

UNIVERSITY OF TWENTE.

# Computational Thinking in Primary School: An Examination of Abstraction and Decomposition for Different Ages

Author: Wouter J. Rijke

University of Twente

Department of Instructional Technology

P.O. Box 217, 7500 AE Enschede

The Netherlands

w.j.rijke@gmail.com

**Abstract** Computational thinking is a term used to describe the thought process of formulating problems and their solutions in a way that can be carried out by a computer. Despite a growing effort to implement computational thinking skills in primary schools, little research has been conducted about what skills to teach at what age. This is a problem for teachers working in primary education, wanting to teach computational thinking skills. The research questions that guide this study are as follows: (1) *How is age influencing the students' success in tasks related to computational thinking?* (2) *How difficult are lessons about computational thinking perceived by students?* (3) *What are the students' perceptions of their learning experiences?* 210 students between the age of 6 and 12 participated in this study, all of whom enrolled in a primary school. Lessons from the Barefoot Computing project are used as an introduction into two computational thinking subjects: abstraction and decomposition. The first main finding concerns the relation between age and the discussed computational thinking skills; abstraction and decomposition. Second, an interaction is found between gender and the abstraction task. Third, for both tasks, there are no significant differences between age groups on perceived difficulty, cognitive load, and flow. Implications and directions for future research are discussed.

**Supervisors:** Dr. L. Bollen (1<sup>st</sup>)  
Dr. T. H. S. Eysink (2<sup>nd</sup>)  
Dr. J. L. J. Tolboom (SLO)

**Date:** February 17<sup>th</sup>, 2017

## **Keywords**

Computational Thinking, Abstraction, Decomposition, Age, Perceived Difficulty, Flow, Cognitive Load, Programming, Primary Education

## Introduction

In the Netherlands, there is increasingly more attention for computational thinking as one of the 21<sup>st</sup> century skills (Thijs, Fisser, & Hoeven, 2014). Computational thinking is a term first used by Wing (2006), and was used to describe the thought process of formulating problems and their solutions in a way that can be carried out by a computer. It is now more and more argued by educators that computational thinking should be picked up not only by computer scientists, but by everyone: it can make everyday activities much more efficient, and might create a better understanding of today's pervasive usage of computers and software (Lee, Mauriello, Ahn, & Bederson, 2014). Up until recently, there has been little attention for teaching computational thinking skills in Dutch primary education. This is in contrast with countries around the world, like the United Kingdom, where computational thinking is implemented in primary school curricula. A growing effort is put in developing curricula for children that foster computational thinking: computer scientists are creating games (Lee et al., 2014), instruction guides for overlapping courses like math (Ministry of Education, 2003), and trying to get girls excited for the Science, Technology, Engineering and Math (STEM)-fields as well (Corbett & Hill, 2015; Modi, Schoenberg, & Salmond, 2012). More practical applications include physical programmable robots like Dash & Dot, MiP and Sphero. Programming is thought of to be closely related to computational thinking skills.

Despite a growing effort to implement computational thinking skills in primary schools, there are still no clear guidelines on the appropriate age to implement computational thinking skills in primary school. This is a problem for teachers working in primary education, wanting to teach computational thinking skills. To increase the quality of Dutch programming education, it is important to know from what age lessons about computational thinking skills are appropriate. This will result in more evidence based curricula, which are more likely to find a permanent place in primary education. Seeing the scale of interest in implementing computational thinking skills in primary education, it will be of great relevance to also research how lessons about computational thinking are perceived by children. The research questions that guide this study are as follows: *(1) How is age influencing the students' success in tasks related to computational thinking? (2) How difficult are lessons about computational thinking perceived by students? (3) What are the students' perceptions of their learning experiences?*

In this study, the following definition of computational thinking is used (Selby, 2013):

Computational thinking is an activity, often product oriented, associated with, but not limited to, problem solving. It is a cognitive or thought process that reflects the ability to think in abstractions, the ability to think in terms of decomposition, the ability to think algorithmically, the ability to think in terms of evaluation, and the ability to think in generalizations. (p. 5)

There is some research on computational thinking for younger students, like Duncan and Bell (2015), who evaluated the teaching of two programming topics (i.e., programming and data representation) with twelve-year-olds. They found that teaching computational thinking alone is likely to be difficult without combining the lessons with computer science. However, they did not address lower age groups, and mention that existing curricula differ over which key topics to introduce at what age. This is mainly because programming contains demanding concepts to grasp, like abstraction. Even for most adults it takes time to apprehend and employ some of these concepts in practice. Research on teaching programming to learners who have no prior programming experience, has shown that the programming concepts and theories are perceived very difficult, and in result programming courses often have a lot of dropouts (Robins, A., Rountree, J. and Rountree, 2003; Stachel et al., 2013). Students from these studies have difficulties understanding the abstract concepts that are encountered in programming.

One of the cornerstones of computational thinking is (problem) decomposition. Decomposition is used when a problem is too big or complex to solve at once. It is regarded as both a computational thinking skill as well as a problem solving skill (Polya, 1945; Wing, 2006). It is defined as breaking down a problem into smaller, more easily solved, parts (Selby, 2015). When planning on carrying out a large problem with a computer, decomposition becomes essential for the skill of programming. When programming, a code is produced to carry out the command of the programmer. This code should be carried out by a computer step by step – called an algorithm (step by step instructions to get something done). For decomposition, it was found that it is teachable for 5 to 6 year olds in an arithmetic setting, when students are still developing their number sense (Cheng, 2012). Decomposition is found to be the most difficult computational thinking skill to master (Selby, 2015). Selby found that one of the reasons for this, was that sometimes the problem to solve isn't completely understood. Also, students appeared to understand the concept of decomposition, but struggled to implement the process in new situations.

Another explanation for decomposition to be so difficult can be derived from Jean Piaget (1972). His theory of cognitive development distinguishes stages of development of children to adulthood, where from 13 years and older, children start using abstract and theoretical reasoning. Piaget states that in the preceding stages children are not finished forming schemas (storage and organization structures) in their brains, which makes it more difficult to understand the world around them until puberty (Myers, 2007). It would make sense that engaging in a new skill such as problem decomposition would give difficulties. According to Selby (2015), a way to make decomposition easier is to use problems to decompose students fully understand. Also, asking if they did similar exercises before (previous experience) could give insights on how difficult they experience decomposing a problem. Both these factors should be examined. Selby states that understanding decomposition is a necessary condition before assessing the other cornerstones (i.e., abstraction, algorithm design, evaluation, and generalizations).

Problem decomposition might be regarded as the most *difficult* computational thinking skill to master, abstraction is seen by various experts as the most *important* computational thinking skill (Hazzan, 1999; Kramer, 2007; Se, Ashwini, Chandran, & Soman, 2015; Wing, 2006). It is stated that “the abstraction process – deciding what details we need to highlight and what details we can ignore – underlies computational thinking” (Wing, 2008, p. 3718). It is advocated to start teaching abstraction as early as possible, as it is one of the most fundamental ideas in computer science (Statter & Armoni, 2016). In computational thinking, abstract thinking is essential for recognizing similar conditions in different problems, which is essential for transferring lines of reasoning to other situations (Nickerson, Brand, & Repenning, 2015). You see those abstractions back in simulations and models, made to teach about concepts like gravity, physics or even history. Although decomposition is reported to be the most difficult computational thinking skill, abstraction should be challenging as well. Continuing Piaget’s theory, children should develop through the first stages of cognitive development before they can be expected to use abstraction for programming purposes. However, according to Kuhn, Langer, Kohlberg and Haan (1977), only around 35% of adolescents have achieved the abstract reasoning skills linked to the last (i.e., formal operational) stage. Some adults never even achieve that stage of reasoning. According to Piaget, the cognitive development of abstract reasoning goes hand-in-hand with the biological maturation of the brain (Lister, 2011). This would mean that children would consistently show the same level of abstract reasoning on very different problems – something we now know is not the case (Smith, 1992). According to neo-Piagetian theory, people still go through the developmental stages. They are, however, thought to develop through abstract forms of reasoning, regardless of their age (Lister, 2011). This means that people can become an expert in a particular subject (e.g., math), and still be a novice in another subject (e.g., geography), and thus show different levels of abstract reasoning depending on the subject. This implies that age shouldn’t *have* to play a role in reaching a form of abstract reasoning in a particular subject. Seeing how young children can exceed in forms of abstract reasoning (e.g., when practicing art or music), it seems clear that not only age, but (lack of) experience limits children’s level of abstract reasoning. When students are still novices, they should have the same skill in abstract reasoning as their younger or older peers.

