Using Computer Programming as an Effective Complement to Mathematics Education: Experimenting with the Standards for Mathematics Practice in a Multidisciplinary Environment for Teaching and Learning with Technology in the 21st Century

By Pavel Solin[1] and Eugenio Roanes-Lozano[2]

[1]University of Nevada, Reno, 1664 N Virginia St, Reno, NV 89557, USA. Founder and Director of NCLab (http://nclab.com).
[2]Instituto de Matemática Interdisciplinar & Departamento de Didáctica de las Ciencias Experimentales, Sociales y Matemáticas, Facultad de Educación, Universidad Complutense de Madrid, c/ Rector Royo Villanova s/n, 28040 – Madrid, Spain.

solin@unr.edu, eroanes@mat.ucm.es

Received: 30 September 2018 Revised: 12 February 2019

DOI: 10.1564/tme_v27.3.03

Many mathematics educators are not aware of a strong connection that exists between the education of computer programming and mathematics. The reason may be that they have not been exposed to computer programming. This connection is worth exploring, given the current trends of automation and Industry 4.0. Therefore, in this paper we take a closer look at the Common Core's eight Mathematical Practice Standards. We show how each one of them can be reinforced through computer programming. The following discussion is virtually independent of the choice of a concrete programming language. Therefore, in the interest of simplicity, we will use a well-known educational programming language named Karel the Robot based on (Pattis, 1995) which is freely available online in NCLab (NCLab, 2019). The visual character of this language will allow us to provide more illustrative examples than would be possible with a standard programming language such as Java, C++ or Python.

1. INTRODUCTION

The U.S. Common Core's Mathematical Practice Standards are strongly influenced by the mathematics standards of top-performing countries, and they are of international interest - see, for example (Common Core Standards, 2019; TeacherStep, 2015; DreamBox Learning, 2013; Everett, 2013). However, none of these standards mentions computer programming. The purpose of this paper is to suggest that computer programming can be an attractive and highly effective complement to mathematics education (this is one of the different possible approaches to STEM, like the use of dynamic geometry systems (Budinski, 2017), computer algebra systems (Rosa and Petrášková, 2017) or a dynamic geometry system with computer algebra capabilities (Kovács, Recio and Vélez, 2018)). This claim has been confirmed by many K-12 schools in the U.S. and other countries which use NCLab to teach computer programming. NCLab (2018) is a free public cloud computing platform which provides K-12 schools with easy access to computer programming and related activities which include mathematics, statistics, geometry, 3D modeling, that belong to critical 21st century STEM skills (Solin, 2017).

2. KAREL THE ROBOT

Karel the Robot is a widely used educational programming language which was introduced by Richard E. Pattis in his 1981 textbook *Karel the Robot: A Gentle Introduction to the Art of Computer Programming* (Pattis, 1995). Let us note that Karel the Robot constitutes an environment related to Turtle Geometry (Abbelson and diSessa, 1981), but is not yet another implementation, as will be detailed below.

Summarizing, in Turtle Geometry the graphic cursor (denoted "turtle") can move forward and backward any number of steps and can turn clockwise and counter clockwise any angle. Therefore, the corresponding commands (`FD`, `BK`, `RT`, `LT`) have an input. This input can be either an integer or a (finite) decimal representation or finite decimal approximation of a real number.

There are many implementations of Turtle Geometry available. The programming language directly related to Turtle Geometry is Logo. An updated comprehensive list of Logo implementations and related software, including more than 300 references to different dialects, can be found in Logo Tree (2019). Among them we could underline Berkeley Logo (Harvey, 2008), FMSLogo (FMSLogo, n. a.) and NetLogo (Wilensky, 2019). Nevertheless the Logo language is not widely used today.

In the 1980s and 1990s, Turtle Geometry implementations were included in programming languages such as Turbo Pascal and Turbo Prolog. The second author was one of the authors of an improved version for Turbo Pascal (Roanes-Lozano and Roanes-Macías, 1994a) and a version for the computer algebra system Maple (Roanes-Lozano and Roanes-Macías, 1994b).

But implementations of Turtle Geometry can also be found in "modern" computer languages such as Scratch 3 (Scratch, n. a.), Python (Rachum, 2015), Haskell (Graphics.X11.Turtle, n.a.); Vera Ruiz, 2011) or Java (Haas, 2016). The computer algebra system Xcas also includes an implementation of Turtle Geometry. Also, the One Laptop per Child project includes turtle activities.

