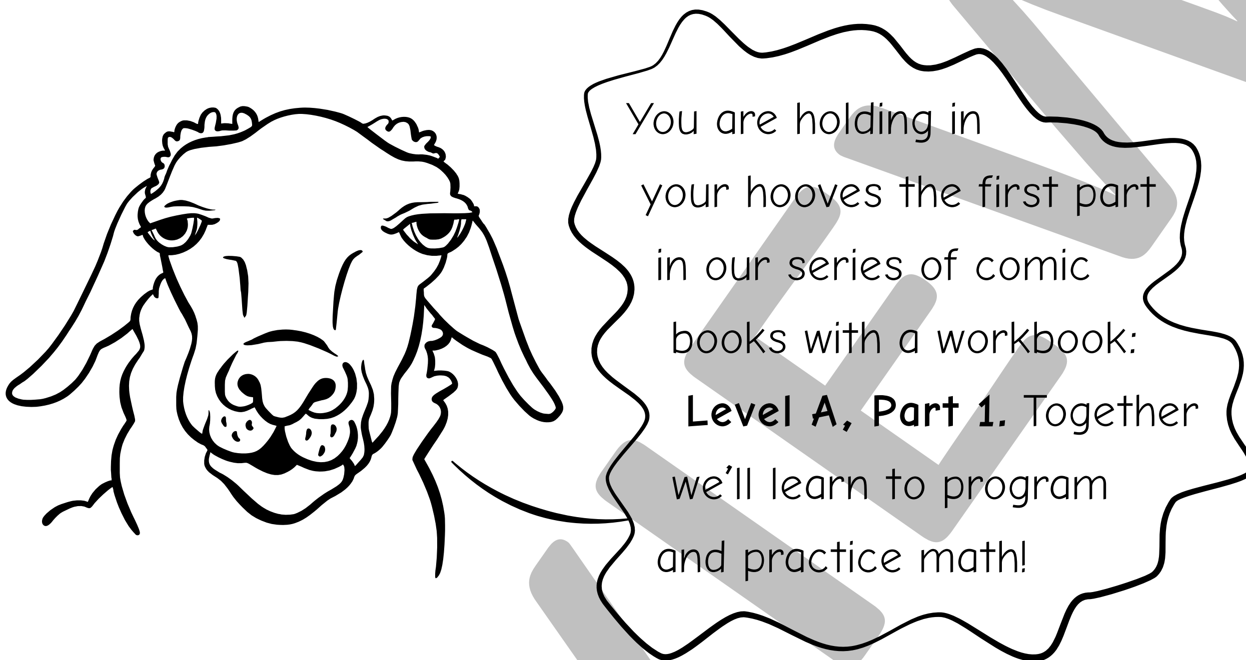
Programming and Mathematics

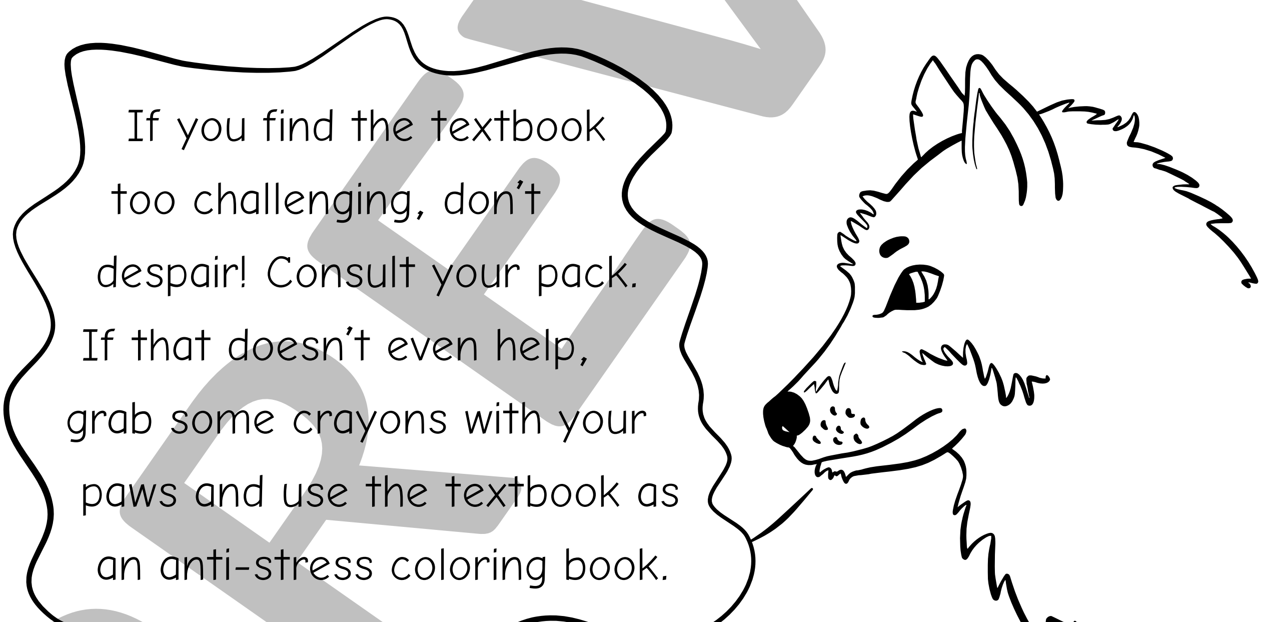
A1

Vilém Vychodil

PM Books s.r.o., 2023

Programming and Mathematics A1

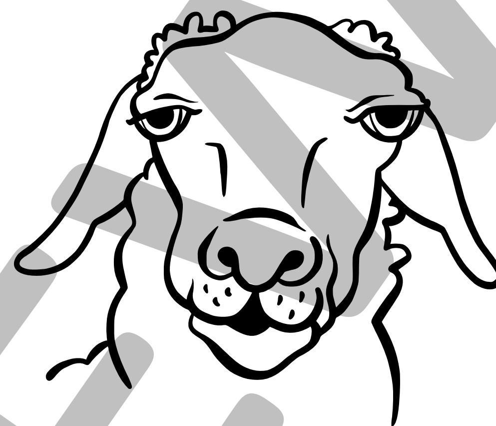




Vilém Vychodil

PM Books s.r.o., 2023

Programming and Mathematics A1



Author: Vilém Vychodil

Original title: Programování a matematika A1

Original publisher: PM Books s.r.o.

Translation: Nicola Catherine Fořtová, Vilém Vychodil

Illustrations: Monika Klimková

Typesetting and graphic design: Vilém Vychodil

Cover design: Vilém Vychodil

Published by: PM Books s.r.o., Dittrichova 346/4, Nové Město, 120 00 Praha 2, CZ (https://prog-mat.com)

Print: Lulu Press, Inc., 627 Davis Drive Suite 300, Morrisville, NC 27560, USA (https://lulu.com)

The chapter *Expressions and Abaku* follows the Abaku® method of teaching developed by AL.21 s.r.o.
Detailed information about the method, the board game and its online version can be found at https://abaku.cz.

The author of the board game Animal Husbandry (Hodowla zwierzątek), mentioned on page 68, is Karol Borsuk.
Its modern version Superfarmer is published by Granna Sp. z o.o. (https://granna.pl).

The author of the game Mastermind, mentioned on page 78, is Mordecai Meirowitz.
MASTERMIND is a registered trademark owned by Hasbro, Inc. (https://hasbro.com).

1st edition

ISBN 978-80-908730-2-5

Dear reader,

In the pages that follow you can immerse yourself in the adventures of Ramsy the ram and Barky the wolf. They are good old chaps, who decided to become programmers. Actually, they didn't really have a choice, but they soon found out that it was a lot of fun. We would love for you to share their enthusiasm and enjoy discovering new things while reading the comic.

If you've already completed the first grade, you can start reading with confidence. It's quite possible that you won't understand everything the first time round, and that's perfectly fine. Our heroes also have to learn for themselves, and they sometimes find it hard. They have their guide, Mr. Lambda, who's there to nudge them in the right direction. If you can also find someone older to go through the text with you, it may be super helpful.

If you're a more mature reader, we have no doubt that you'll be able to go through the textbook independently. Some introductory chapters may seem to be too simple, but don't be fooled, they hide important messages and interesting examples. And why are we bringing mathematics into programming? Programming, as a skill and art, is related to mathematics on many levels, but above all in the way it forces a person to think and develop their abstraction ability. That's exactly what we're after.

Author

Dear parents, mentors, teachers, programmers, hackers, geeks and nerds of all kinds!

Thank you for your support, because it is most likely you who have made sure that your loved ones, eager for education, are in possession of our non-conformist textbook. Let us give you some tips on how to work with the book and additional materials.

The basic procedure for working with this book is to read through the chapter first. Every chapter usually introduces new concepts. At the end of each chapter there are a series of problems which need to be solved. Many of the problems are designed to be verified on a computer, which we highly recommend. We don't expect the reader to become an active programmer from the outset. Using the computer to verify solutions to problems by writing expressions of a particular programming language is, from our point of view, the simplest form of programming. Although we consider working with text to be key, we will gradually create interactive versions of the textbook examples, which will be available on our website (https://prog-mat.com).