Now that is determined that primary school students should be able to learn abstraction and decomposition, it is important to look at the most efficient way to do this. Instructional methods should try to optimize people’s limited cognitive processing capacity, in order to increase their ability to transfer knowledge and skills to new situations (Mayer & Moreno, 2010). For optimizing people’s cognitive capacity, we look at cognitive load theory, which divides cognitive capacity into three, co-dependent parts (Sweller, 1994). One part is occupied with processing information and distractions not directly related to the learning content, such as the design of the instructional material. This part, called extraneous load,

should be reduced as much as possible. In a computational thinking lesson, this could be a small font size or redundant media. Then, a fixed part deals with the inherent complexity (the number and way of interacting elements) of the content. This is referred to as intrinsic load. In decomposition, these could be the steps a problem can be decomposed in. Finally, the remaining *germane* load is used during schema construction and knowledge building. The better the course design, the more cognitive space frees up for this part. Although cognitive load theory gives a direction for instructional design principles, De Jong (2010) poses some critical questions. He questions the falsifiability of the theory, because “every outcome fits within the theory post-hoc” (p. 125). He also criticizes the cognitive load measurements, which fail to distinguish all three elements, and recognizes *cognitive overload*. Even though these points are valid, cognitive load is still a concept worth exploring when looking if abstraction and decomposition are appropriate for primary school students. Young students could feel overwhelmed by either the complexity of the content, or the design of the material. This could result in students halting their efforts to make sense of the material, and therefore ceasing the learning process. How cognitive load should be described can be debated, but the presence seems very real.

How appropriate abstraction and decomposition are for primary school students, is also linked to how difficult young students perceive these concepts. More difficult assignments trigger a more slow, analytical processing of information (Alter, Oppenheimer, Epley, & Eyre, 2007). A more difficult text, for example, seems to work as a warning sign for the brain to use more analytical ways of thinking, instead of intuitive reasoning. This means that when students experience a concept like abstraction or decomposition to be difficult, it does not necessarily result in inferior learning. When students feel capable and challenged, a state of *flow* could occur. Flow describes a state of mind in which a person is not *over* challenged, nor *under* challenged, and is completely engaged in the task (Csikszentmihalyi, 1975). During flow, the balance between skill and challenge is very sensitive; it can easily be disrupted (Wang & Degol, 2013). Both the skill level and the complexity of the task should increase simultaneously in order to maintain a level of flow. When that is not the case, students might be disinterested when engaging in an (obligatory) task. Or, on the other end of the spectrum, the course material could be perceived boring, monotonous or unexciting. Measuring flow will give an indication if abstraction and decomposition are appropriate for young students; the more flow is experienced, the less cognitive overload or boredom is to be expected.

The literature about computational thinking, cognitive development, perceived difficulty and flow give enough insights to formulate hypotheses on what to expect when studying these concepts.

## **The current study**

In the current study, lessons from the Barefoot Computing project<sup>1</sup> are used as an introduction into two computational thinking subjects: abstraction and decomposition. These lessons are given to students from two different primary schools in the Netherlands, with students as young as six and as old as twelve years old. After the lesson, students are asked if they experienced a state of flow, how much cognitive (over)load was experienced, and to rate their learning experience, like the amount of difficulty they had with the lessons. In addition, their success in the task related to abstraction and decomposition is measured.

Based on the primary school students' maturing brain and likeliness of previous experience in the subject, age is anticipated to correlate with students' success in decomposition or abstraction. Also, it is expected that age is positively correlated with flow, and negatively correlated with students' perceived difficulty and cognitive load, meaning that when students get older, they will find the tasks easier. Furthermore, decomposition is thought to be more difficult for students than abstraction. For decomposition in particular, this could result in an above average score on perceived difficulty, cognitive load, and a negative correlation of these two constructs with flow. Finally, although the perceived difficulty is expected to be high, the perceptions of students are thought of to be positive. This is based on the fact that the lessons provide something new, are challenging and get the students out of their daily routine. The older students get, the more likely it is that they already did something similar, so it is expected that the expected positive attitude wears off when students get older.

## **Method**

### **Participants**

In this study, 210 students between the age of 6 and 12 participated ( $M = 9.08$ ,  $SD = 1.92$ ), all of whom enrolled in a primary school. The participants were collected from two different schools, one very large school located in Vleuten, and an average sized school in Nijmegen. There was a total of nine teachers who volunteered to participate in the study with their class, which lead to a total of 210 students varying from 1<sup>st</sup> to 6<sup>th</sup> grade. There were two 6<sup>th</sup> grade classes, no 5<sup>th</sup> grade class, for the abstraction lesson there was no 3<sup>rd</sup> grade and for the decomposition lesson there was no 2<sup>nd</sup> grade. The nine classes were then randomly assigned to one of the two conditions, where students remained in their own class. This resulted in the distribution displayed in Table 1.

---

<sup>1</sup> In the UK, primary school teachers are supported in various ways to implement computational thinking skills. The Barefoot Computing project is one of those initiatives. On [barefootcas.org.uk](http://barefootcas.org.uk), teachers can download and co-operate in exemplar primary computing resources. The project provides workshops for teachers, with the aim to deliver new computing subjects with confidence, in addition to exemplar teaching activities. These activities focus on showing how computing and other subjects (e.g. maths, English and science) can be combined.

Table 1. *Participant Distribution per Condition and Grade*

Condition	Grade (Dutch: groep)						Total
	1 (3)	2 (4)	3 (5)	4 (6)	5 (7)	6 (8)	
Abstraction	28	23	-	29	-	16	121
Decomposition	21	-	26	27	-	15	89
Total	49	23	26	56	-	25	210

Among the participants, the number of boys ( $n = 104$ ) and girls ( $n = 106$ ) were about equal. Students were classified as young (everyone younger than 8 years old), middle (those between 8 and 10 years old), and old (everyone older than 10 years old). The cut-off points for the age groups are based on the ages on which children are admitted in grades in primary school in the Netherlands (Onderwijsraad, 2005). All lessons were given by the same researcher, to control for the influence of the teacher on the lessons. The lessons were given in the students' own classroom, with the presence of their own teacher. Since the participants were under-age, the parents were contacted for their permission for participation. Passive consent was used to collect their permission.

## Materials

For the lessons, several materials were used. All materials were piloted. For this research, two unplugged (without computers) lessons were used as experimental manipulations (Appendices A and B). The lessons were translated from English to Dutch by the researcher. The researcher hereby used the steps provided by the World Health Organization to achieve a Dutch version of the lessons that are conceptually equal to the original. These steps comprise forward translation (to Dutch), expert panel back-translation (with supervisors), and pre-testing (i.e., a pilot) before finalizing the eventual lessons.

