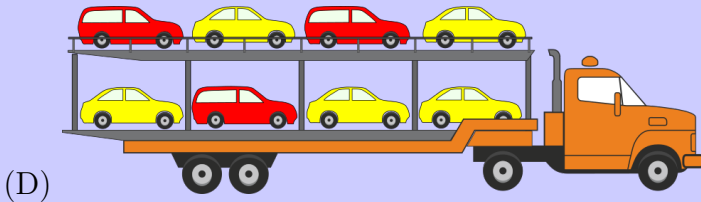
A new red car comes from a manufacturing line every 7 minutes. A new yellow car comes from another manufacturing line every 5 minutes. Both manufacturing lines start working at the same time. A driver parks the cars on the back of a large transport truck in the order the cars leave their respective manufacturing lines. The top floor of the transport truck is loaded first.

Question

What will the large transport truck look like after loading?



Answer

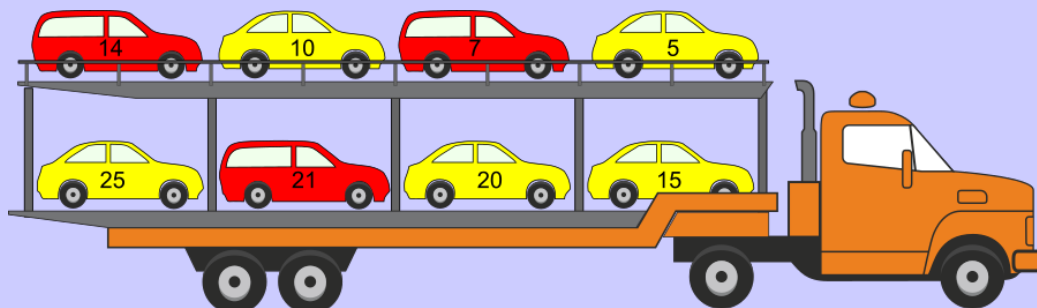


Explanation of Answer

Consider the ordering of the cars as they leave both manufacturing lines. We mark the time that they are produced on the side of each car.



Then, as we load up the cars on the top level first and bottom level second, they must be in the numerical order, as shown below:



Connections to Computer Science

Most industries, including car manufacturing, are highly *automated*, and this automation relies on computers to control and coordinate production.

As such, the production needs to be carefully planned and synchronized so that various demands (availability of transport trucks, the need for particularly designed items to go on a particular truck, etc.) can be managed.

The need to understand, create, manage and improve these automated systems is a real-world example of computer science being applied.

Country of Original Author

Czech Republic



QB-Code

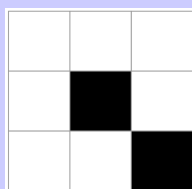
Story

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

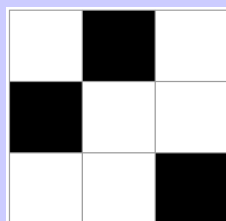
To encode a number, the beavers darken some squares. The number encoded is the sum of the values of the dark squares.

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



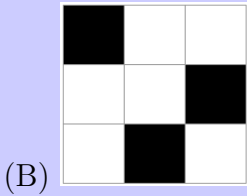
Question

Which of the following encodes the largest number?



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

Answer



Explanation of Answer

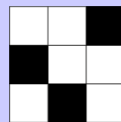
The answer can be obtained without doing any complicated calculation. The square in the top-left corner has the highest value (256). Note that the sum of all the other squares (i.e., $128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$) is less than 256. Hence, the highest possible number encoded is the QB-Code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