Meanwhile Karel the Robot movement commands have no input: one step forward and 90 degrees clockwise and counter clockwise turns are considered (`go`, `right`, `left`). Therefore we could say that the turtle lives in a continuous world meanwhile Karel the Robot lives in a discrete world.

Moreover, Karel the Robot can interact with his world in ways that do not exist in Turtle Geometry. For example, as seen above, there are commands for picking and dropping objects, more related to robotics than to geometry.

These characteristics make Karel the Robot really appropriate for beginners in programming, especially in a beginners friendly environment such as NCLab.

Interestingly, the same two approaches mentioned above (continuous / discrete) can be found in educational robots such as Pro-Bot (continuous) (Pro-Bot, n.a.) and Code and Go Mouse (discrete). Let us mention that, originally, the turtle was a mechanical device because, when Logo was developed in the 1960s (Logo, 2019), the computers usually only had text monitors.

3. NCLAB

NCLab (NCLab, 2019) is a free public cloud computing platform which provides a large number of free apps related to mathematics and computing. These apps include computer programming in several languages including Karel the Robot, Python, Java, Javascript and others. It also provides 3D modeling apps based on the open source libraries PLaSM (PlaSM, 2019) and OpenSCAD (OpenSCAD, 2019). It provides the widely used typesetting system LaTeX (The LaTeX Project, 2019), a computer algebra system named Sympy based on Python (Sympy, 2019), computing with GNU Octave (GNU Octave, 2019), SciPy and NumPy (SciPy.org / NumPy, 2019), statistical computations with R (R, 2019) etc.

4. COMMON CORE'S MATHEMATICS PRACTICE STANDARD #1

"Make sense of problems and persevere in solving them."

The first Common Core mathematical practice standard is found in almost every mathematics problem across the board. It means that students must understand the problem, figure out how to solve it, and then work until it is finished. Common Core standards (that cover from Kindergarten to Grade 12) encourage students to work with their current knowledge bank and apply the skills they already have while evaluating themselves in problem-solving. This standard is easily tested using problems with a tougher skill level than already mastered. While students work through more difficult problems, they focus on the process of solving the problem instead of just getting to the correct answer.

If one omits the interaction with the hardware, computer programming is mostly logic, occasionally complemented with other areas of mathematics such as algebra, geometry or calculus. Therefore, the previous paragraph in its entirety applies to computer programming as well as to mathematics. To solve a problem, students need to understand it well, and persevere in solving it. The student's perseverance is often more heavily tested in computer programming because the smallest glitch in syntax or logic means a complete failure – the program crashes with an error message, or just delivers an incorrect outcome.

For illustration let us solve a relatively simple task where Karel (the yellow robot) needs to move three books on the marks (Figure 1). The students must understand the task, and figure out the correct sequence of operations which will lead to the desired outcome.

Figure 1 Karel needs to move three books on the marks.

The students know the commands `go` (make one step forward), `get` (collect an object which is beneath the robot), `put` (drop an object on the ground) and `repeat N` (repeat something `N` times). They also know that they should write one command per line, and that the commands inside the `repeat` loop must be indented. Anyway, most of them need three or more attempts to solve this task. To get a better idea of what it takes, try it yourself! The solution is provided at the end of the paper in the Appendix.

In general, computer programs hardly ever work the first time. Students learn quickly that trying to solve a problem without understanding it well does not work, and that trying without thinking does not help either. They learn to go back, analyse where their thinking was wrong, and make adjustments. This strengthens their perseverance skills through "try and try" (Capraro et al, 2012; Shaheed Hartley and Treagust, 2014).

5. COMMON CORE'S MATHEMATICS PRACTICE STANDARD #2

"Reason abstractly and quantitatively."

When trying to problem solve, it is important that students understand there are multiple ways to break apart the problem in order to find the solution. Using symbols, pictures or other representations to describe the different sections of the problem will allow students to use context skills rather than standard algorithms.

Let's use an example of recursion to show the abstract thinking process which takes place. Karel's task is to collect all shields and enter the home square in Figure 2.

Figure 2 Karel needs to recursively collect shields.

This task is suitable for recursion because after moving forward one step and collecting one shield, the robot is ready to solve the same task: Collect all shields and enter the home square. Here is the corresponding code. Notice that after collecting one shield and moving one step forward, the command `walk` calls itself on line 9:

```
# Recursive command:
def walk
  if shield
    get
  go
  if not home
    walk
  return

# Main program:
walk
```

Understanding recursion requires abstract thinking. Namely, each new call to the command walk creates and starts a new copy of the command, while the current copy is put on hold. The sequence of calls is visible in the diagram of Figure 3.