### *Abstraction task*

In the abstraction task, the students paired up and received a deck of sixteen cards, containing words to portray, like school, shoes, and dinosaur. For example, when a student had to portray the word 'school', they had to abstract the most important details of the concept. This means they could sketch a large building, children, or materials like books and pens. This way they learned ignoring unimportant details and only including what is most important, and in doing so were abstracting. When the starting sign from the teacher was given, the first student drew as many sketches as possible for his or her partner, within a short, pre-determined amount of time. These set times are described in the procedure section. When time was expired, students switched in their role of drawer (which performance is measured) and guesser.

### *Guess What Cards*

A total of sixteen blank cards with one word written on it were used for the main activity of the *abstraction* lesson; guessing what sketches drawn by a partner represented. Students from the first grade (~ age 6) were assisted by the researcher and teacher when they could not read the word. Because the students asking for assistance took more time, the results from this grade should be carefully interpreted. It was planned to use printed *Guess What?* cards, provided in the slides from the original abstraction lesson. These cards, however, were accidentally not brought to the school location on the first data collection. With no available printer on such short notice, handwritten cards were crafted to use in the lesson. The cards were written carefully and neatly, to ensure they were readable. To retain the controllability of the used materials, these handwritten cards were used for all abstraction lessons.

### *Blank sketch sheet*

In the abstraction lesson, students were guessing what sketches represent. To capture their produced sketches (in this study further referred to as *artefacts*), a blank sketch sheet was provided to every student pair. The students each drew on one side of the sheet.

### *Decomposition task*

In the decomposition task, students created hand clapping, hand tutting, or hand jive sequences of movements. Students broke the sequence of actions down into parts and in so doing were decomposing. In a discussion, students were encouraged to link this idea to breaking problems down when creating computer programs such as animations or games. The students paired up and received a design sheet to document their movements. They had a limited, pre-determined amount of time to come up with a decomposed dance. They were instructed to make as many decompositions as possible, and to make it clear enough that another pair from their class could perform their dance with only their design sheet. For example, students could decompose a dance like the hand jive performed in the movie *Grease* (1978). The students were instructed to decompose difficult dance moves, and possibly add iterations when a dance move (e.g., ‘hand clap’) should occur more than once.

### *Decomposition design sheet*

For the *decomposition* lesson, a design sheet was used. Two columns, called ‘part’ and ‘drawing’, structured their recordings of their hand clap, tut, or jive sequence. The additional instruction on the sheet supported students in evaluating their sequence; “Are the parts in the right order? Have you got all the parts? How do you start?”.



### *Lesson presentation*

A set of eight (abstraction) and nine (decomposition) slides were used in a supporting role. These slides contained examples, learning goals, and discussion topics, like how these skills are used when working with computers. The materials used in the lesson were displayed on the slides, as well as the discussion topics when discussing the role of abstraction and decomposition in programming.

### **Measurements**

The constructs perceived difficulty, flow, cognitive load, and students' perceptions were measured. The questionnaire used in the current study can be found in Appendix C.

#### *Perceived difficulty*

Perceived difficulty was measured by asking students to rate the lesson on three items about difficulty, length, and clarity on a four-point Likert scale, with a higher number representing more perceived difficulty. The questions were translated from the Perceived Difficulty Assessment Questionnaire, or PDAQ (Ribiero & Yarnal, 2010). Although the items about difficulty, length, and clarity are related, they are more valuable to be interpreted separately than to combine into one construct.

#### *Flow*

Flow was measured by items from the translated Flow Short Scale (Rheinberg, Vollmeyer, & Engeser, in Eysink et al., 2015), consisting of nine items about how students experienced the activity. Students rated the questions on a five-point Likert scale, with a higher number representing more flow. This is adapted from the original, where a seven-point Likert scale was used and a higher number represented less flow. An example of one of the nine items was "I had the feeling that I had everything under control". The measured reliability was measured at Cronbach's  $\alpha = .68$  for young students,  $\alpha = .78$  for middle-aged students, and  $\alpha = .86$  for the oldest students. The nine flow items (with five answer possibilities) were then combined into one variable, with a number between 9 (no flow) and 45 (maximum amount of flow).

#### *Cognitive load*

Cognitive load was measured by an adapted and translated version of the NASA Task Load Index. The students answered the five items on a five-point Likert scale, with a higher number representing more cognitive load. An example of a question was "How hurried or rushed was the pace of the task?". These items measure mental demand, temporal demand, overall performance, frustration level, and effort. The TLX does not distinguish between the three types of cognitive load. The Cronbach's  $\alpha$  was measured too

low to use the separate items as one construct ( $\alpha = .55, .43$ , and  $.51$  for young, middle, and old students respectively).

### *Perceptions*

The perceptions of the students' learning experiences were measured by asking students to rate how much they liked the task on a five-point Likert scale, and describe how they would describe the lesson in one word. In addition, they were asked to give the lesson a grade on a scale from 1 to 10 (a way students are used to grade something). A five-point Likert scale is used for the other questions, because 10 points would be too broad of a self-reporting scale. Previous experience is measured by one item; "Did you ever do an activity like this before?", with a "yes" or "no" answer option. A follow-up question is provided when "yes" is answered; "If yes, what did you do?".

Students who participated in the problem decomposition lesson got a question about their understanding of the problem (i.e., the decomposing of the hand clap sequences). When students indicate that they did not understand the problem to decompose, their perceived difficulty rating should be interpreted differently.

## **Data analysis**

### *Abstraction task*

To measure students' success in the abstraction task, the number of cards a student could get their partner to guess was counted. This way, a student who is very good at determining the important details to sketch (and therefore presumably is good at abstraction) would have more cards guessed by their partner than someone less skilled. To compare between ages, the number of cards guessed was divided by the time on task.

### *Decomposition task*

Decomposition success was measured by the number of dance steps students created. A dance step is defined as a separate drawing of a movement, different from the previous step, and independent from the number of iterations. When decomposing their dance, more skilled students would create more steps than less skilled students. Here too, the number of decomposed steps was divided by the time on task.

## **Procedure**

At the start of the lesson, the researcher introduced himself to the students. Then, he explained the format of the lesson. First, the students participated in a 30-minute course, in which they learned either about abstraction or decomposition. Here, the lesson plans for abstraction and decomposition were followed, as

shown in Appendix A and B respectively. In this lesson, students engaged in a plenary discussion, where they were encouraged to link the practical uses of the tasks to simple computer simulations and games.

Students were given a set time for their abstraction or decomposition task, expressed in minutes. For the abstraction task, students from first grade got 5 minutes, second grade got 4 minutes, fourth grade got 3 minutes and sixth grade got 1.5 minutes. For the decomposition task, all students got 15 minutes. The nature of the decomposition task asked for more time, and during the pilot it was determined 15 minutes was suitable for all grades.

After the 30-minute lesson was finished, students were asked to fill in the questionnaire (Appendix C). All students from 1<sup>st</sup> grade ( $n = 49$ ) did not fill in the questionnaire. This was decided after the pilot run, where students had too much trouble with reading (students start reading in 1<sup>st</sup> grade), and the unfamiliarity with questionnaires. The 1<sup>st</sup> grade students' artefacts (i.e., drawings and decomposed dance steps) did get analysed. Students from the other grades were assisted by the researcher and their teacher, when they did not understand a question.

## Results

### Abstraction

#### *Success on task*

Table 2 gives mean scores and standard deviations for students in the abstraction condition on their task, divided in young, middle-aged, and older students.