Part B

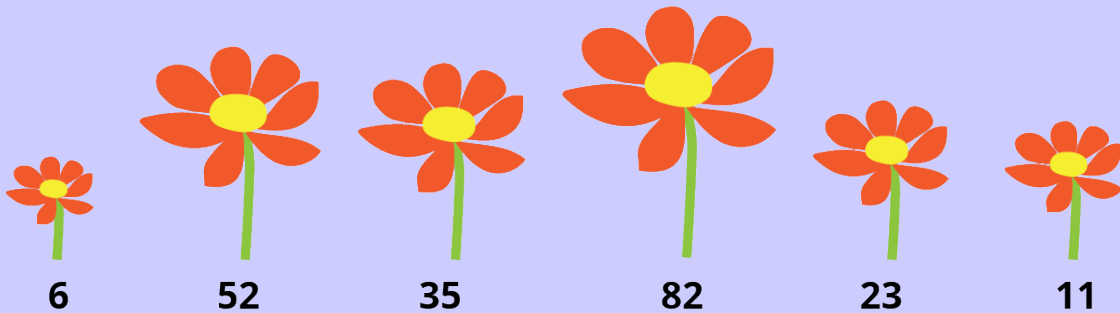
Collecting Pollen

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



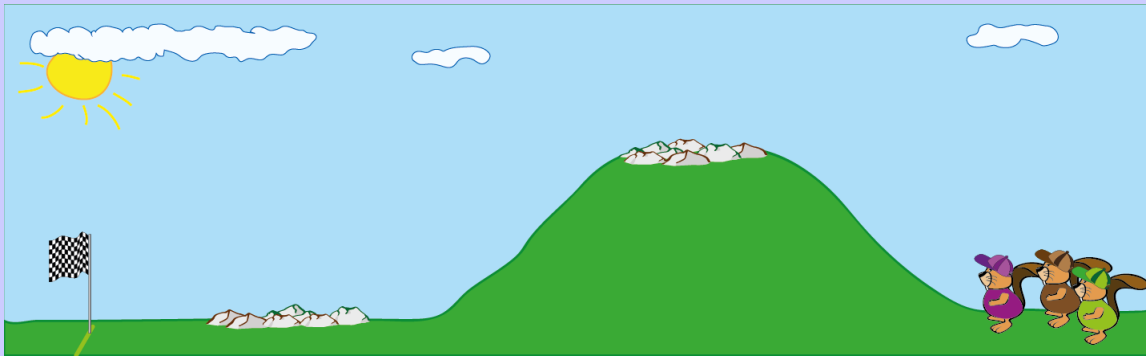
Cross-Country

Story

Three competitive runners will compete in a cross-country race.

When running uphill, Mr. Brown will overtake one beaver.	
When running downhill, Mrs. Pink will overtake one beaver.	
When running over rocks, Mrs. Green will overtake one beaver.	

The terrain is as shown in the picture: uphill, followed by some rocks, downhill and then again some rocks. Mrs. Pink starts in the first position, followed next by Mr. Brown and finally by Mrs. Green.



Question

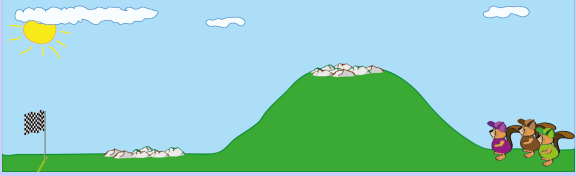
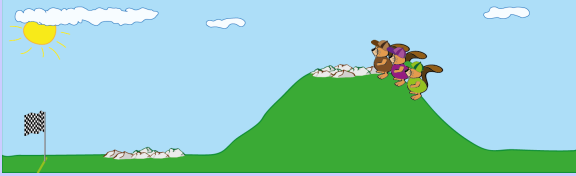

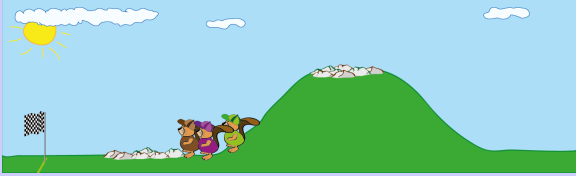
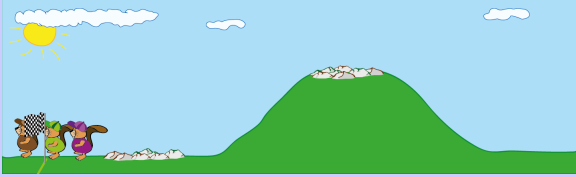
In which order will they finish the race?

- (A) Mrs. Pink, Mr. Brown, Mrs. Green
- (B) Mr. Brown, Mrs. Green, Mrs. Pink
- (C) Mrs. Green, Mrs. Pink, Mr. Brown
- (D) Mr. Brown, Mrs. Pink, Mrs. Green

Answer

(B) Mr. Brown, Mrs. Green, Mrs. Pink

Explanation of Answer

<p>Start</p>	<p>1st: Pink 2nd: Brown 3rd: Green</p>	
<p>Uphill Brown overtakes Pink</p>	<p>1st: Brown 2nd: Pink 3rd: Green</p>	
<p>Rocks Green overtakes Pink</p>	<p>1st: Brown 2nd: Green 3rd: Pink</p>	
<p>Downhill Pink overtakes Green</p>	<p>1st: Brown 2nd: Pink 3rd: Green</p>	
<p>Rocks Green overtakes Pink</p>	<p>Final Result: 1st: Brown 2nd: Green 3rd: Pink</p>	

Connections to Computer Science

Programmers must carefully understand how their programs execute. This is especially true when the programs do not work well: in this case, programmers carefully go through and check the effect of each line of the program.