Some of the tasks go beyond the relevant chapters and are intended to stimulate curiosity. It's not necessary for the reader to solve all the tasks. Likewise, it's OK if they make a mistake. The reader will eventually figure out most of the mistakes on their own as their horizon gradually expands, or whilst interacting with the computer itself.

The following typographic conventions are used in the text:

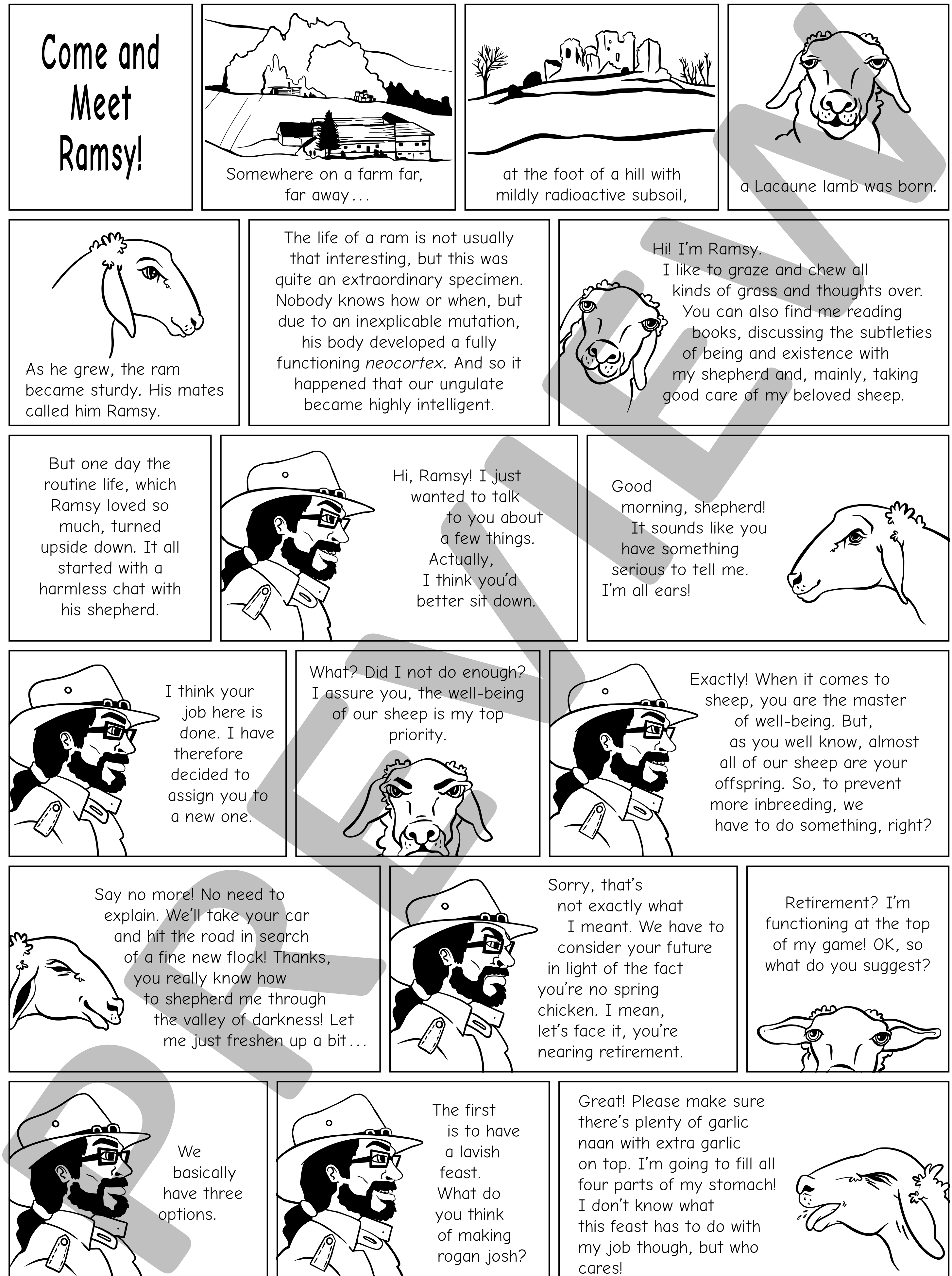
new concepts are written in **bold**,

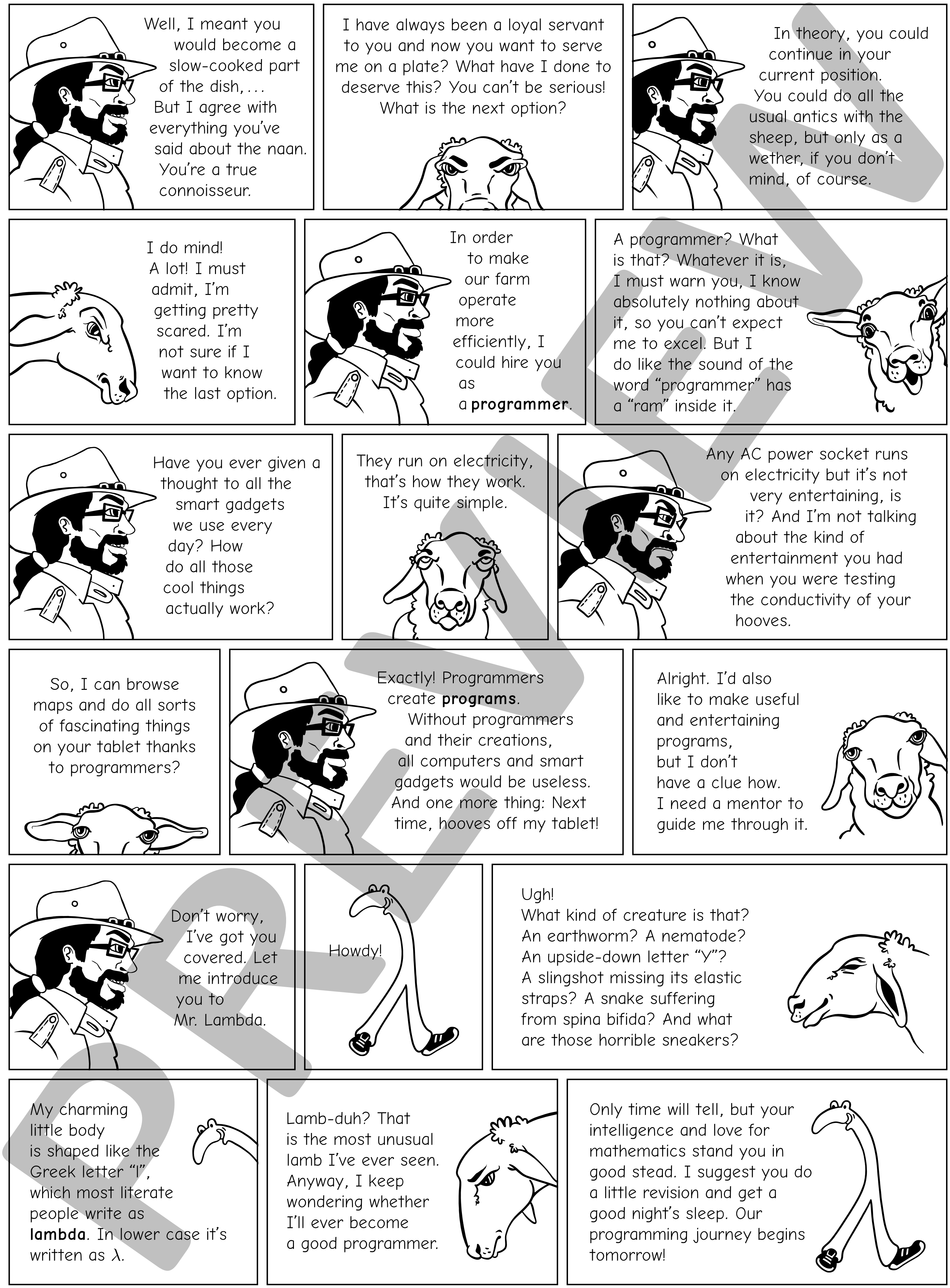
highlighted passages are *slanted*,

mathematical expressions such as $8 + 5 = 13$ are written in the Computer Modern font,

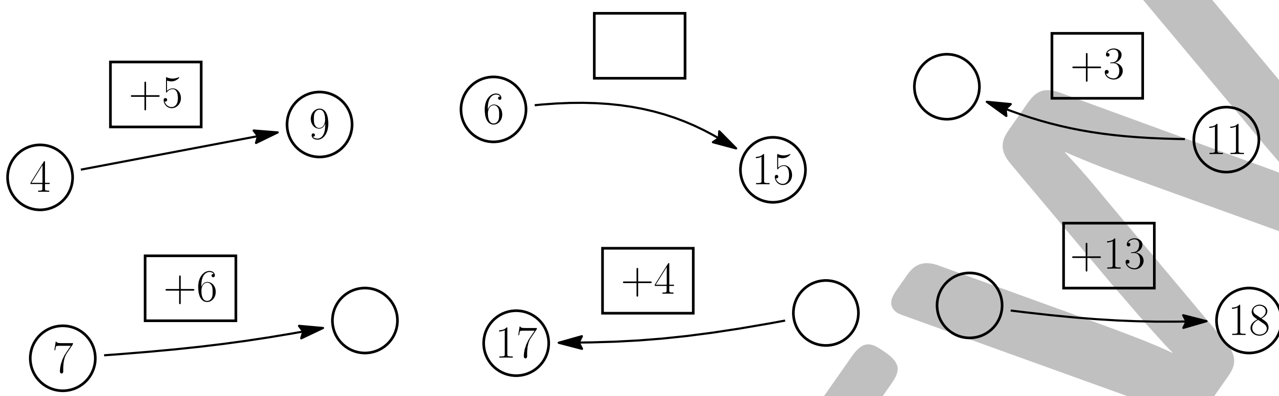
programming language expressions such as `(= (+ 8 5) 13)` are written in the `Inconsolata` font,

the examples in the exercise parts are divided into 1 simple, 2 medium and 3 tricky.