Table 2. *Average Number of Cards Guessed Right per Minute in the Abstraction lesson per Age Group*

Age group	<i>n</i>	<i>M</i>	( <i>SD</i> )	95% CI
Young	43	.60	(.29)	[.51, .69]
Middle	26	1.46	(.72)	[1.16, 1.75]
Old	46	2.60	(1.53)	[2.14, 3.05]

*Note.* CI = confidence interval.

A one-way ANOVA showed that there were differences between all age groups on the abstraction task ( $F(2,112) = 40.901$ ,  $p < .001$ ,  $\eta_p^2 = .42$ ). Pairwise comparisons using the Bonferroni procedure show that all age groups performed significantly different from each other. Older students performed higher ( $M = 2.60$ ,  $SD = 1.53$ ) than middle-aged students ( $M = 1.46$ ,  $SD = .72$ ) and young students ( $M = .60$ ,  $SD = .29$ ). The difference between middle-aged and young students was significant as well. A Pearson correlation also showed a moderate positive correlation between age (before grouping) and the abstraction task ( $r =$

.66,  $p < .001$ ). This means that, on average, as students become older, their drawings from the abstraction task were guessed more.

#### *Perceived difficulty, cognitive load, and flow*

Table 3 gives mean scores and standard deviations for perceived difficulty, cognitive load, and flow on the abstraction task per age group.

Table 3. *Average Scores on Perceived Difficulty, Cognitive Load, and Flow for the Abstraction Task per Age Group*

Age groups		Perceived difficulty			Cognitive load					Flow
		Difficulty	Length	Clarity	Attention	Rush	Unsuccessful	Effort	Exhausting	
Young (n = 16)	<i>M</i>	1.77	2.41	1.65	2.88	2.31	1.94	2.62	1.75	36.07
	<i>SD</i>	.87	.95	1.11	1.03	1.45	1.18	1.15	1.24	6.49
Middle (n = 27)	<i>M</i>	1.60	2.00	1.77	2.04	2.00	2.07	2.08	1.70	33.58
	<i>SD</i>	.50	.85	.86	.94	1.18	1.47	1.16	1.17	5.58
Old (n = 48)	<i>M</i>	1.81	2.04	1.57	2.31	2.25	1.83	2.37	1.58	37.54
	<i>SD</i>	.49	.41	.62	.97	1.31	1.05	1.04	1.11	6.27

*Note.* Perceived difficulty: minimum = 1 (low perceived difficulty), maximum = 4 (high perceived difficulty), cognitive load: minimum = 1 (low cognitive load), maximum = 5 (high cognitive load), flow: minimum = 9 (low flow), maximum = 45 (high flow)

There were no differences in the perceived difficulty items for younger, middle, or older students. 75.7% of all students self-reported the abstraction task to be short, whereas 24.3% thought it lasted (too) long. After a one-way ANOVA analysis, it appeared that students who thought the lesson was too long ( $M = 1.06$ ,  $SD = 1.04$ ) scored less on the abstraction task than students who felt it was short ( $M = 1.82$ ,  $SD = 1.43$ ). This difference is significant ( $F(1, 107) = 6.148$ ,  $p = .015$ ,  $\eta_p^2 = .05$ ). 89.1% of the students thought the task was clear, compared to 10.9% who felt it was unclear. The students who thought the task was very clear, experienced more flow ( $M = 39.65$ ,  $SD = 5.28$ ) than those who felt it was very unclear ( $M = 29.25$ ,  $SD = 5.44$ ), through ANOVA analysis ( $F(3, 85) = 8.688$ ,  $p < .001$ ,  $\eta_p^2 = .24$ ). There were no significant differences between these groups in their success on the abstraction task.

As can be derived from Table 3, young students reported to find it more difficult to keep their attention during the abstraction lesson than middle or older students ( $F(2, 90) = 3.754$ ,  $SD = .03$ ,  $\eta_p^2 = .08$ ). There were no differences in the other cognitive load items for younger, middle, or older students.

8.8% of the students from the abstraction lesson found the task mentally exhausting. These students experienced less flow ( $M = 27.6$ ,  $SD = 10.67$ ) than students who did not find it mentally exhausting ( $M = 37.76$ ,  $SD = 6.26$ ), through ANOVA analysis ( $F(4, 87) = 4.103$ ,  $p = .004$ ,  $\eta_p^2 = .17$ ).

### *Perceptions*

There were no differences in the perceptions of the abstraction task for younger, middle, or older students. Students gave an average grade of 8.3 ( $SD = 1.8$ ) on a scale from 1 to 10. Students ( $n = 15$ ) who said to have done something similar before (prior experience), performed better ( $M = 2.69$ ,  $SD = 1.57$ ) on the abstraction task than those ( $n = 69$ ) who had no prior experience ( $M = 1.83$ ,  $SD = 1.32$ ,  $F(1, 82) = 4.890$ ,  $p = .03$ ,  $\eta_p^2 = .06$ ). When controlled for students with prior experience, the difference on the abstraction task between age groups is still present ( $F(2,68) = 12.355$ ,  $p < .001$ ,  $\eta_p^2 = .27$ ).

## **Decomposition**

### *Success on task*

Table 4 gives mean scores and standard deviations for students in the decomposition condition on their task, divided in young, middle-aged, and older students.

Table 4. *Average Number of Decompositions Made per Minute per Age Group*

Age group	<i>n</i>	<i>M</i>	( <i>SD</i> )	95% CI
Young	23	.44	(.17)	[.37, .52]
Middle	42	.50	(.15)	[.46, .55]
Old	20	1.01	(.39)	[.83, 1.20]

*Note.* CI = confidence interval.

A one-way ANOVA test showed that there were differences between age groups and the decomposition task ( $F(2, 82) = 40.480$ ,  $p < .001$ ,  $\eta_p^2 = .50$ ). Pairwise comparisons using the Bonferroni procedure show that only older students ( $M = 1.01$ ,  $SD = .39$ ) performed significantly different from younger ( $M = .44$ ,  $SD = .17$ ) and middle-aged students ( $M = .50$ ,  $SD = .15$ ). A Pearson correlation also showed a moderate positive correlation between age (before grouping) and the decomposition task ( $r = .67$ ,  $p < .001$ ). This means that, on average, as students become older, they decomposed their dance in more steps.

### *Perceived difficulty, cognitive load, and flow*

Table 5 gives mean scores and standard deviations for perceived difficulty, cognitive load, and flow on the

decomposition task per age group. There were no differences in the perceived difficulty, cognitive load and flow items for middle and older students.

Table 5. *Average Scores on Perceived Difficulty, Cognitive Load, and Flow for the Decomposition Task per Age Group*

		Perceived difficulty			Cognitive load					Flow
Age groups		Difficulty	Length	Clarity	Attention	Rush	Unsuccessful	Effort	Exhausting	
Middle (n = 45)	<i>M</i>	1.76	2.13	1.70	2.51	1.96	1.84	2.44	1.53	34.95
	<i>SD</i>	.57	.63	.85	1.10	1.13	1.17	1.14	.76	6.73
Old (n = 20)	<i>M</i>	2.00	2.25	1.70	2.45	2.55	1.90	2.80	1.35	32.45
	<i>SD</i>	.65	.44	.57	1.05	1.19	.72	.834	.67	6.06

*Note.* Students from the young age group did not answer these items. Perceived difficulty: minimum = 1 (low perceived difficulty), maximum = 4 (high perceived difficulty), cognitive load: minimum = 1 (low cognitive load), maximum = 5 (high cognitive load), flow: minimum = 9 (low flow), maximum = 45 (high flow)

89.6% of all students reported the task to be easy, against 10.4% who found it difficult. 74.6% of the students reported the lesson to be short, against 25.4% who found the lesson long. The students who found the lesson to be short, experienced less flow ( $M = 30.86$ ,  $SD = 7.13$ ) than students who thought it was long ( $M = 37.65$ ,  $SD = 5.18$ ), through ANOVA analysis ( $F(2, 62) = 4.111$ ,  $p = .021$ ,  $\eta_p^2 = .12$ ). When conducting a univariate analysis of variance, there appears to be an interaction effect between age and perceived length of the lesson ( $F(1, 58) = 4.976$ ,  $p = .03$ ). This means that the difference in decompositions made between different ages, is influenced by whether the lesson is perceived long or short. 16.4% of all students self-reported that they really had to concentrate during the lesson, 14.9% felt rushed, 9% did not feel successful in the task, 13.4% felt like they had to put in a lot of effort, and 1.5% found the task mentally exhausting. It was expected that students would found decomposition more difficult than abstraction. This difference was not found. Students from both conditions responded in similar proportions to the perceived difficulty questions.