This task is similar to *tracing* through the *execution* of a *program*. We are given data – the initial sequence of runners. There are four steps in the “program”: uphill, rocks, downhill, rocks. We must observe and record the effects of each step on the sequence and thus discover the “output” of the program: that is, the order of runners at the end of the sequence.

Country of Original Author




South Africa



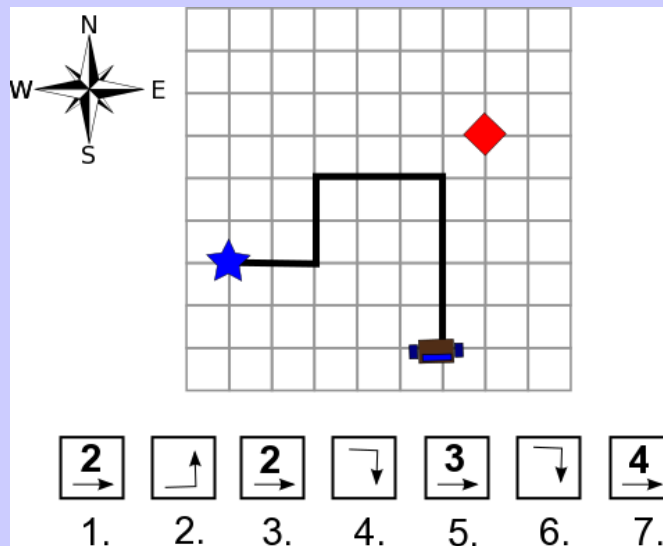
Mistakes

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

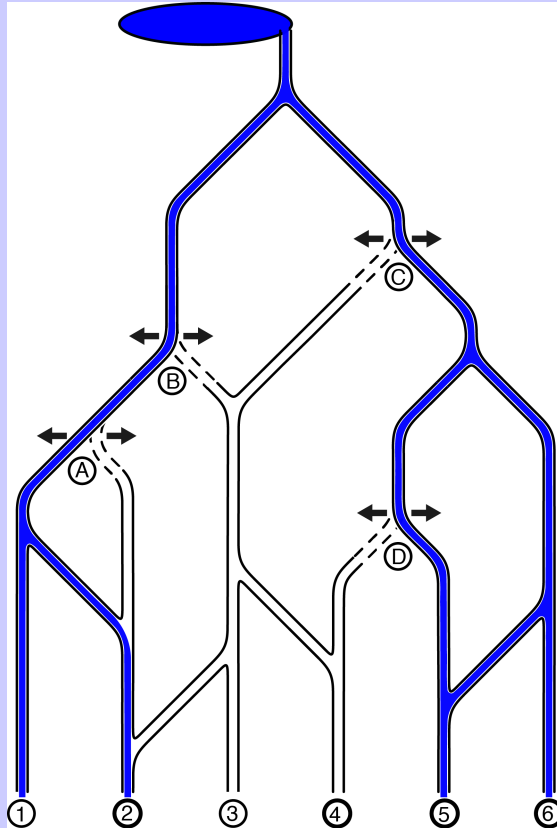


Irrigation System

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: → B: ← C: → D: ←

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



Dogs versus Beavers

Story

Beavers and dogs compete. The nine participants scored the following points: 1, 2, 2, 3, 4, 5, 5, 6, 7.



We know that no dog scored more than any beaver, but one dog had the same score as a beaver and two dogs also had the same score.

Question

How many dogs took part in the competition?

- (A) 2
- (B) 3
- (C) 6
- (D) 7

Answer

(C) 6

Explanation of Answer

If no dog scored more than any beaver, we can order the animals in a row so that a separator can be used to separate the dogs and beavers. For example: (dogs in front) 1, 2, 2, 3, | 4, 5, 5, 6, 7 (beavers at end), where | means the separator between dogs and beavers.

If two dogs scored 5, then since all dogs scored less than beavers, then two dogs must have also scored 2. This, however, does not allow for the fact that a dog and a beaver tied. Hence, the two 5s must be a dog and a beaver, which means that the separator between the dogs and the beavers must be between the two 5's:

(dogs) 1, 2, 2, 3, 4, 5, | 5, 6, 7 (beavers).