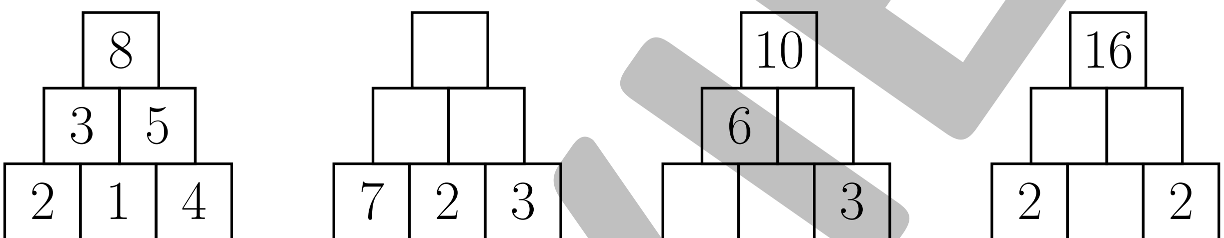

Come and Meet Ramsy!
Somewhere on a farm far, far away . . .
at the foot of a hill with mildly radioactive subsoil,
a Lacaune lamb was born.
As he grew, the ram became sturdy. His mates called him Ramsy.
The life of a ram is not usually that interesting, but this was quite an extraordinary specimen. Nobody knows how or when, but due to an inexplicable mutation, his body developed a fully functioning neocortex. And so it happened that our ungulate became highly intelligent.
Hi! I'm Ramsy. I like to graze and chew all kinds of grass and thoughts over. You can also find me reading books, discussing the subtleties of being and existence with my shepherd and, mainly, taking good care of my beloved sheep.
But one day the routine life, which Ramsy loved so much, turned upside down. It all started with a harmless chat with his shepherd.
Hi, Ramsy! I just wanted to talk to you about a few things. Actually, I think you'd better sit down.
Good morning, shepherd! It sounds like you have something serious to tell me. I'm all ears!
I think your job here is done. I have therefore decided to assign you to a new one.
What? Did I not do enough? I assure you, the well-being of our sheep is my top priority.
Exactly! When it comes to sheep, you are the master of well-being. But, as you well know, almost all of our sheep are your offspring. So, to prevent more inbreeding, we have to do something, right?
Say no more! No need to explain. We'll take your car and hit the road in search of a fine new flock! Thanks, you really know how to shepherd me through the valley of darkness! Let me just freshen up a bit . . .
Sorry, that's not exactly what I meant. We have to consider your future in light of the fact you're no spring chicken. I mean, let's face it, you're nearing retirement.
Retirement? I'm functioning at the top of my game! OK, so what do you suggest?
We basically have three options.
The first is to have a lavish feast. What do you think of making rogan josh?
Great! Please make sure there's plenty of garlic naan with extra garlic on top. I'm going to fill all four parts of my stomach! I don't know what this feast has to do with my job though, but who cares!


Well, I meant you would become a slow-cooked part of the dish, . . . But I agree with everything you've said about the naan. You're a true connoisseur.
I have always been a loyal servant to you and now you want to serve me on a plate? What have I done to deserve this? You can't be serious! What is the next option?
In theory, you could continue in your current position. You could do all the usual antics with the sheep, but only as a wether, if you don't mind, of course.
I do mind! A lot! I must admit, I'm getting pretty scared. I'm not sure if I want to know the last option.
In order to make our farm operate more efficiently, I could hire you as a **programmer**.
A programmer? What is that? Whatever it is, I must warn you, I know absolutely nothing about it, so you can't expect me to excel. But I do like the sound of the word "programmer" has a "ram" inside it.
Have you ever given a thought to all the smart gadgets we use every day? How do all those cool things actually work?
They run on electricity, that's how they work. It's quite simple.
Any AC power socket runs on electricity but it's not very entertaining, is it? And I'm not talking about the kind of entertainment you had when you were testing the conductivity of your hooves.
So, I can browse maps and do all sorts of fascinating things on your tablet thanks to programmers?
Exactly! Programmers create **programs**. Without programmers and their creations, all computers and smart gadgets would be useless. And one more thing: Next time, hooves off my tablet!
Alright. I'd also like to make useful and entertaining programs, but I don't have a clue how. I need a mentor to guide me through it.
Don't worry, I've got you covered. Let me introduce you to Mr. Lambda.
Howdy!
Ugh! What kind of creature is that? An earthworm? A nematode? An upside-down letter "Y"? A slingshot missing its elastic straps? A snake suffering from spina bifida? And what are those horrible sneakers?
My charming little body is shaped like the Greek letter "l", which most literate people write as **lambda**. In lower case it's written as λ.
Lamb-duh? That is the most unusual lamb I've ever seen. Anyway, I keep wondering whether I'll ever become a good programmer.
Only time will tell, but your intelligence and love for mathematics stand you in good stead. I suggest you do a little revision and get a good night's sleep. Our programming journey begins tomorrow!

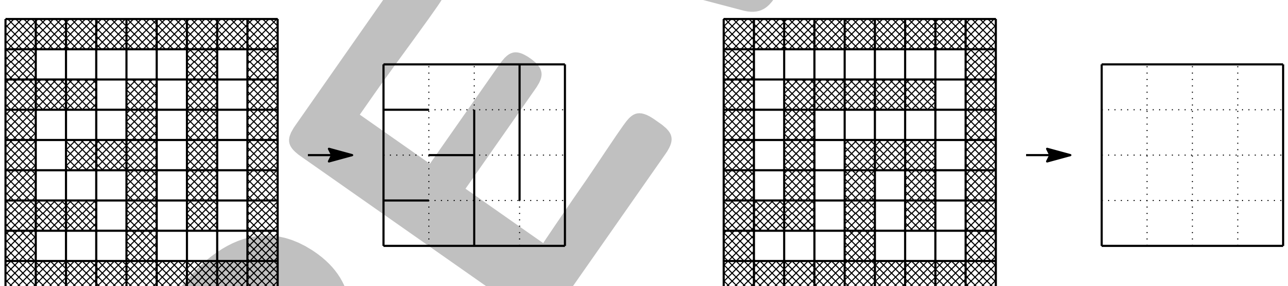
1 Complete the state diagrams. Look carefully at the direction of the arrow.



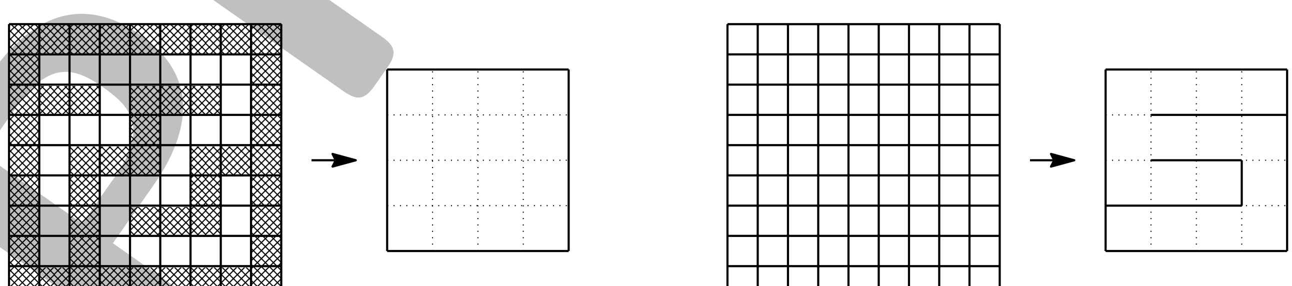
2 Fill in the missing numbers in the sum pyramids.