### *Perceptions*

Although the decomposition lesson got a good average grade ( $M = 8.3$ ,  $SD = 1.9$ ), there is a difference between older and middle-aged students ( $F(2, 64) = 5.363$ ,  $p = .007$ ,  $\eta_p^2 = .14$ ). Pairwise comparisons using the Bonferroni procedure show that older students gave a significantly lower grade ( $M = 7.3$ ,  $SD = 1.6$ ) than middle-aged students ( $M = 8.7$ ,  $SD = 1.8$ ). 78.8% of all students reported to like the lesson, 84.8% said they understood the task, 57.6% likes to dance (a considerable part of the decomposition task),

37.9% has had (or still has) dancing lessons, and 10.4% said to have done a similar task before. There were no significant differences between these groups in their decomposition skill.

## Ancillary analyses

### Abstraction

After analysing the results that were linked to the research questions, supportive and exploratory analyses were conducted. A one-way ANOVA analysis showed that there were differences between males and females in the abstraction task ( $F(1, 113) = 4.504$ ,  $p = .036$ ,  $\eta_p^2 = .04$ ). However, there also appeared an interaction effect between gender and the three age groups ( $F(2, 109) = 7.334$ ,  $p = .001$ ). This is displayed in Figure 1a.

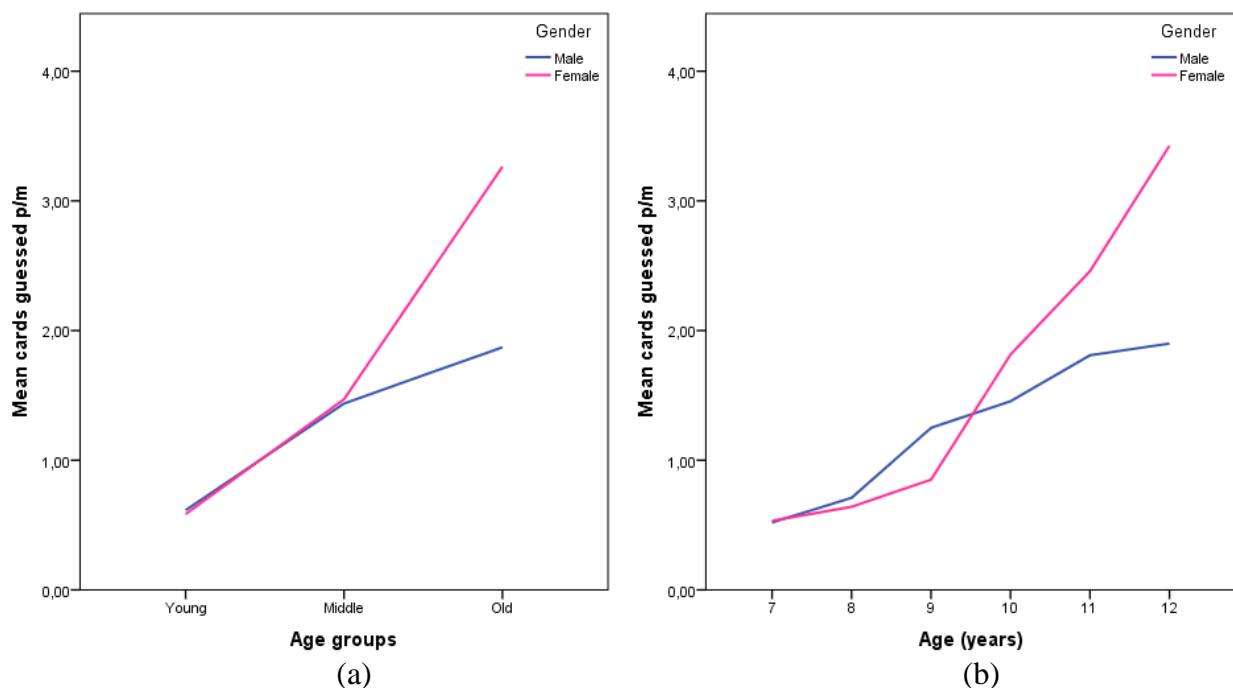


Figure 1. Interaction effect between gender and age groups (a) and age (b) on the abstraction task.

As can be seen in the graph, young males ( $M = .62$ ,  $SD = .23$ ) and females ( $M = .58$ ,  $SD = .26$ ) tend to perform basically the same on the abstraction task. This is also the case for middle-aged students ( $M = 1.44$ ,  $SD = .76$  and  $M = 1.47$ ,  $SD = .72$  respectively). But when looking at the oldest students (older than 10 years old), there is a strong difference between males ( $M = 1.87$ ,  $SD = 1.06$ ) and females ( $M = 3.26$ ,  $SD = 1.62$ ). This difference is significant ( $F(1, 44) = 11.717$ ,  $p = .001$ ,  $\eta_p^2 = .21$ ). In Figure 1b, age is divided per year, to illustrate the course of the development between males and females in the abstraction task. There is a point visible around the age of 9.5 years old, where females start to surpass males on the abstraction task. Their scores on the abstraction task start increasing more rapidly than that of males. In

addition, it was found that flow is moderately correlated with the grade students give the abstraction lesson ( $r = .657, p < .001$ ). Thus, when students experienced more flow, they rated the lesson with a higher grade.

### *Decomposition*

There is no interaction effect between gender and age groups for the decomposition task, as displayed in Figure 2a.