Notice that if the separator between dogs and beavers was between the 2's, two dogs must have scored 5 and 5 and then they will be better than the beaver that scored 2. This is in contrary to the task statement, so it cannot be possible.

Therefore, 6 dogs participated in the competition.

Connections to Computer Science

When working with data, some *organization* is necessary. This task requires us to understand how the data is ordered and how the ordering rules are used.

This problem also requires some *logic* in a solution of the task by narrowing down the possibilities to a small set of choices and ruling out possibilities due to a *contradiction*.

Country of Original Author

Czech Republic

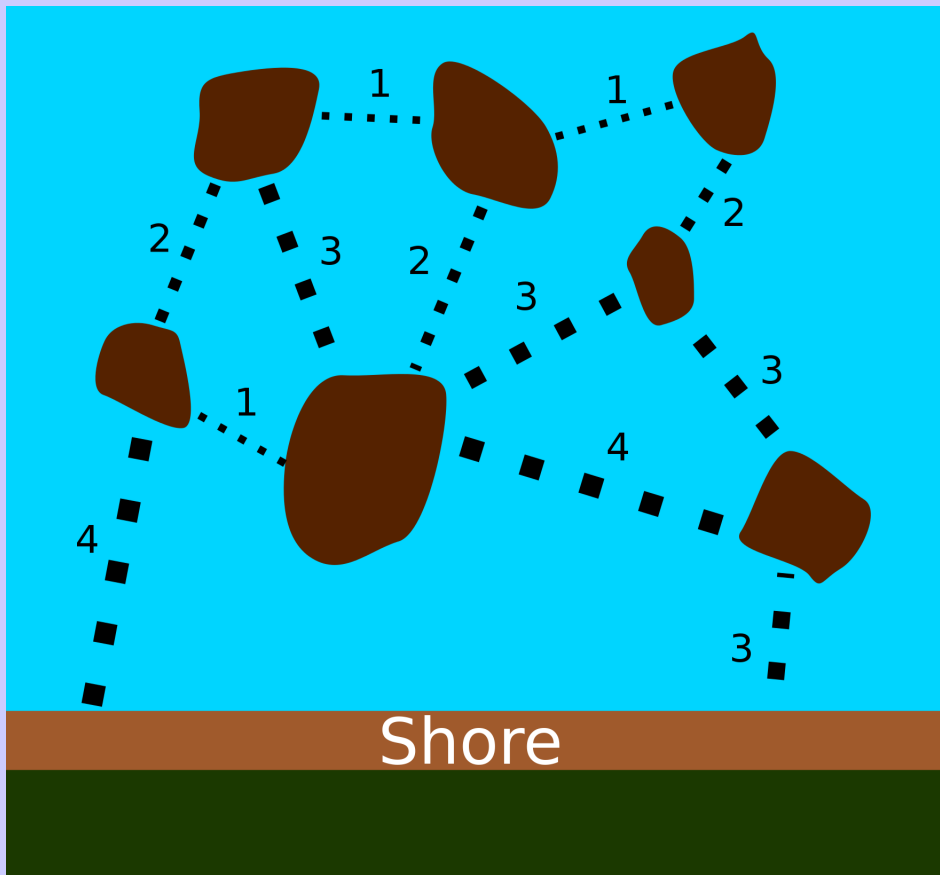


Part C

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

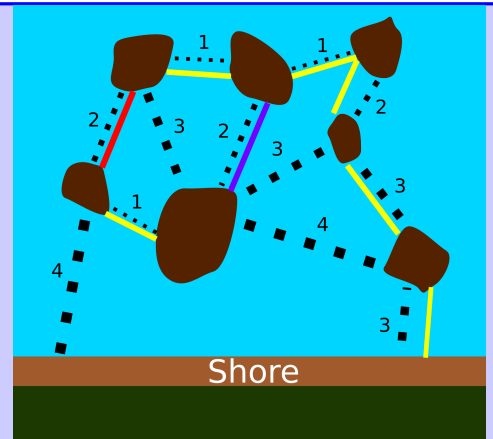
To use as few trees as possible, the yellow bridges and either the red bridge or purple bridge shown to the right should be built. Building these seven bridges requires a total of $1 + 1 + 1 + 2 + 2 + 3 + 3 = 13$ trees.