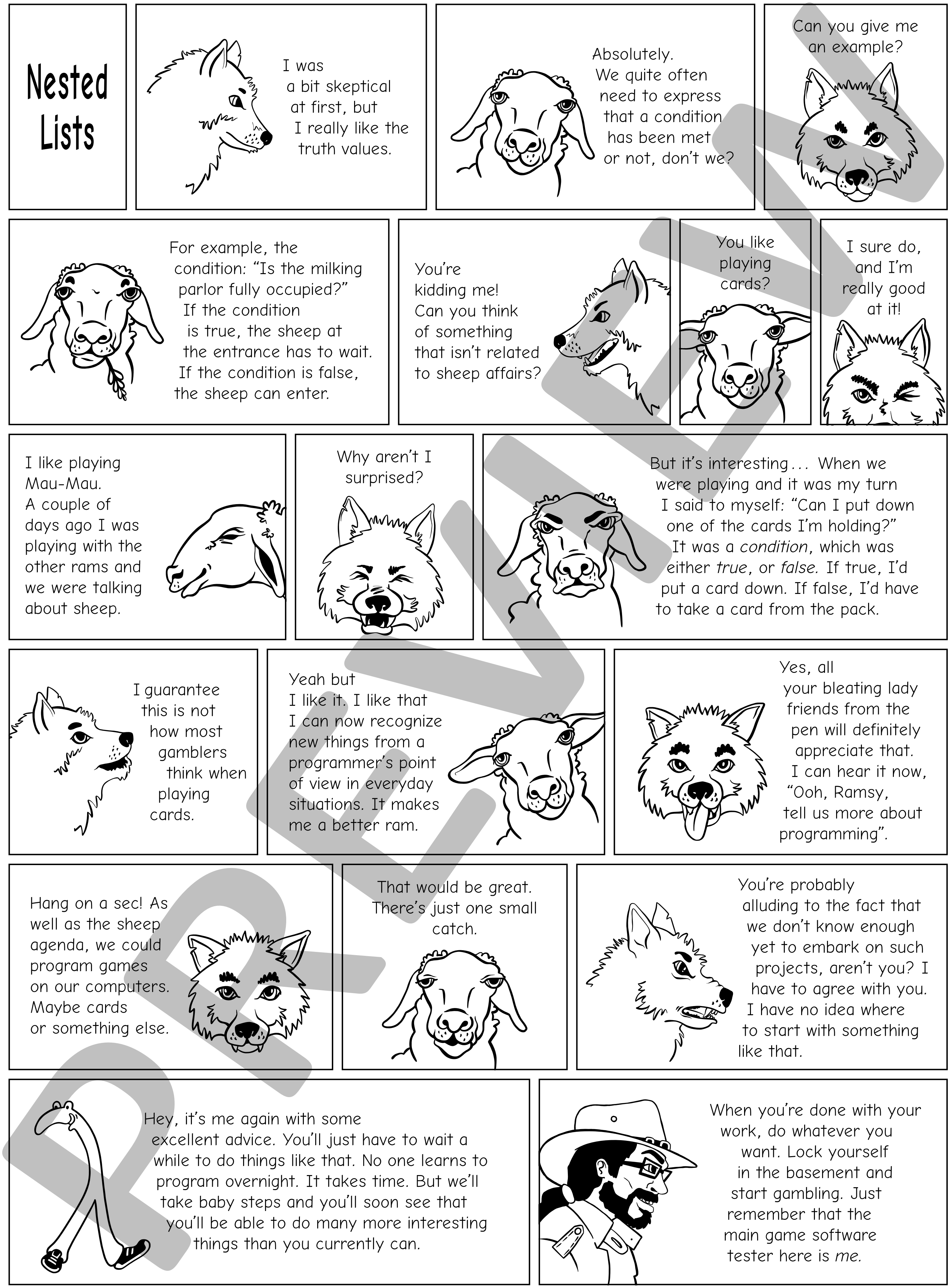


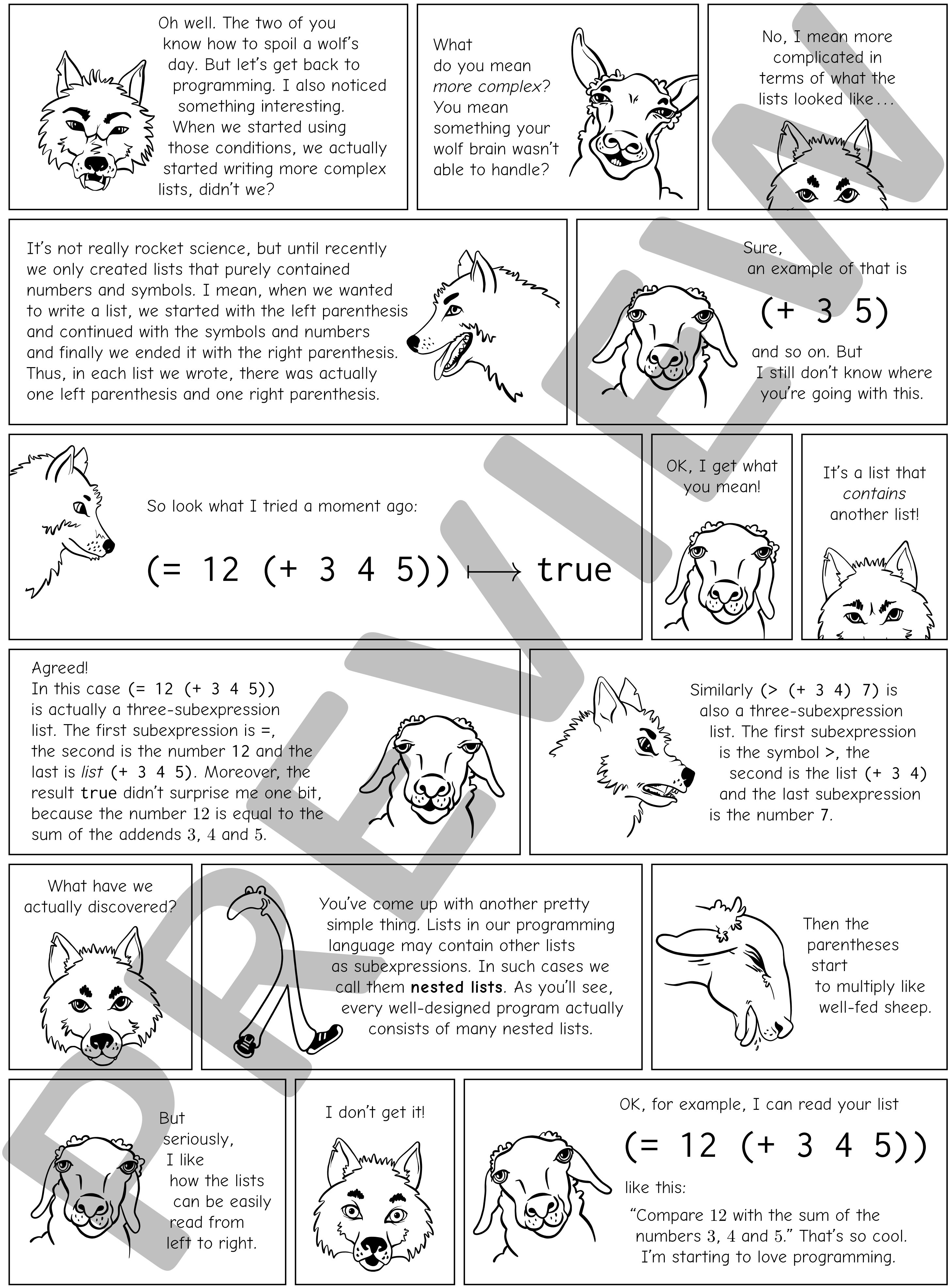
3 Redraw the maze according to the example.



4 Fill in the lines and squares on the blank maze.

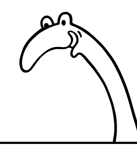



Nested Lists
I was a bit skeptical at first, but I really like the truth values.
Absolutely. We quite often need to express that a condition has been met or not, don't we?
Can you give me an example?
For example, the condition: "Is the milking parlor fully occupied?" If the condition is true, the sheep at the entrance has to wait. If the condition is false, the sheep can enter.
You're kidding me! Can you think of something that isn't related to sheep affairs?
You like playing cards?
I sure do, and I'm really good at it!
I like playing Mau-Mau. A couple of days ago I was playing with the other rams and we were talking about sheep.
Why aren't I surprised?
But it's interesting... When we were playing and it was my turn I said to myself: "Can I put down one of the cards I'm holding?" It was a *condition*, which was either *true*, or *false*. If true, I'd put a card down. If false, I'd have to take a card from the pack.
I guarantee this is not how most gamblers think when playing cards.
Yeah but I like it. I like that I can now recognize new things from a programmer's point of view in everyday situations. It makes me a better ram.
Yes, all your bleating lady friends from the pen will definitely appreciate that. I can hear it now, "Ooh, Ramsy, tell us more about programming".
Hang on a sec! As well as the sheep agenda, we could program games on our computers. Maybe cards or something else.
That would be great. There's just one small catch.
You're probably alluding to the fact that we don't know enough yet to embark on such projects, aren't you? I have to agree with you. I have no idea where to start with something like that.
Hey, it's me again with some excellent advice. You'll just have to wait a while to do things like that. No one learns to program overnight. It takes time. But we'll take baby steps and you'll soon see that you'll be able to do many more interesting things than you currently can.
When you're done with your work, do whatever you want. Lock yourself in the basement and start gambling. Just remember that the main game software tester here is *me.*


Oh well. The two of you know how to spoil a wolf's day. But let's get back to programming. I also noticed something interesting. When we started using those conditions, we actually started writing more complex lists, didn't we?
What do you mean *more complex*? You mean something your wolf brain wasn't able to handle?
No, I mean more complicated in terms of what the lists looked like . . .
It's not really rocket science, but until recently we only created lists that purely contained numbers and symbols. I mean, when we wanted to write a list, we started with the left parenthesis and continued with the symbols and numbers and finally we ended it with the right parenthesis. Thus, in each list we wrote, there was actually one left parenthesis and one right parenthesis.
Sure, an example of that is
(+ 3 5)
and so on. But I still don't know where you're going with this.
So look what I tried a moment ago:
(= 12 (+ 3 4 5)) ⟼ true
OK, I get what you mean!
It's a list that *contains* another list!
Agreed! In this case (= 12 (+ 3 4 5)) is actually a three-subexpression list. The first subexpression is =, the second is the number 12 and the last is *list* (+ 3 4 5). Moreover, the result **true** didn't surprise me one bit, because the number 12 is equal to the sum of the addends 3, 4 and 5.
Similarly (> (+ 3 4) 7) is also a three-subexpression list. The first subexpression is the symbol >, the second is the list (+ 3 4) and the last subexpression is the number 7.
What have we actually discovered?
You've come up with another pretty simple thing. Lists in our programming language may contain other lists as subexpressions. In such cases we call them **nested lists**. As you'll see, every well-designed program actually consists of many nested lists.
Then the parentheses start to multiply like well-fed sheep.
But seriously, I like how the lists can be easily read from left to right.
I don't get it!
OK, for example, I can read your list
(= 12 (+ 3 4 5))
like this:
"Compare 12 with the sum of the numbers 3, 4 and 5." That's so cool. I'm starting to love programming.