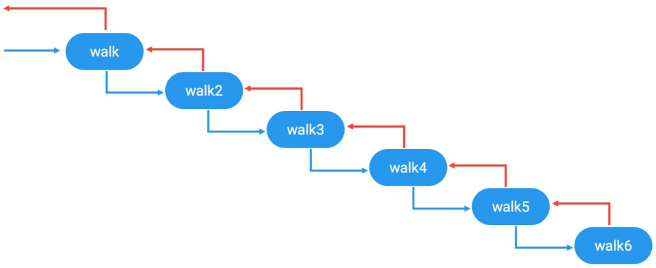

Figure 3 Scheme of the recursive calls.

6. COMMON CORE'S MATHEMATICS PRACTICE STANDARD #3

"Construct viable arguments and critique the reasoning of others."

This standard is aimed at creating a common mathematical language that can be used to discuss and explain mathematics as well as support or object others' work. Mathematics vocabulary is easily integrated into daily lesson plans in order for students to be able to communicate effectively. "Talk moves" are important in developing and building communication skills and can include such simple tasks as restating a fellow classmate's reasoning or even supporting their own reason for agreeing or disagreeing. Prompting students to participate further in class mathematical discussion will help build student communication skills.

Computer programming requires logical thinking and systematic problem solving. According to our experience, the students tend to work together, exchange ideas, and discuss various strategies to solve the programming tasks (in comparison with traditional work not based on the use of technological resources). Obviously, this doesn't mean that all difficulties disappear. For instance, in (Fujita, Doney and Wegeri, 2019), the authors underline that even moving from the level of collaborative learning process "collective image making" to the level "collective property noticing" is not straightforward. But we believe that our assertion is clearly influenced by the use of this kind of technological environment: for instance in Spain computer labs at all levels (from primary school to university) almost always have one computer for every two students, what forces them to collaborate and trains them during many years in collaborating (!), at least with a computer mate. There will be students who surprisingly and unexpectedly will excel at it. Here is one example for all (Figure 4):

> *"I was in the lab the other day with Jan and watched the students working with Karel. It was SO EXCITING. We had one student who is especially hard to work with normally, that excelled at Karel. He became the helper to other students who often think of him as a bully. Instead, he was the resource for help. It was so exciting to see!"* (Cammie Briggs, vice-principal at the David E. Norman Elementary, White Pine County, Nevada, U.S.A., 2018)

Figure 4 Students communicate while solving programming tasks

7. COMMON CORE'S MATHEMATICS PRACTICE STANDARD #4

"Model with mathematics."

Mathematics does not end at the classroom door. Learning to model with mathematics means that students will use mathematics skills to problem-solve real world situations. This can range from organizing different types of data to using mathematics to help understand life connections. Using real world situations to show how mathematics can be used in many different aspects of life helps mathematics to be relevant outside of mathematics classroom.

Area under a curve

For illustration, let us use programming to calculate the area under a curve. Karel has a GPS device, represented by the commands `gpsx` and `gpsy`. These commands return the horizontal and vertical coordinates of the robot in the maze, respectively. The Southwest corner of the maze has coordinates (0, 0) and the Northeast corner has (14, 11). The students are asked to write a program for the robot to count the number of crates that form a building (Figure 5). The shape of the building is random. In reality, students are calculating the area under a curve that is represented by the contour of the pile of crates (otherwise called "definite integral" in calculus).

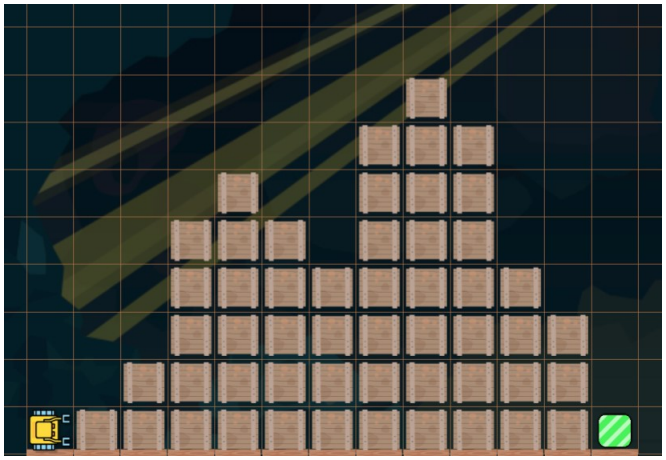

Figure 5 Karel calculates the area under a curve

The corresponding program has two parts – a custom command `nexttop` which moves Karel to the top of the next column, and a main program which adds the height of all columns to the variable area:

```
def nexttop
  while crate
    left
    go
    right
  go
  right
  while not (crate or wall)
    go
  left
area = 0
while not home
  nexttop
  area += gpsy
print("Area =", area)
```

With the above maze, the output of this program is "`Area = 52`."