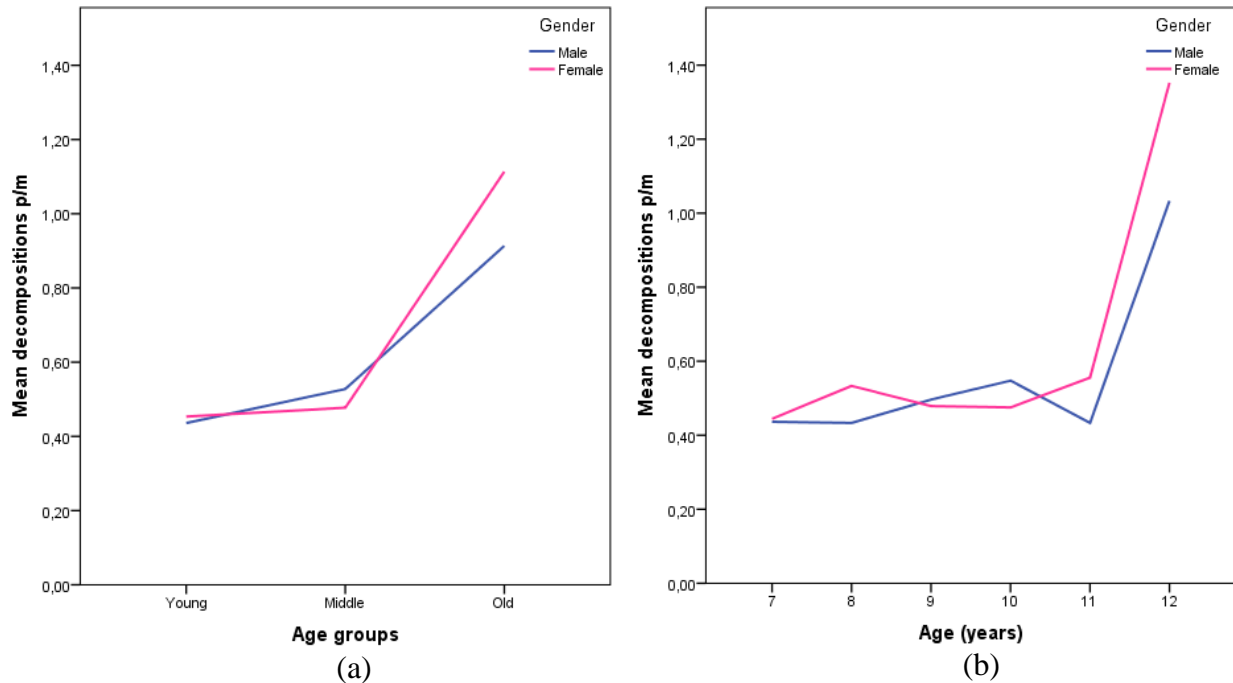


Figure 2. The difference between age groups (a) and the development between gender and age (b) for the decomposition task.

However, when dividing age per year (Figure 2b), a more refined picture of the development becomes visible. When performing a univariate analysis of variance, an interaction effect between gender and age is found ( $F(5, 73) = 2.501, p = .04$ ). This means that females do perform better on the decomposition task, but only after surpassing the age of 11 years old.

Age was found to have a weak, negative correlation with how much students like to dance ( $r = -.323, p = .008$ ) and the grade students give to the lesson ( $r = -.386, p = .001$ ). This means that when students get older, they like dancing less, and give a lower grade. There are no differences found in this trend between males and females.

Flow has a weak correlation with how much students like to dance ( $r = .47, p < .001$ ) and their understanding of the task ( $r = .438, p < .001$ ). Thus, when students like to dance and have a good understanding of the task, they experience more flow. Flow is moderately correlated with the grade they



give the lesson ( $r = .603$ ,  $p < .001$ ), which means the grade they give the lesson becomes higher, when students experience more flow. Understanding of the task is also correlated moderately with how much they like the task ( $r = .509$ ,  $p < .001$ ) and the grade students give the lesson ( $r = .571$ ,  $p < .001$ ). Thus, when students have a good understanding of the task, they like the task more and give the lesson a higher grade.

## **Discussion**

The aim of this study was to investigate from what age lessons about decomposition and abstraction are appropriate. Three main findings are worth discussing in further detail. The first main finding concerns the relation between age and abstraction and decomposition. Second, an interaction is found between gender and the abstraction task. Third, for both tasks, there are no significant differences between age groups on perceived difficulty, cognitive load, and flow.

### **Abstraction**

Results show that students do not show the same level of abstraction skill across all ages, as expected. When students got older, they got better at the abstraction task. This is largely in line with the results of Marini and Case (1994). They found that the capacity for abstract reasoning begins to appear at the age of 11 or 12. In the current study, abstract reasoning seems to appear earlier, but it is unclear at what level. In a study from Dumontheil (2014), it was found that on a task for relational reasoning, 7 to 9 and 14 to 17 year olds make significant improvements during that span. This could be related to the improvements seen in this study around that age. It was expected that age would have a positive correlation with flow, but this correlation is not found. This could be because older students might not have felt as challenged as expected, which would make them feel bored. This could indicate that this subject is more suitable for somewhat younger students (i.e., 10 years old and younger).

During the ancillary analyses, it was found that after the age of 9.5 years old, females begin to distance themselves of males on the abstraction task. A similar finding was reported by Statter & Armoni (2016), who found that “female students achieved better grades (...) in our abstract scale grading” (p. 83). Their study involved 7<sup>th</sup> graders (ages 13 to 14), with the main goal to develop an intervention to teach abstraction skills as early as possible. Also, using functional magnetic resonance imaging (fMRI), girls between 9 and 15 years old have been found to show significantly greater activity in brain areas linked to abstract thinking through language than boys of the same age (Burman, Bitan, & Booth, 2008). Boys, on the other hand, showed much more activity in visual and sensory brain areas. The findings of the current study could be related to the notion that girls develop faster cognitively than boys, due to their tendency to

optimize brain connections earlier than boys (Lim, Han, Uhlhaas, & Kaiser, 2015). Also, girls are believed to hit puberty earlier than boys; a period linked to the development of abstract reasoning (Kipke, 1999).

When analysing how difficult the lessons about abstraction were perceived by students, the expectation was that the older students got, the easier the lessons would be experienced. Results show that there are no differences in perceived difficulty, cognitive load, or flow between age groups. This means that age does not influence these constructs. Although the abstraction task should be suitable for all ages in primary school, it was expected that younger students would have more difficulty with the task than older students. It could be that the task resulted in a so-called ceiling effect, meaning that the task was too easy for all respondents, making it difficult to differentiate between ages. When this would be the case, however, the correlation of the task with age, and interaction with gender, would not be present. Another explanation lies in the measurements. The current study used self-reporting questionnaires for a sample consisting in its entirety of primary school students. The questionnaires were already dismissed for students in 1<sup>st</sup> grade, but even in higher grades the reliability of the questionnaire is to be questioned. It was observed that students tend to answer in extremes on questions with a five-point Likert scale, sometimes out of enthusiasm (e.g., thinking the extremes represent all positive answers), out of boredom (e.g., not feeling like answering the questionnaire), or because they see their neighbour filling in the questionnaire a certain way. The inferences drawn from the self-reported questions should therefore be interpreted with caution. Future research should focus on measuring these constructs without self-reporting questionnaires, but instead use interviews with the students, either individually or in small groups. Another result shows that students with prior experience in the abstraction task, perform better. Prior experience, of course, influences student achievement (Hailikari, Katajavuori, & Lindblom-Ylänne, 2008).

## **Decomposition**

Results show a different pattern between the decomposition task and age groups than with abstraction. Between the ages of 7 and 10 years old, there is no difference in performance on the decomposition task, also not between males and females. However, when looking at students older than 11 years old, the number of decompositions made per minute more than doubles, and females begin to outperform males significantly. This age could be an indication of an appropriate moment to invest in more lessons on decomposition in relation to computational thinking, as decomposition in relation to arithmetic problems is already taught to students as young as 5 years old (Cheng, 2012). Due to a severe lack of literature about gender and decomposition, it is difficult to explain the superiority of girls other than their general earlier cognitive development (Lim et al., 2015).

When analysing how difficult the lessons about decomposition were perceived by students, the expectation was that the older students got, the easier the lessons would be experienced. Also, the decomposition lesson was expected to be perceived more difficult than the abstraction lesson. Similar to the abstraction task, results show that there are no differences in perceived difficulty, cognitive load, or flow between age groups. This means that age does not influence these constructs. However, students' perceptions of the decomposition task were significantly less positive in the oldest age group. Older students sometimes showed confusion in how decomposing a dance routine fits in with programming, when this link was discussed. This is reflected in the differences in grades per age group; older students gave a lower grade than middle aged students. There is no difference in gender for this difference, which means that older boys did not like the lessons less or more than older girls.