1 Complete the symbolic expressions. Check they're correct in REPL.

```
(= (+ 13 14) (+ 11 16)) ⟼ [    ]
(< (+ 3 14) (+ 8 5)) ⟼ [    ]
([    ] 8 (+ 2 5)) ⟼ true
(> (+ 8 15) [    ]) ⟼ true
(< (+ 8 7) [    ] (+ 8 9)) ⟼ true
([    ] (+ 7 4) (+ 4 7)) ⟼ false
(= (+ 7 9) (+ 5 [    ])) ⟼ true
(= (+ [    ] 5) (+ 2 6)) ⟼ true
(< (+ 2 8) (+ [    ] 3)) ⟼ true
```



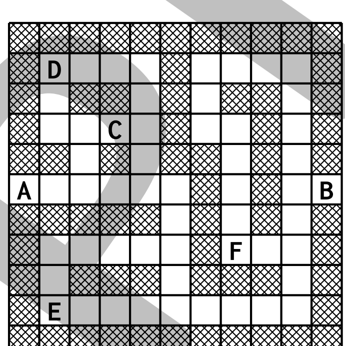
2 Estimate the results and check them in REPL.

```
(< 7 7) ⟼ [    ]
(<= 7 7) ⟼ [    ]
(< 5 7 7 9) ⟼ [    ]
(<= 5 7 7 9) ⟼ [    ]
```

3 Which value is called `secret`?

```
(def secret [    ])
(def x (+ secret 7))
(def y (+ secret x))
y ⟼ 23
```

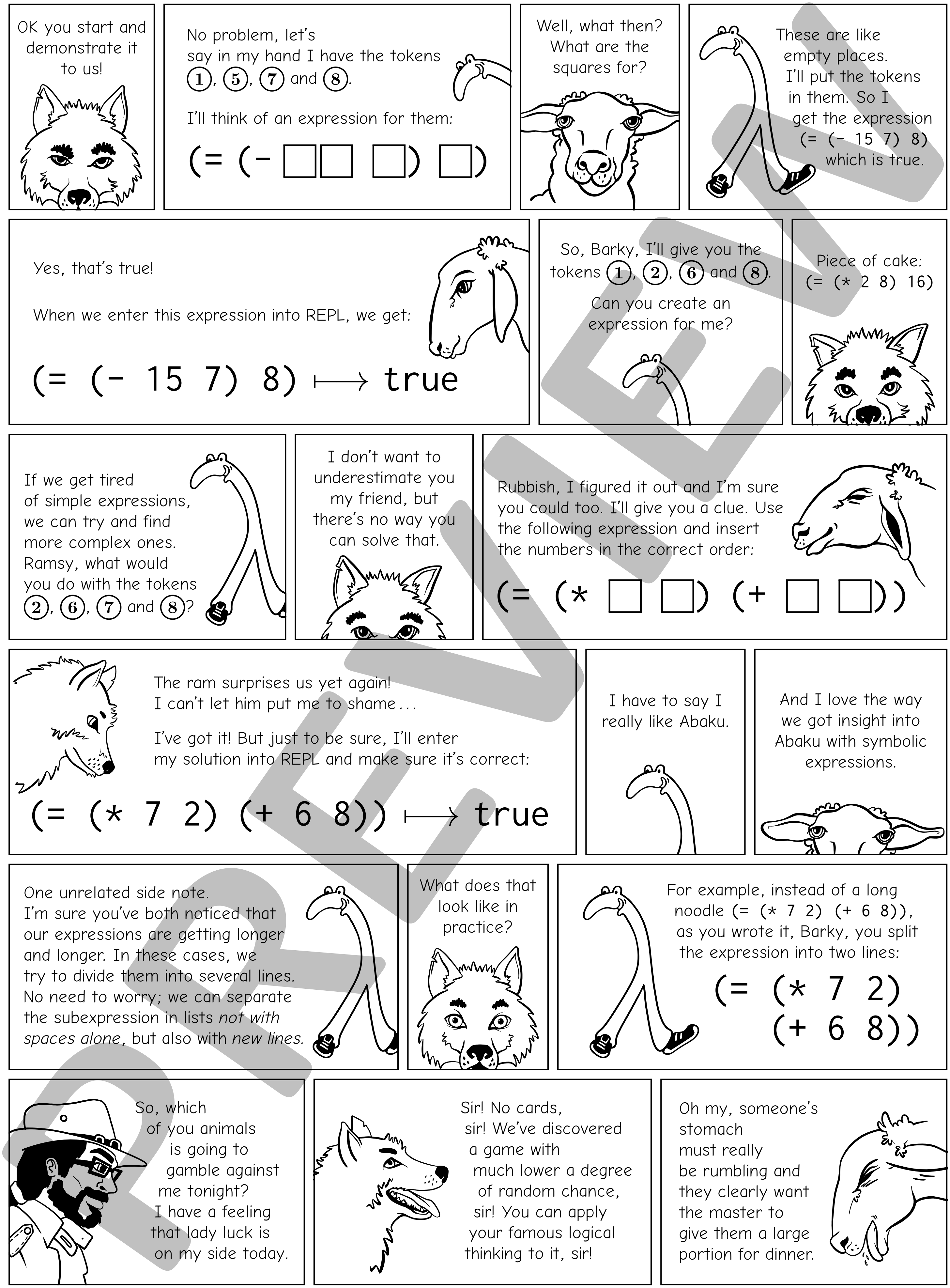
4 Navigate through the maze so that you enter each square no more than once. Write down every route from **A** to **B**.



A — B
A — ○ — B
A — ○ — B
A — ○ — B
A — ○ — B
A — ○ — B
A — ○ — ○ — B

A — ○ — ○ — B
A — ○ — ○ — B
A — ○ — ○ — B
A — ○ — ○ — B
A — ○ — ○ — ○ — B
A — ○ — ○ — ○ — B


Expressions and Abaku
It was pretty dense theory last time, Ramsy, let me tell you. All those functions, applications, and arguments. I feel like my cerebrospinal fluid is about to boil.
I'm glad we soaked up some of the technical terminology, but I suggest you have a day off today. Shall we play something together?
Sure, but I've had enough of arrow commands. Try to come up with something unusual.
I discovered a great game where we can both test our wit, speed of reasoning and combinatorial skills. It's called *Abaku* and it can be played with two or more players. All we need are mat and tokens.
Abaku you say? I'm curious what it's about.
I won't explain all the rules now, I'll just focus on the essentials. How about the token sequence 8 5 1 3?
They're like little wheels and there's a number on each of them!
Could you insert arithmetic operations and an equals sign between the tokens to create an equation?
I'd rather you show me how, I'm not in the best shape this morning.
So watch: $8 + 5 = 1\,3$ or $8 + 5 = 13$.
I see! That's what you meant! Of course, that's simple. Please give me another example.
How about 1 7 9 8?
Got it: $1\,7 - 9 = 8$ or $17 - 9 = 8$!
Or it could be expressed as $17 = 9 + 8$. This is essentially the basic principle of Abaku. You get tokens with numbers and you lay them out on the game board to form the longest possible correct arithmetic expressions.
This is an interesting game man. Can I join you?
Of course.
That day our three friends spent several hours playing the new game. At the end of a pleasantly spent afternoon, Mr. Lambda had an idea.
Gentlemen! I've had a little idea. How about we spice it up a bit?
Spice? Like adding something hot to sheep's milk?
Spice? Like something you'd add to lamb's broth to heighten the flavor?
Let's combine the game with programming! I have a sequence of digits—your task is to construct a symbolic expression containing these digits in the same order so that the expression is true.