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

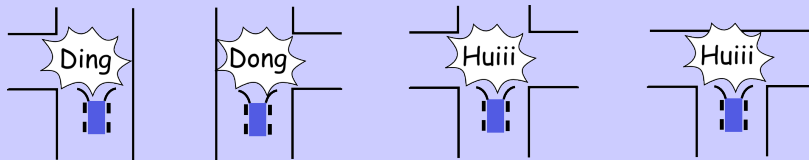
Switzerland



Robotic Car

Story

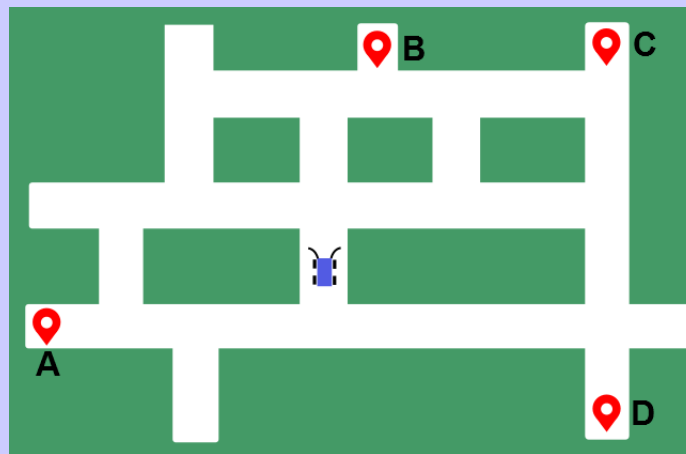
Beavers have developed a robotic car. It has sensors that detect intersections. It produces the sounds shown below, when it is possible to turn left, right or both directions. The robotic car can go straight through an intersection (when possible), turn right (when possible) or turn left (when possible). 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?

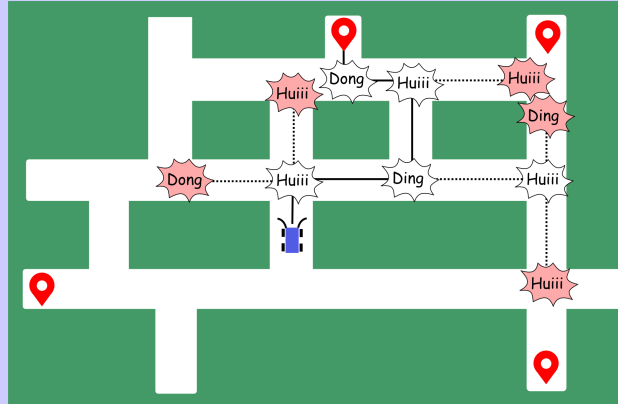
- (A) Location A
- (B) Location B
- (C) Location C
- (D) Location D

Answer

(B) Location B

Explanation of Answer

The image below shows the one and only route that the robotic car must take while producing the given sequence of sounds:



Notice that we try each possible direction at each intersection, and if the sound produced is not the correct/expected one, we *backtrack* to the previous intersection and try a different path.

Connections to Computer Science

The robotic car in this task is a simple version of an *autonomous* car. Autonomous cars sense their surroundings with radar, GPS or computer vision. They identify appropriate navigation paths as well as obstacles and relevant signage. Major companies and research organizations have developed working prototypes of autonomous cars. The development of control systems for automated vehicles is an important field in informatics.

Moreover, the task algorithmically requires *exploration of the search space*. The particular method used here is *backtracking*. It starts from the beginning and when it can make a choice (the first intersection) it picks one of the possible choices (left turn in the example). At the next choice, it makes another decision if it can. In case it cannot make a decision (in our case the car reached a possible right turn, but it expected a left turn), it backtracks to the place where it made the previous decision (which was incorrect) and makes a new decision (in our case go straight). This process is repeated until the goal is reached or we run out of possible decisions.

In our case the second decision, to go straight was also incorrect. This can be seen in the diagram. We then backtrack to where the decision was made and use a right turn, the third possibility.