The decomposition lesson was not experienced more difficult (or easier) than the abstraction lesson, against expectations. This is an interesting result, seeing how decomposition was thought of to be the most difficult computational thinking skill to master (Selby, 2015). And while Selby found that decomposition is very difficult, other research pleads for the difficult nature of abstraction for primary school students (Booth, 2013). In addition to the fact that students did not feel any significant differences in how they perceived the two tasks, this could mean that statements about what computational thinking skill is the most difficult to master, are mostly conjecture. Seeing how students responded to the abstraction and decomposition tasks in the current study, it seems that the tasks do not transcend each other in difficulty. Students with prior experience do not perform better on the decomposition task. A previous study by Tobias (1994) found a linear relationship between prior knowledge and interest. However, when including the items that resemble interest in the subject ('I liked the lesson', or 'I like dancing'), there was still no association found. To engage in relatively simple decomposition, apparently prior experience is not required.

### **Factors of influence**

For the decomposition task, it could be argued that the way the students' success in decomposition is expressed (i.e., in decomposition made per minute), has low external validity. Students had to come up with a dance themselves, which they had to decompose. Also, a class could have had a differently oriented education, one that could possibly lead to better or worse performances on the decomposition task. Future research should make it a point of emphasis to operationalize decomposition skill in a more valid way. A first way to do this could be to make several decomposition tasks in different domains (e.g., eating lunch), to prevent students' domain specific skills. For example, students with a dancing background could have benefited from this decomposition task. A second way of making more valid decomposition tasks, would be to add variations in the complexity of the tasks (e.g., a more extensive lunch, a lunch with different

kinds of food). Finally, a predetermined task (e.g., the same dance for all students) could more effectively differentiate students' decomposition skill. When piloted and standardized, a norm could be made for different age groups.

Also, the construct validity of the abstraction and decomposition tasks can be questioned. That is, was it really abstraction and decomposition that were appealed to during the tasks? The tasks are used in the UK national curriculum to introduce these concepts, but also call upon students' skill in among others reading, drawing, creativity, motivation, and motor skills.

### **Relevance & implications**

The current study gives directions from what age lessons about abstraction and decomposition are appropriate, and contributes to the scarce collection of empirical studies on computational thinking, and decomposition in particular. It was observed that students gradually become more skilled in abstraction like tasks as they become older, with females following a steeper development than males after the age of 9.5 years old. These results are interesting and relevant. With the upcoming desires to implement computational thinking skills earlier in the curricula, educators should consider possible differences between males and females when it comes to abstraction. Compared to existing literature, the current study provides more insights on the differences between males and females and computational thinking, and the paradox that shows when faced with the reality: a considerable shortage of women in the Science, Technology, Engineering and Math (STEM)-fields, despite multiple studies reporting that girls perform as good, or even better in these fields (Corbett & Hill, 2015; Grover & Pea, 2013; Modi et al., 2012; Wang & Degol, 2013). Also, most existing curricula, like the national curriculum for computing in the U. K., have abstraction introduced around 11 years old. However, it is observed that both lessons about abstraction as decomposition could be introduced as early as 1<sup>st</sup> grade (6 years old), although students' performance on these task will double (and eventually triple) rapidly when reaching higher grades.

For decomposition, it would be interesting to see what the role of cognitive load and Miller's working memory theory of 'seven plus or minus two' would be when decomposing a problem. The steep development of decomposition after the age of 11 is not a flawless result, as the influences of external variables are still to be determined. However, since it is the age most computational thinking curricula start with, it does provide some insights in the development of decomposition.

When translating the results of the current study for primary school teachers, it is important to describe specific guidelines. For abstraction, expect a gradual development for both boys and girls. When students reach 4<sup>th</sup> grade (groep 6), girls will likely start to distinguish themselves from boys on abstraction tasks. Additional and more challenging material would be desirable for this scenario. For decomposition, these effects seem to be postponed to 6<sup>th</sup> grade (groep 8). When focussing on computational thinking skills

that involve decomposition, a significant improvement can be expected around this grade.

## **Conclusion**

Research to computational thinking skills in primary education is still rare. It is advised to offer primary school students the opportunity to get acquainted with these 21<sup>st</sup> century skills as soon as possible. As Kramer (2007) states, computational thinking skills are not only useful for programming or computer science, but for all disciplines. Some opponents feel like young children should not be exposed to more screen time, technology, or computers at all. That is why unplugged, hands-on learning materials should be developed and advocated *in addition to* the computer related content. Questions that started this study were about how difficult computational thinking skills would be for primary school students and if there would be a particular age at which students should be educated in these skills. It seems that students did not experience the unplugged lessons to be too difficult. Also, there are clear directions about the development of these skills through different ages. And, without anticipating gender differences, there were. The results of this study provide handles to continue to improve lesson material in this new and exciting area of education, with clues for adaptations for gender. The overall conclusion is that although inferences should be made carefully, abstraction and decomposition have a future in primary education. Especially in the advanced stage of primary education (grade 4 through 6), students start to make progress in their abstract reasoning. In a next study, it would be interesting to invest in closing the gap on gender differences. The lesson material could be extended to more difficult and easier tasks, and present these tasks to boys and girls from different ages. Also, it would be interesting to explore the other computational thinking skills (i.e., algorithms, evaluation, and generalizations) in primary school, and look for gender differences there. Girls in particular should be motivated by teachers to explore these STEM-fields they are traditionally not involved in. Although we are nearly two decades in, it is not too late to start investing in these 21<sup>st</sup> century skills.

## **Acknowledgments**

I thank my supervisors Lars Bollen, Tessa Eysink, and Jos Tolboom for their extensive support and feedback. I would also like to thank Ton de Jong for suggestions made during the thesis. Further, I would like to thank Allard Strijker, Petra Fisser, Hylke Faber, Natasa Grgurina, Sandra Legters, Wietse van Bruggen, Teun Meijer, Jane Waite, and the students and teachers of the participating schools for their contributions to the study.

## References

- Alter, A. L., Oppenheimer, D. M., Epley, N., & Eyre, R. N. (2007). Overcoming intuition: metacognitive difficulty activates analytic reasoning. *Journal of Experimental Psychology: General*, 136, 569–576. doi:10.1037/0096-3445.136.4.569
- Booth, W. A. (2013). Mixed-methods study of the impact of a computational thinking course on student attitudes about technology and computation. *Graduate School*. Retrieved from [https://baylor-ir.tdl.org/baylor-ir/bitstream/handle/2104/8726/William\\_Booth\\_phd.pdf?sequence=1](https://baylor-ir.tdl.org/baylor-ir/bitstream/handle/2104/8726/William_Booth_phd.pdf?sequence=1)
- Burman, D. D., Bitan, T., & Booth, J. R. (2008). Sex differences in neural processing of language among children. *Neuropsychologia*, 46, 1349–1362. doi:10.1016/j.neuropsychologia.2007.12.021
- Cheng, Z. J. (2012). Teaching young children decomposition strategies to solve addition problems: An experimental study. *Journal of Mathematical Behavior*, 31, 29–47. doi:10.1016/j.jmathb.2011.09.002
- Corbett, C., & Hill, C. (2015). Solving the equation - the variables for women's success in engineering and computing. *Aauw*. doi:10.1103/PhysRevA.75.063427
- Csikszentmihalyi, M. (1975). Beyond boredom and anxiety: experiencing flow in work and play. *The Jossey-Bass Behavioral Science Series*, 231. doi:10.2307/2065805
- De Jong, T. (2010). Cognitive load theory, educational research, and instructional design: Some food for thought. *Instructional Science*, 38, 105–134. doi:10.1007/s11251-009-9110-0
- Dumontheil, I. (2014). Development of abstract thinking during childhood and adolescence: The role of rostral lateral prefrontal cortex. *Developmental Cognitive Neuroscience*, 10, 57–76. doi:10.1016/j.dcn.2014.07.009
- Duncan, C., & Bell, T. (2015). A pilot computer science and programming course for primary school students. *Workshop in Primary and Secondary Computing Education*, 1, 39–48. doi:10.1145/2818314.2818328
- Eysink, T. H. S., Gersen, L., & Gijlers, H. (2015). Inquiry learning for gifted children. *High Ability Studies*, 26, 63–74. doi:10.1080/13598139.2015.1038379
- Grover, S., & Pea, R. (2013). Computational thinking in K-12: a review of the state of the field. *Educational Researcher*, 42, 38–43. doi:10.3102/0013189X12463051
- Hailikari, T., Katajavuori, N., & Lindblom-Ylänne, S. (2008). The relevance of prior knowledge in learning and instructional design. *American Journal of Pharmaceutical Education*, 72, 1–8. doi:10.5688/aj7205113
- Hazzan, O. (1999). Reading abstraction level when learning abstract algebra concepts. *Educational Studies in Mathematics*, 40, 71–90.