OK you start and demonstrate it to us!
No problem, let's say in my hand I have the tokens 1, 5, 7 and 8.
I'll think of an expression for them:
(= (- □□ □) □)
Well, what then? What are the squares for?
These are like empty places. I'll put the tokens in them. So I get the expression (= (- 15 7) 8) which is true.
Yes, that's true!
When we enter this expression into REPL, we get:
(= (- 15 7) 8) ⟼ true
So, Barky, I'll give you the tokens 1, 2, 6 and 8.
Can you create an expression for me?
Piece of cake: (= (* 2 8) 16)
If we get tired of simple expressions, we can try and find more complex ones. Ramsy, what would you do with the tokens 2, 6, 7 and 8?
I don't want to underestimate you my friend, but there's no way you can solve that.
Rubbish, I figured it out and I'm sure you could too. I'll give you a clue. Use the following expression and insert the numbers in the correct order:
(= (* □ □) (+ □ □))
The ram surprises us yet again! I can't let him put me to shame...
I've got it! But just to be sure, I'll enter my solution into REPL and make sure it's correct:
(= (* 7 2) (+ 6 8)) ⟼ true
I have to say I really like Abaku.
And I love the way we got insight into Abaku with symbolic expressions.
One unrelated side note. I'm sure you've both noticed that our expressions are getting longer and longer. In these cases, we try to divide them into several lines. No need to worry; we can separate the subexpression in lists *not with spaces alone*, but also with *new lines*.
What does that look like in practice?
For example, instead of a long noodle (= (* 7 2) (+ 6 8)), as you wrote it, Barky, you split the expression into two lines:
(= (* 7 2)
   (+ 6 8))
So, which of you animals is going to gamble against me tonight? I have a feeling that lady luck is on my side today.
Sir! No cards, sir! We've discovered a game with much lower a degree of random chance, sir! You can apply your famous logical thinking to it, sir!
Oh my, someone's stomach must really be rumbling and they clearly want the master to give them a large portion for dinner.

1 Fill the values from the tokens in the correct order so that all the given symbolic expressions are true.

(1) (3) (5) (8)

(= (+ 5 8) 13)

(1) (5) (7) (8)

(= (+ □ □) □□)

(1) (4) (5) (9)

(= (- □□ □) □)

(2) (3) (7) (9)

(= (* □ □) □□)

2 Use the tokens (3), (5) and (8) to fill in the expressions. Find all the solutions.

(= (+ □ □) □) ⟼ true

(= (+ □ □) □) ⟼ true

(= (- □ □) □) ⟼ true

(= (- □ □) □) ⟼ true

(= □ (+ □ □)) ⟼ true

(= □ (+ □ □)) ⟼ true

(= □ (- □ □)) ⟼ true

(= □ (- □ □)) ⟼ true

3 Use the numbers from the tokens and form true symbolic expressions according to the pattern.

(1) (2) (4) (6) (8)

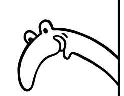
(2) (2) (3) (5) (7)

(3) (3) (4) (5) (8)

(1) (2) (3) (6) (7)

(4) (4) (5) (5) (9)