Law of large numbers

As another example of modeling with mathematics, let's model the Law of large numbers. This law of probability states that with larger number of experiments, the relative frequency of an event converges to its predicted probability.

Karel has a backpack full of ribbons, and he wants to split them into two parts by tossing a coin (Figure 6). He can toss a coin using the command `rand` which returns `True` or `False` with 50% probability (that is, `rand` is a Boolean command).

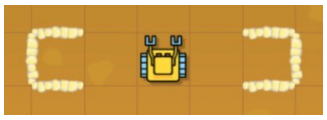

Figure 6 Karel models the Law of large numbers

Here is the corresponding program:

```
repeat 20
  if rand
    left
    go
    go
    put
    right
    right
    go
    go
    left
  else
    right
    go
    go
    put
    left
    left
    go
    go
    right
```

The result after 20 coin tosses is 12 ribbons on the left and 8 on the right (Figure 7).

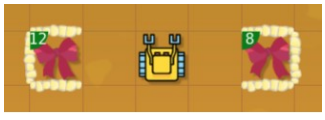

Figure 7 Result after 20 coin tosses (12 left, 8 right)

The result after 100 coin tosses is 44 ribbons on the left and 56 on the right (Figure 8).

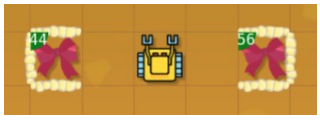

Figure 8 Result after 100 coin tosses (44 left, 56 right)

The result after 500 coin tosses is 238 ribbons on the left and 262 on the right (Figure 9).

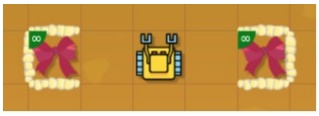

Figure 9 Result after 500 coin tosses (238 left, 262 right)

If the perfect outcome is defined as a split into two halves, then in the first case, the relative error was (10 - 8) / 20 = 10%. In the second case, the relative error was (50 – 44) / 100 = 6% and in the last case (250 – 238) / 500 = 2.4%. Hence one can clearly see that the relative frequency of the event with probability 0.5 converges to 0.5.

8. COMMON CORE'S MATHEMATICS PRACTICE STANDARD #5

"Use appropriate tools strategically."

One of the Common Core's biggest components is to provide students with the assets they need to navigate the real world. In order for students to learn what tools should be used in problem solving it is important to remember that no one will be guiding students through the real world – telling them which mathematics tool to use. By leaving the problem open ended, students can select which mathematics tools to use and discuss what worked and what didn't ¾different authors like Alman (2017); Cha, Kwon and Lee (2007); Yazgan-Sağ and Emre-Akdoğan (2016), Zhang and Biswas (2019), etc. have found that dealing with open-ended problems have a positive influence in certain students' skills like problem solving and creativity.

Computer programming offers many tools, of which only one or a few are optimal to use in a given situation. For example, there are two types of loops – the counting loop (`repeat`) and the conditional loop (`while`). The former should be used when the number of repetitions is know in advance, the latter in a situation when something needs to be repeated while some condition is satisfied. There are conditional statements (`if-else`) that students use to design general algorithms that are applicable to a variety of situations as opposed to a single scenario. In this way, they learn how to generalize and think abstractly. Let us give an example:

It is Halloween! Karel needs to pick up three chocolate eyeballs and enter his home square (Figure 10).

Figure 10 Karel is collecting chocolate eyeballs

The solution of this problem admits at least 6 levels of programming know-how (see the Appendix):

- using just elementary commands without any programming logic
- taking advantage of a repeating pattern
- taking advantage of nested repeating patterns:
- using an `if-else` statement to reduce the code to a single nested loop
- using an `if-else` statement to reduce the code to a single loop
- a more general procedure, that takes advantage of the `while` loop.

9. COMMON CORE'S MATHEMATICS PRACTICE STANDARD #6

"Attend to precision."

Mathematics, like other subjects, involves precision and exact answers. When speaking and problem-solving in mathematics, exactness and attention to detail is important because a misstep or inaccurate answer in mathematics can be translated to affect greater problem-solving in the real world. The importance in this step comes in the speaking demeanour of students to explain what is understood and what is not.