Country of Original Author












Germany



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

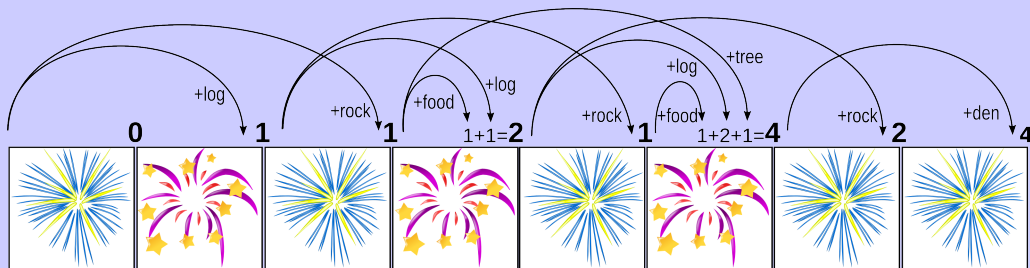
Explanation of Answer

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



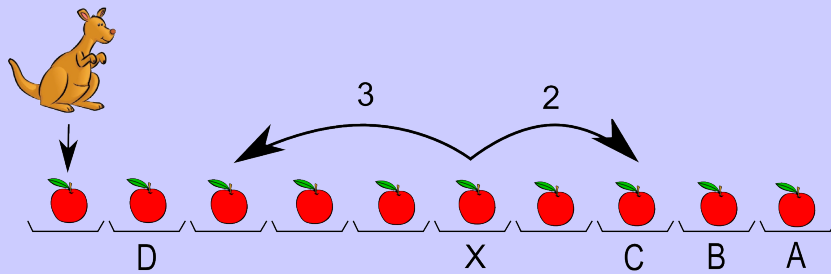
Jumping Kangaroo

Story

There are 10 plates in a row. There is one apple on each plate.

Kangaroo Thomas loves to jump. First, he jumps onto the leftmost plate. On each single jump after this, he either jumps forward two plates, or backwards three plates. Thomas only jumps onto plates with an apple. If he jumps onto a plate, he collects the apple from it, and therefore, can only jump on each plate at most once.

An example of the two possible jumps from one plate, labelled X, is shown with arrows in the picture below:



Question

If Thomas collects all 10 apples, which apple does he collect last?

- (A) The rightmost apple
- (B) The second apple from the right
- (C) The third apple from the right
- (D) The second apple from the left

Answer

(B) The second apple from the right

Explanation of Answer

We number the plates from left to right from 1 to 10. Then Thomas can collect all ten apples in the order 1, 3, 5, 2, 4, 6, 8, 10, 7, 9.

This is the only sequence of jumps that allows Thomas to collect all the apples. Why? To begin, Thomas must jump on plates 1, 3 and then 5 because otherwise he jumps to the left of the first plate. Next, he must jump to plate 2 because he can only get to plate 2 from plate 5 and he will not return to plate 5 later. The same kind of reasoning can be used to see that all the remaining jumps are also determined uniquely.

Connections to Computer Science

One way to solve this problem is to consider all possible sequences of plates and look for one that consists of only valid jumps. Each possible sequence is called a *permutation* and there are many of them: for 10 plates, there are $10! = 10 \cdot 9 \cdot 8 \cdots 1 = 3628800$. So this approach, which is called *brute-force* or *exhaustive search*, takes a lot of time.

Another approach is to build a permutation one plate at a time. Once you figure out that a permutation or the start of a permutation is not valid (such as determining that the second plate cannot be anything but plate 3), you can remove the last plate(s) and continue building new permutations. This is called *backtracking* and if you can rule out many permutations early in your search, you can find a valid permutation much faster. This short-cut is called *pruning*.

One way to look at this problem is to view it as a *graph*. The plates are *vertices* and we join two plates by an *edge* if Thomas can jump between them. The task involves finding a path moving along edges that visits every vertex exactly once. This is called a *Hamiltonian path*. In general, it is very hard to find such a path. However, in this case, the graph is small and has special properties.

The general problem of finding a Hamiltonian path is known to be *NP-complete* which means that it belongs to a collection of very important problems for which we do not have efficient solutions. Interestingly, we know that if somebody finds an efficient solution to one of these important problems, then we instantly have a way to solve every one of these important problems efficiently.

Country of Original Author

Russia



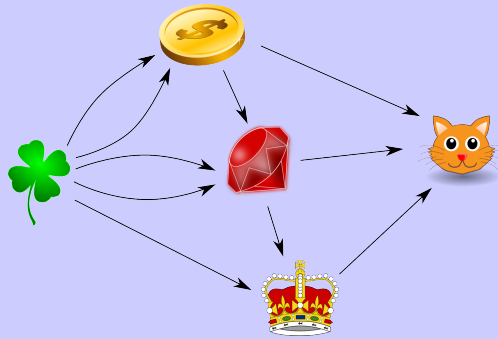
Beaver the Alchemist

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