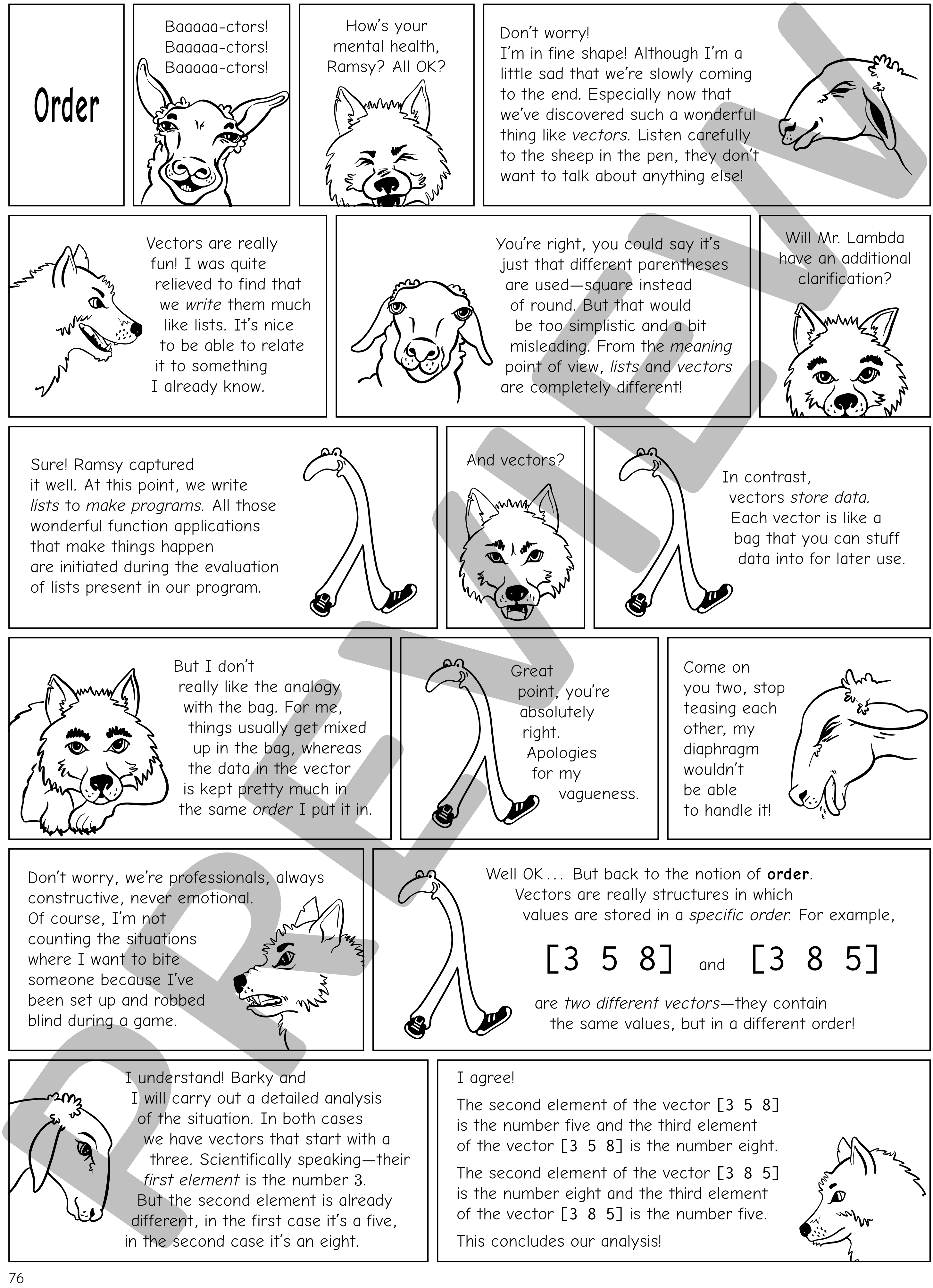
(= (+ 16 8) 24) ⟼ true

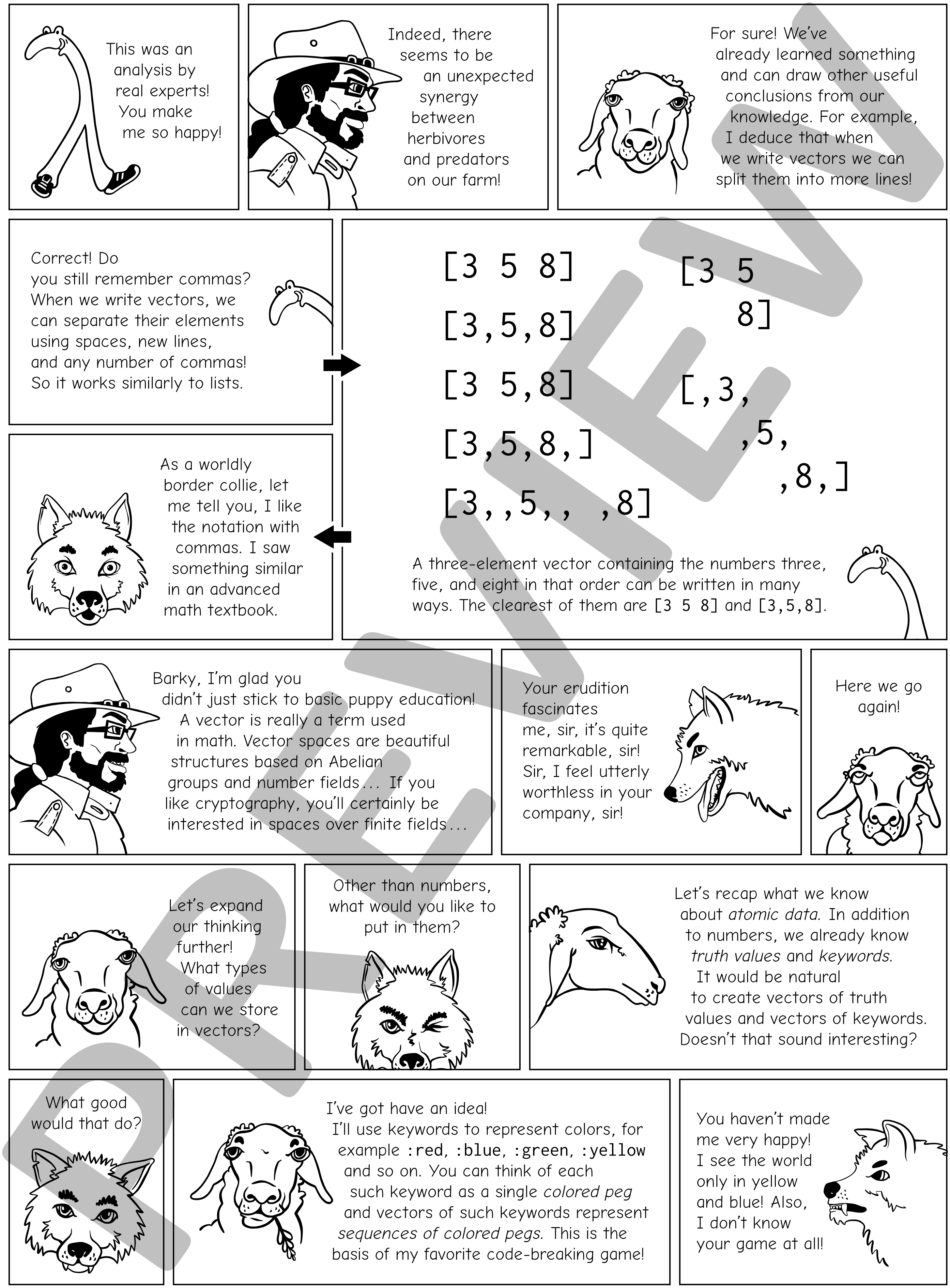


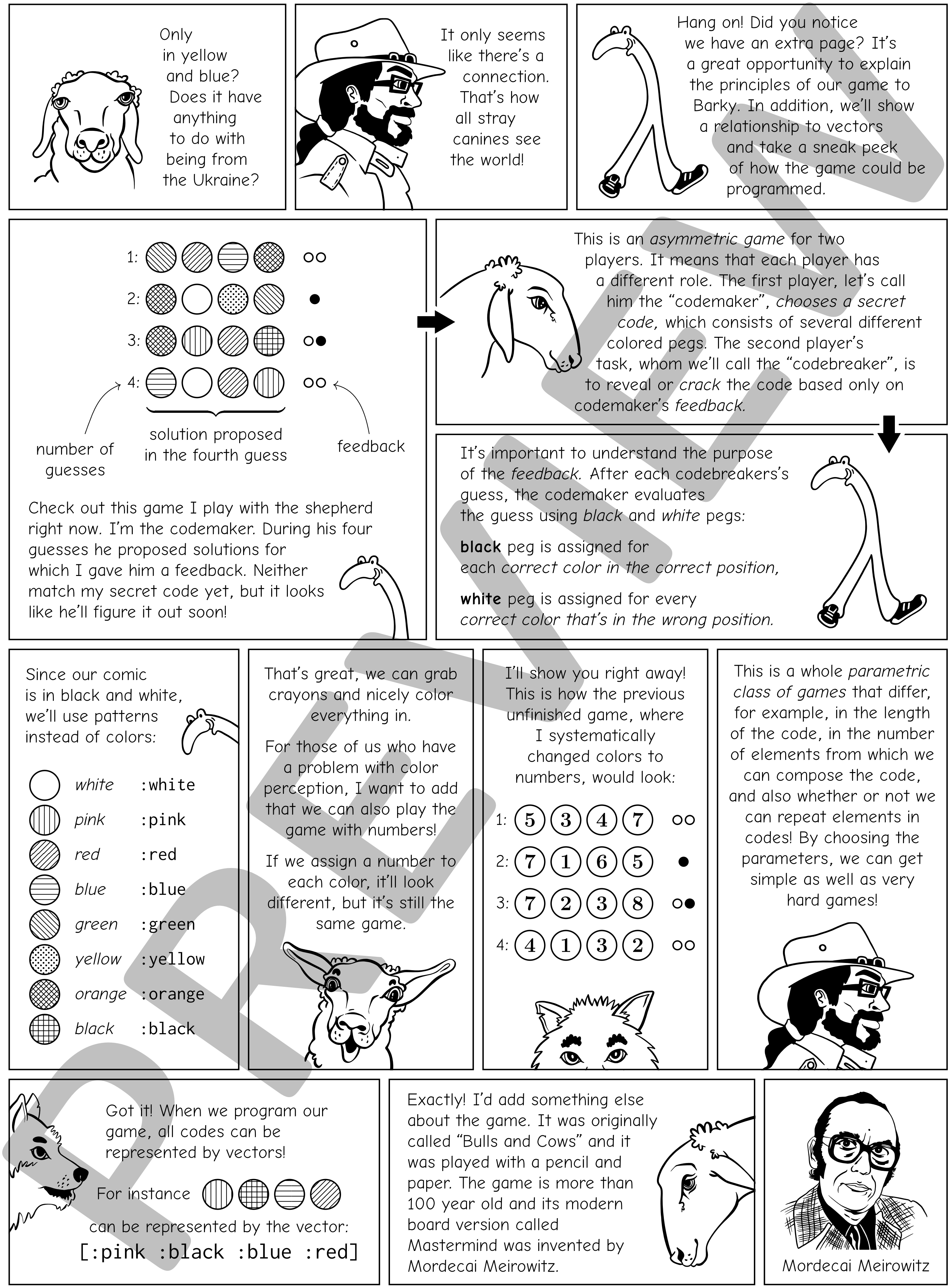
4 Which groups of tokens will you use to make a true expression?

(1) (2) (6) (8)	☑
(2) (4) (6) (7)	☐
(3) (7) (8) (9)	☐
(1) (3) (6) (8)	☐
(1) (2) (3) (4)	☐

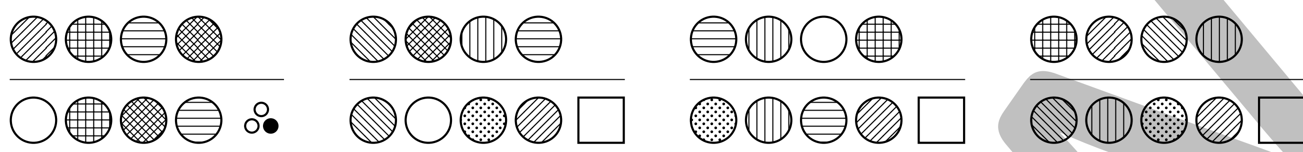
5 Construct a true expression from the tokens (1), (2), (4) and (9).


Order
Baaaaa-ctors! Baaaaa-ctors! Baaaaa-ctors!
How's your mental health, Ramsy? All OK?
Don't worry! I'm in fine shape! Although I'm a little sad that we're slowly coming to the end. Especially now that we've discovered such a wonderful thing like *vectors.* Listen carefully to the sheep in the pen, they don't want to talk about anything else!
Vectors are really fun! I was quite relieved to find that we *write* them much like lists. It's nice to be able to relate it to something I already know.
You're right, you could say it's just that different parentheses are used—square instead of round. But that would be too simplistic and a bit misleading. From the *meaning* point of view, *lists* and *vectors* are completely different!
Will Mr. Lambda have an additional clarification?
Sure! Ramsy captured it well. At this point, we write *lists* to *make programs.* All those wonderful function applications that make things happen are initiated during the evaluation of lists present in our program.
And vectors?
In contrast, vectors *store data.* Each vector is like a bag that you can stuff data into for later use.
But I don't really like the analogy with the bag. For me, things usually get mixed up in the bag, whereas the data in the vector is kept pretty much in the same *order* I put it in.
Great point, you're absolutely right. Apologies for my vagueness.
Come on you two, stop teasing each other, my diaphragm wouldn't be able to handle it!
Don't worry, we're professionals, always constructive, never emotional. Of course, I'm not counting the situations where I want to bite someone because I've been set up and robbed blind during a game.
Well OK... But back to the notion of **order**. Vectors are really structures in which values are stored in a *specific order.* For example,
[3 5 8] and [3 8 5]
are *two different vectors*—they contain the same values, but in a different order!
I understand! Barky and I will carry out a detailed analysis of the situation. In both cases we have vectors that start with a three. Scientifically speaking—their *first element* is the number 3. But the second element is already different, in the first case it's a five, in the second case it's an eight.
I agree!
The second element of the vector [3 5 8] is the number five and the third element of the vector [3 5 8] is the number eight.
The second element of the vector [3 8 5] is the number eight and the third element of the vector [3 8 5] is the number five.
This concludes our analysis!


This was an analysis by real experts! You make me so happy!
Indeed, there seems to be an unexpected synergy between herbivores and predators on our farm!
For sure! We've already learned something and can draw other useful conclusions from our knowledge. For example, I deduce that when we write vectors we can split them into more lines!
Correct! Do you still remember commas? When we write vectors, we can separate their elements using spaces, new lines, and any number of commas! So it works similarly to lists.
[3 5 8]
[3,5,8]
[3 5,8]
[3,5,8,]
[3,,5,, ,8]
[3 5
8]
[,3,
,5,
,8,]
A three-element vector containing the numbers three, five, and eight in that order can be written in many ways. The clearest of them are [3 5 8] and [3,5,8].
As a worldly border collie, let me tell you, I like the notation with commas. I saw something similar in an advanced math textbook.
Barky, I'm glad you didn't just stick to basic puppy education! A vector is really a term used in math. Vector spaces are beautiful structures based on Abelian groups and number fields... If you like cryptography, you'll certainly be interested in spaces over finite fields...
Your erudition fascinates me, sir, it's quite remarkable, sir! Sir, I feel utterly worthless in your company, sir!
Here we go again!
Let's expand our thinking further! What types of values can we store in vectors?
Other than numbers, what would you like to put in them?
Let's recap what we know about *atomic data.* In addition to numbers, we already know *truth values* and *keywords.* It would be natural to create vectors of truth values and vectors of keywords. Doesn't that sound interesting?
What good would that do?
I've got have an idea! I'll use keywords to represent colors, for example :red, :blue, :green, :yellow and so on. You can think of each such keyword as a single *colored peg* and vectors of such keywords represent *sequences of colored pegs.* This is the basis of my favorite code-breaking game!
You haven't made me very happy! I see the world only in yellow and blue! Also, I don't know your game at all!