Computer programming forces students to attend to precision. Of course, the logic of the algorithm must be precise, or the program will not work and the given problem will not be solved. But also the syntax - in other words avoiding typos and being able to comply with simple formatting rules - teaches students to attend to precision.

To illustrate this, we will use the last program from the previous section:

```
while not home
  if eye
```

```
    get
  go
```

The following program will not work due to a typo. Can you find it?

```
while not home
  if eve
    get
  go
```

Neither will the following program, due to a mistake in formatting (the second, third and fourth lines aren't correctly indented):

```
while not home
if eye
  get
go
```

The last example contains a formatting mistake which does not cause the program to be invalid, but it changes its logic completely - the program becomes an infinite loop and the robot never makes a single step forward:

```
while not home
  if eye
    get
    go
```

10. COMMON CORE'S MATHEMATICS PRACTICE STANDARD #7

"Look for and make use of structure."

When students can identify different strategies for problem solving, they can use many different skills to determine the answer. Identifying similar patterns in mathematics can be used to solve problems that are out of their learning comfort zone. Repeated reasoning helps bring structure to more complex problems that might be able to be solved using multiple tools when the problem is broken apart into separate parts. Subdividing a problem into subproblems is a successful strategy applied since the very beginning of teaching mathematics with technology (Roman, 1974) to the present times (Saritepeci, 2019; Zhang and Biswas, 2017).

Looking for patterns and making use of structure is a fundamental component of computer programming. Students learn quickly that finding a pattern simplifies the logic, and makes the solution of the given problem easier. Often this is realized by means of custom commands. Custom command is nothing else than a "small program" that is used to solve a "small task" inside of the "bigger task". As a result, the bigger task becomes simpler, and moreover, the custom command can be used to solve the "small problem" in a different context. Let us illustrate this on an example.

Karel is in his cellar. He needs to go through all four aisles, and move all objects to the opposite shelf across the isle (Figure 11).

Figure 11 Karel is rearranging things in his cellar

Here, a repeating pattern is to proceed one step forward, move one object across the aisle, return into the aisle, and turn right. Let us create a custom command `oneobject` for that:

```
def oneobject
  go
  left
  go
  get
  repeat 2
    right
  repeat 2
    go
  put
  repeat 2
    left
  go
  right
```

But this is not all. Another (higher-level) repeating pattern is to go through one aisle and move all nine objects from the left shelf to the right shelf. Let us create a custom command `oneaisle` for that:

```
def oneaisle
  repeat 9
    oneobject
  go
```

With these two custom commands in hand, the task becomes much simpler. Can you do it? The solution is presented in the Appendix.

11. COMMON CORE'S MATHEMATICS PRACTICE STANDARD #8

"Look for and express regularity in repeated reasoning."

In mathematics, it is easy to forget the big picture while working on the details of the problem. In order for students to understand how a problem can be applied to other problems, they should work on applying their mathematical reasoning to various situations and problems. If a student can solve one problem the way it was taught, it is important that they also can relay that problem-solving technique to other problems.

This standard makes sure that students can generalize their thinking, and use what they already know, to solve a more complicated problem. Let us illustrate this on an example.

Karel is in a diamond mine. His task is to move all gems on the marks and enter his home square (Figure 12).

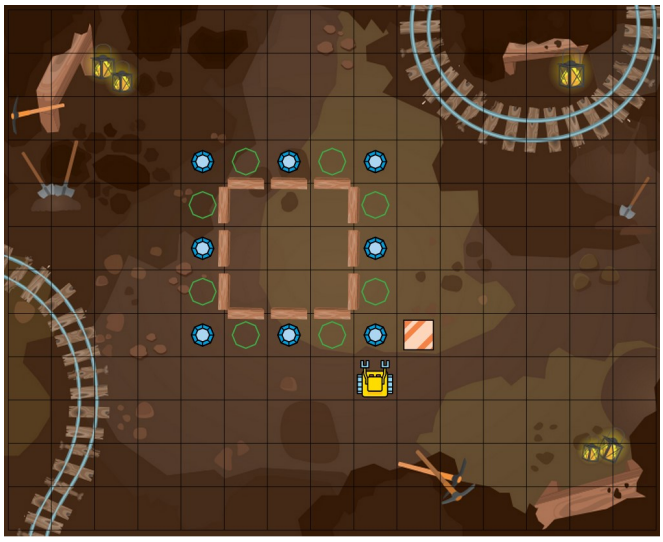

Figure 12 Karel is in a diamond mine

This time we will not discuss how the problem is solved because we have a different objective in mind. The solution program is:

```
go
repeat 4
  repeat 2
    get
    go
    put
    go
  left
right
go
```

Now let us make the problem harder, and see if the students can generalize what they did before to solve it. We will install additional walls in between the gems and the marks (Figure 13).