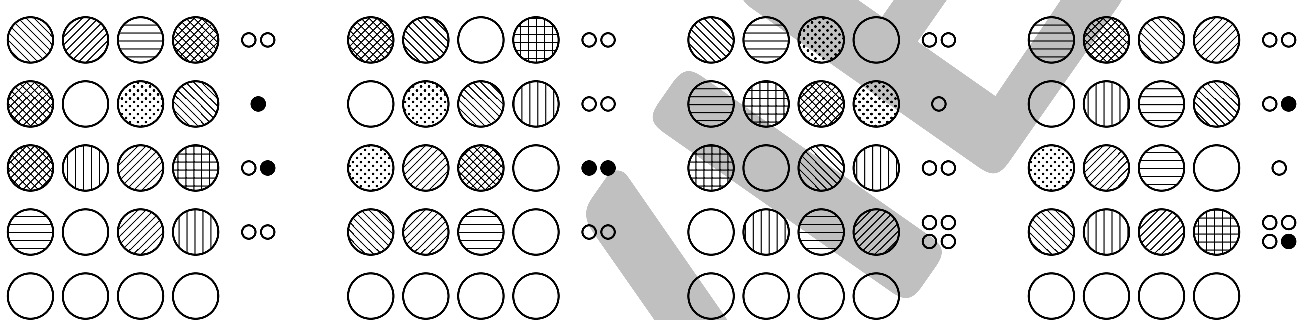

Only in yellow and blue? Does it have anything to do with being from the Ukraine?
It only seems like there's a connection. That's how all stray canines see the world!
Hang on! Did you notice we have an extra page? It's a great opportunity to explain the principles of our game to Barky. In addition, we'll show a relationship to vectors and take a sneak peek of how the game could be programmed.
1:
2:
3:
4:
number of guesses
solution proposed in the fourth guess
feedback
Check out this game I play with the shepherd right now. I'm the codemaker. During his four guesses he proposed solutions for which I gave him a feedback. Neither match my secret code yet, but it looks like he'll figure it out soon!
This is an *asymmetric game* for two players. It means that each player has a different role. The first player, let's call him the "codemaker", *chooses a secret code,* which consists of several different colored pegs. The second player's task, whom we'll call the "codebreaker", is to reveal or *crack* the code based only on codemaker's *feedback.*
It's important to understand the purpose of the *feedback.* After each codebreakers's guess, the codemaker evaluates the guess using *black* and *white* pegs:
black peg is assigned for each *correct color in the correct position,*
white peg is assigned for every *correct color that's in the wrong position.*
Since our comic is in black and white, we'll use patterns instead of colors:
white :white
pink :pink
red :red
blue :blue
green :green
yellow :yellow
orange :orange
black :black
That's great, we can grab crayons and nicely color everything in.
For those of us who have a problem with color perception, I want to add that we can also play the game with numbers!
If we assign a number to each color, it'll look different, but it's still the same game.
I'll show you right away! This is how the previous unfinished game, where I systematically changed colors to numbers, would look:
1: 5 3 4 7
2: 7 1 6 5
3: 7 2 3 8
4: 4 1 3 2
This is a whole *parametric class of games* that differ, for example, in the length of the code, in the number of elements from which we can compose the code, and also whether or not we can repeat elements in codes! By choosing the parameters, we can get simple as well as very hard games!
Got it! When we program our game, all codes can be represented by vectors!
For instance
can be represented by the vector:
[:pink :black :blue :red]
Exactly! I'd add something else about the game. It was originally called "Bulls and Cows" and it was played with a pencil and paper. The game is more than 100 year old and its modern board version called Mastermind was invented by Mordecai Meirowitz.
Mordecai Meirowitz

1 Be the codemaster! Complete the feedback for each pair consisting of the code (top) and the guess (bottom).



3 2 5 1	7 3 1 4	[2 8 1 3]	[1 6 4 8]
8 2 5 3 □	1 4 3 7 □	[8 3 1 5] □	[2 5 8 3] □

2 Try to break codes in the following situations. Each pattern can appear only once in the code.



3 Try to break codes in the following situations. Each pattern can appear only once in the code.

3 1 8 4 ○○	8 7 5 4 ○○	6 5 7 8 ○●	5 2 1 3 ○○●
5 8 4 2 ○○●	3 8 1 7 ○●	3 5 2 6 ○○	3 6 1 2 ○○
5 3 2 8 ○●	1 8 2 5 ●	4 3 6 8 ●●	1 2 5 4 ○○
6 2 4 8 ●●	1 6 4 7 ○○●●	1 2 6 8	5 3 2 7 ●●●
○○○○	○○○○	○○○○	○○○○

4 The same task again. This time the guesses are represented by vectors of numbers.

[3 4 7 1] ●	[6 5 4 3] ○○○	[2 5 4 8] ●●●	[3 6 5 8] ●
[6 5 7 8] ○●	[2 4 5 6] ○○	[8 5 4 1] ●●	[1 6 4 7] ○○
[6 4 2 5] ○○	[4 6 3 7] ○●	[6 5 4 2] ●●●	[3 2 7 4] ○○●
[2 5 8 1] ○○	[3 6 1 5] ●●●	[7 5 4 2] ●●●	[3 7 2 1] ○●
[ ]	[ ]	[ ]	[ ]

Notes

N o t e s

Enjoying our textbook?

Join us on Patreon

for more content and support!

https://patreon.com/progmat

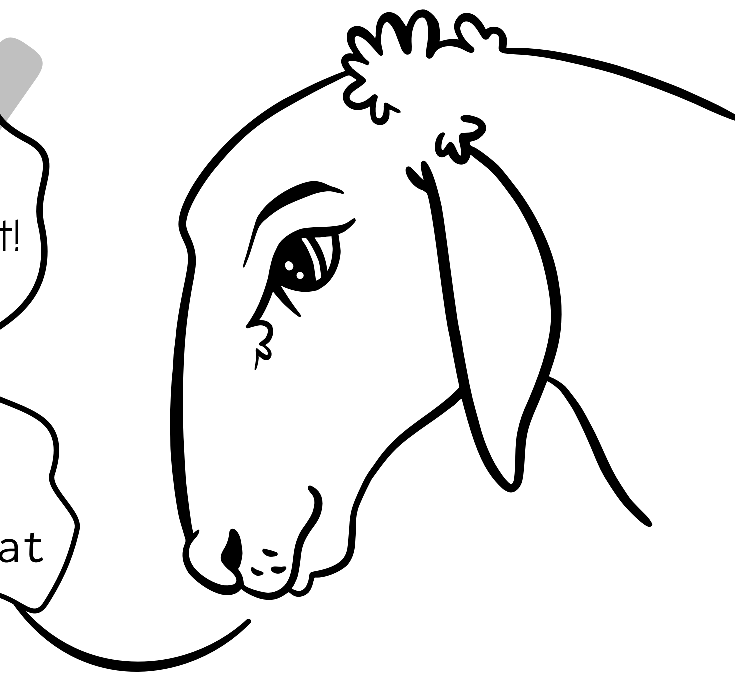
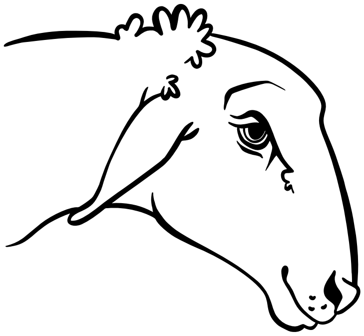
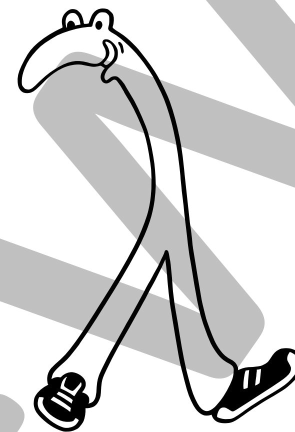


Table of Contents





Programming and Mathematics A1

Author: Vilém Vychodil
Original title: Programování a matematika A1
Original publisher: PM Books s.r.o.
Translation: Nicola Catherine Fořtová, Vilém Vychodil
Illustrations: Monika Klimková
Typesetting and graphic design: Vilém Vychodil

Published by: PM Books s.r.o., Dittrichova 346/4, Nové Město, 120 00 Praha 2, CZ (https://prog-mat.com)
Print: Lulu Press, Inc., 627 Davis Drive Suite 300, Morrisville, NC 27560, USA (https://lulu.com)

First published in: 2023

ISBN 978-80-908730-2-5

Programming and Mathematics A1

https://prog-mat.com

ISBN 978-80-908730-2-5

9 788090 873025