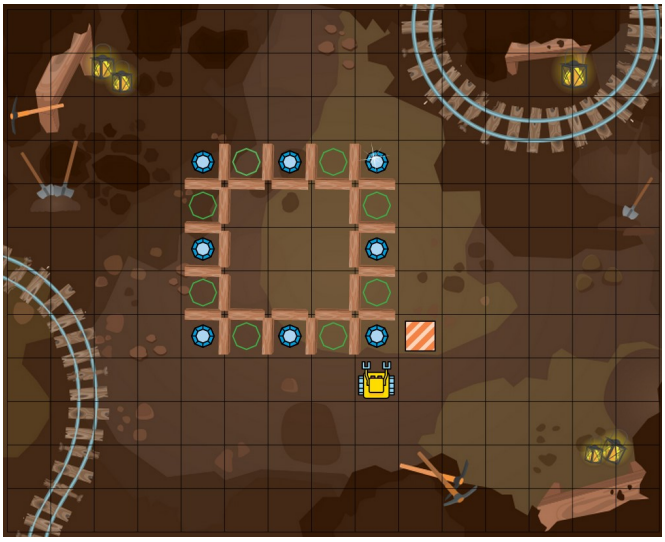

Figure 13 Karel is in a diamond mine problem made harder

The key here is to realize that with the exception of the very first and very last `go` command, every other go command would mean to hit a wall. So all one needs to do is replace these `go` commands (on lines 5 and 7) with a custom command `goaround` to make the robot go around the new walls. The complete solution then looks as follows:

```
def goaround
  right
  go
  repeat 2
    left
    go
  right

# Main program:

go
repeat 4
  repeat 2
    get
    goaround
    put
    goaround
  left
right
go
```

Here one can see clearly that almost the entire logic from the solution of the simpler problem was preserved. Only minor adjustments were needed to solve the more complicated problem.

12. CONCLUSION

We have discussed the Common Core's eight Mathematical Practice Standards and showed on examples how each of them can be addressed using computer programming. In our experience, computer programming actually provides more engaging ways to teach these

standards to students. We also introduced the educational programming language Karel the Robot, and a free public cloud computing platform NCLab (http://nclab.com/apps/) where Karel the Robot is available online for instant use. The authors have been using the methods and examples described in this paper since 2010 while training teachers and working with students, while organizing numerous spring / summer / fall and winter camps as well as after school programs.

APPENDIX: SOLUTION TO SELECTED PROBLEMS

Solution to the problem from Section 2:

```
repeat 3
  go
  get
  go
  go
  put
  go
```

Alternative solutions to the problem in section 8:

Program #1 uses just elementary commands without any programming logic:

```
go
go
go
get
go
go
go
get
go
go
go
get
go
go
go
```

Program #2 takes advantage of a repeating pattern, but it fails to realize that there is one additional repeating pattern:

```
repeat 3
  go
  go
  go
  get
go
go
go
```

Program #3 takes advantage of two levels of repeating patterns:

```
repeat 3
  repeat 3
    go
  get
repeat 3
  go
```

Program #4 uses an `if-else` statement to reduce the code to a single nested loop:

```
repeat 4
  repeat 3
    go
  if eye
    get
```

Program #5 uses an `if-else` statement to reduce the code to a single loop. One extra advantage of this approach is that the eyeballs can be anywhere in the row connecting the robot with the home square:

```
repeat 12
  if eye
    get
  go
```

Last, Program #6 also works for randomly distributed objects, but moreover it does not require the home square to be exactly 12 steps away from the robot – it is more general than Program #5:

```
while not home
  if eye
    get
  go
```

Solution to the problem from Section 10:

```
repeat 2
  oneaisle
  go
  right
  repeat 3
    go
  right
  oneaisle
  go
  if not home
    left
    repeat 3
      go
    left
```

REFERENCES

Abbelson, H. and diSessa, A. (1981). *Turtle Geometry. The Computer as a Medium for Exploring Mathematics*. The MIT Press, Cambridge, MA.

Alman, A. (2017). The Influence of Open-Ended and STAD Method on the Mathematical Problem-Solving Skills in Terms of Learning Achievement. *Jurnal Prima Edukasia, 5*(2), 112-124. DOI: 10.21831/jpe.v5i2.14280.

Budinski, N. (2017). An Example how Geogebra can be Used as a Tool for STEM. *The International Journal for Technology in Mathematics Education, 24*(3), 149-153. DOI: 10.1564/tme_v24.3.07.

Capraro, M. M., An, S. A., Ma, T., Rangel-Chavez, A. F. and Harbaugh, A. (2012). An investigation of preservice teachers' use of guess and check in solving a semi open-ended mathematics problem. *Journal of Mathematical Behavio*r, 31, 105– 116. DOI: 10.1016/j.jmathb.2011.10.002.

Cha, S. E., Kwon, D. Y. and Lee, W. G. (2007). Using Puzzles: Problem-Solving and Abstraction. In: *Proceedings of the 8th ACM SIGITE conference on Information Technology Education*. ACM, New York (pp. 135-140).

Common Core. Standards. Standards for Mathematical Practice (2019). URL: http://www.corestandards.org/Math/Practice/. Accessed on December 4, 2019.

DreamBox Learning (2013). Explaining CCSS Standards for Mathematical Practice URL: http://www.dreambox.com/blog/explaining-ccss-standards-for-mathematical-practice. Accessed on December 4, 2019.

Everettte, M. (2013). A Guide to the 8 Mathematical Practice Standards. URL: https://www.scholastic.com/teachers/blog-posts/meghan-everette/guide-8-mathematical-practice-standards/. Accessed on December 4, 2019.

FMSLogo (n. a.) Welcome to the world of FMSLogo! URL: http://fmslogo.sourceforge.net/ Accessed on December 4, 2019.

Fujita, T., Doney, J. and Wegerif, R. (2019). Students' collaborative decision-making processes in defining and classifying quadrilaterals: a semiotic/dialogic approach. *Educational Studies in Mathematics*, 101, 341–356. DOI: 10.1007/s10649-019-09892-9.

Graphics.X11.Turtle (n.a.). URL: https://hackage.haskell.org/package/xturtle-0.2.0.0/docs/Graphics-X11-Turtle.html. Accessed on December 4, 2019.

GNU Octave (2019). GNU Octave Scientific Programming Language. URL: https://www.gnu.org/software/octave/. Accessed on December 4, 2019.

Haas, G. M. (2016). BFOIT. Introduction to Computer Programming. Java Turtle Graphics. URL: http://guyhaas.com/bfoit/itp/JavaTurtleGraphics.htm. Accessed on December 4, 2019.

Harvey, B. (2008). Berkeley Logo (UCBLogo). URL: https://people.eecs.berkeley.edu/~bh/logo.html. Accessed on Accessed on December 4, 2019.

Kovács, Z., Recio, T. and Vélez, M. P. (2018). Using Automated Reasoning Tools in GeoGebra in the Teaching and Learning of Proving in Geometry. *The International Journal for Technology in Mathematics Education 25*(2). 33-50. DOI:10.1564/tme_v25.2.03.

The LaTeX Project (2019). LateX – A document preparation system. URL: https://www.latex-project.org/. Accessed on December 4, 2019.

Logo (2019). URL: https://en.wikipedia.org/wiki/Logo_(programming_language) Accessed on December 4, 2019.

Logo Tree (2019). https://pavel.it.fmi.uni-sofia.bg/logotree/. Accessed on December 4, 2019.

NCLab (2019). Create and share your projects in Computer Programming, 3D Modeling and more. URL: http://nclab.com/free-portal/. Accessed on December 4, 2019.

OpenSCAD (2019). The Programmers Solid 3D CAD Modeller. URL: http://www.openscad.org/. Accessed on December 4, 2019.

Pattis, R. E. (1995). *A Gentle Introduction to the Art of Programming*, 2nd Edition, J. Wiley and Sons, 1995.

PLaSM (2019) Programming Language of Solid Modeling. URL: https://en.wikipedia.org/wiki/PLaSM. Accessed on December 4, 2019.

Pro-Bot (n. a.) URL: https://www.terrapinlogo.com/probot.html. Accessed on December 4, 2019.

R (2019). The R Project for Statistical Computing. URL: https://www.r-project.org/. Accessed on December 4, 2019.

Rachum, R. (2015). Python Turtle. URL: http://pythonturtle.org/. Accessed on December 4, 2019.

Roanes-Lozano, E. and Roanes-Macías, E. (1994a). *Nuevas Tecnologías en Geometría.* Editorial Complutense, Madrid.

Roanes-Lozano, E. and Roanes-Macías, E. (1994b). An Implementation of "Turtle Graphics" in Maple V. *Maple Technical Newsletter 1994 Special Issue*, 82-85.

Roman, R. A. (1974). Teaching Problem Solving and Mathematics By Computer: An Interim Report. National Inst. of Education (DHEW). National Science Foundation, Washington D.C. Report No. PU-LRDC-1974-15. Available from: https://files.eric.ed.gov/fulltext/ED101690.pdf. . Accessed on December 4, 2019.

Rosa, P. and Petrášková, V. (2017). Potential of Maple as a tool for improving financial education of future teachers. *International Journal for Technology in Mathematics Education, 24*(3), 161-166. DOI: 10.1564/tme_v24.3.09

Saritepeci, M. (2019). Developing Computational Thinking Skills of High School Students: Design-Based Learning Activities and Programming Tasks. *Asia-Pacific Education*, 1-20. DOI: 10.1007/s40299-019-00480-2.

Shaheed Hartley, M. and Treagust, D. F. (2014). Learner perceptions of the introduction of computer assisted learning in mathematics at a peri-urban school in South Africa. *Learning Environments Research*., *17*(1), 95–111. DOI 10.1007/s10984-014-9157-y

SciPy.org / NumPy (2019). Python-Based Libraries for Scientific Computing. URL: https://www.scipy.org/. Accessed on December 4, 2019.

Scratch (n. a.). Get Creative with Coding! URL: https://scratch.mit.edu/. Accessed on December 4, 2019.

Solin, P. (2017). Bringing More STEAM to Mathematics Education. *International Journal for Technology in Mathematics Education, 24*(4), 191-198

SymPy (2019). URL: http://www.sympy.org/. Accessed on December 4, 2019.

TeacherStep (2015). Breaking Down the Common Core's 8 Mathematical Practice Standards. URL: https://www.teacherstep.com/breaking-down-the-common-cores-8-mathematical-practice-standards/. Accessed on December 4, 2019.

Vera Ruiz, R. (2011). Teaching the Relevance of Mathematics in Information Technologies through Functional Programming in Secondary School. *International Journal for Technology in Mathematics Education, 18*(3), 155–161, 2011.

Wilensky, U. (2019). NetLogo. The NetLogo 6.1.1 User Manual. URL: http://ccl.northwestern.edu/netlogo/docs/NetLogo%20User%20Manual.pdf. Accessed on December 4, 2019.

Yazgan-Sağ, G. and Emre-Akdoğan, E. (2016). Creativity from Two Perspectives: Prospective Mathematics Teachers and Mathematician. *Australian Journal of Teacher Education, 41*(12), 25-40. DOI: 10.14221/ajte.2016v41n12.3

Zhang, N. and Biswas, G. (2017). Assessing Students' Computational Thinking in a Learning by Modeling Environment. In: Kong, S. C., Sheldon, J. and Li, K. Y. (Eds.) Conference Proceedings of International Conference on Computational Thinking Education 2017. The Education University of Hong Kong, Hong Kong (pp. 11-16). Available from: https://www.eduhk.hk/cte2017/doc/CTE2017%20Proceedings.pdf. Accessed on December 4, 2019.

Zhang N. and Biswas G. (2019). Defining and Assessing Students' Computational Thinking in a Learning by Modeling Environment. In: Kong S. C. and Abelson H. (eds) Computational Thinking Education. Springer, Singapore (pp. 203-221). DOI: 10.1007/978-981-13-6528-7_12

BIOGRAPHICAL NOTES

Dr. Pavel Solin is full professor of applied and computational mathematics at the University of Nevada, Reno. Besides his research in numerical methods for multiphysics systems described by systems of partial differential equations, his passion is K-12 outreach. He is Founder and Director of NCLab (http://nclab.com). Through NCLab he is providing K-12 schools with easy access to mathematics, computing, computer programming, 3D modeling, and other critical 21st century STEM skills. Dr. Solin is actively working with hundreds of K-12 schools in the U.S. and other countries.

Dr. Eugenio Roanes-Lozano is full professor in the Departamento de Didáctica de las Ciencias Experimentales, Sociales y Matemáticas of the Universidad Complutense de Madrid. His research interests are in applications of mathematical software to education, artificial intelligence and transportation engineering. He has a PhD in mathematics and another PhD in computer science.