- Kipke, M. D. (1999). *Adolescent development and the biology of puberty*. Washington, D.C.: National Academies Press.
- Kramer, J. (2007). Is abstraction the key to computing? *Communications of the ACM*, 50, 37–42.  
doi:10.1145/1232743.1232745
- Kuhn, D., Langer, J., Kohlberg, L., & Haan, N. S. (1977). The development of formal operations in logical and moral judgment. *Genetic Psychology Monographs*, 95, 97–188.
- Lee, T. Y., Mauriello, M. L., Ahn, J., & Bederson, B. B. (2014). CTArcade: Computational thinking with games in school age children. *International Journal of Child-Computer Interaction*, 2, 26–33.  
doi:10.1016/j.ijcci.2014.06.003
- Lim, S., Han, C. E., Uhlhaas, P. J., & Kaiser, M. (2015). Preferential detachment during human brain development: age- and sex-specific structural connectivity in diffusion tensor imaging (DTI) data. *Cerebral Cortex*, 25, 1477–1489. doi:10.1093/cercor/bht333
- Lister, R. (2011). Concrete and other neo-piagetian forms of reasoning in the novice programmer. *Conferences in Research and Practice in Information Technology Series*, 114, 9–18.
- Marini, Z., & Case, R. (1994). The development of abstract reasoning about the physical and social world. *Child Development*, 65, 147–159.
- Mayer, R. E., & Moreno, R. (2010). Nine ways to reduce cognitive load in multimedia learning. *Educational Psychologist*, 1520, 43–52. doi:10.1207/S15326985EP3801
- Ministry of Education, O. (2003). A guide to effective instruction in mathematics, kindergarten to grade 3. *Number Sense and Numeration*.
- Modi, K., Schoenberg, J., & Salmond, K. (2012). Generation STEM - full report. *Girl Scout Research Institute*. Retrieved from [www.girlscouts.org/research/pdf/generation\\_stem\\_full\\_report.pdf](http://www.girlscouts.org/research/pdf/generation_stem_full_report.pdf)
- Myers, D. G. (2007). *Psychology* (8th ed.). New York, New York, USA: Worth Publishers.
- Nickerson, H., Brand, C., & Repenning, A. (2015). Grounding computational thinking skill acquisition through contextualized instruction. *Proceedings of the International Conference on International Computing Education Research*, 207–216. doi:10.1145/2787622.2787720
- Onderwijsraad. (2005). *Betere overgangen in het onderwijs*. Retrieved from <https://www.onderwijsraad.nl/upload/documents/publicaties/volledig/betere-overgangen-in-het-onderwijs.pdf>
- Piaget, J. (1972). Intellectual evolution from adolescence to adulthood. *Human Development*, 15, 1–12.  
doi:10.1159/000271225
- Polya, G. (1945). How to solve it. *The Mathematical Gazette*. doi:10.2307/3609122
- Ribiero, N. F., & Yarnal, C. M. (2010). The perceived difficulty assessment questionnaire (PDAQ): methodology and applications for leisure educators and practitioners. *Schole: A Journal of Leisure*

- Studies and Recreation Education*, 25, 111–115. Retrieved from <http://js.sagamorepub.com/schole/article/view/52>
- Robins, A., Rountree, J. and Rountree, N. (2003). Learning and teaching programming : a review and discussion. *Computer Science Education*, 13, 137–172. doi:10.1076/csed.13.2.137.14200
- Se, S., Ashwini, B., Chandran, A., & Soman, K. P. (2015). Computational thinking leads to computational learning: Flipped class room experiments in linear algebra. *Proceedings of the International Conference on Innovations in Information, Embedded and Communication Systems*, 1–6. doi:10.1109/ICIECS.2015.7193021
- Selby, C. C. (2013). Computational Thinking : The developing definition. *ITiCSE Conference 2013*, 5–8.
- Selby, C. C. (2015). Relationships: computational thinking, pedagogy of programming, and Bloom's Taxonomy. In *Proceedings of the Workshop in Primary and Secondary Computing Education on ZZZ - WiPSCE*. New York, USA: ACM Press. doi:10.1145/2818314.2818315
- Smith, L. (1992). *Jean Piaget: critical assessments*. Abingdon, UK: Routledge.
- Stachel, J., Marghitu, D., Brahim, T. Ben, Sims, R., Reynolds, L., & Czelusniak, V. (2013). Managing cognitive load in introductory programming courses: A cognitive aware scaffolding tool. *Journal of Integrated Design and Process Science*, 17, 37–54. doi:10.3233/jid-2013-0004
- Statter, D., & Armoni, M. (2016). Teaching abstract thinking in introduction to computer science for 7th graders. In *Proceedings of the Workshop in Primary and Secondary Computing Education on ZZZ - WiPSCE '16* (pp. 80–83). New York, USA: ACM Press. doi:10.1145/2978249.2978261
- Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction*, 4, 295–312. doi:10.1016/0959-4752(94)90003-5
- Thijs, A., Fisser, P., & Hoeven, M. van der. (2014). *21e eeuwse vaardigheden in het curriculum van het funderend onderwijs. SLO*.
- Tobias, S. (1994). Interest, prior knowledge, and learning. *Review of Educational Research Spring*, 64, 37–54. doi:10.3102/00346543064001037
- Wang, M. Te, & Degol, J. (2013). Motivational pathways to STEM career choices: Using expectancy-value perspective to understand individual and gender differences in STEM fields. *Developmental Review*, 33, 304–340. doi:10.1016/j.dr.2013.08.001
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49, 33–35. doi:10.1145/1118178.1118215
- Wing, J. M. (2008). Computational thinking and thinking about computing. *Proceedings of the International Parallel and Distributed Processing Symposium*, 3717–3725. doi:10.1109/IPDPS.2008.4536091





# KS2 Introduction to Abstraction Unplugged Activity

## Guess what?

**Recommended Year Group:** Any Key Stage 2

**Activity Duration:** 30 minutes

## Concepts and approaches



### Abstraction

## Overview

This is an unplugged activity in which pupils create simple models from modelling dough or draw quick sketches for a partner to guess what they are representing. In doing so they learn they are ignoring unimportant details and only including that which is most important, and in so doing are abstracting. Pupils link this idea to what is and is not included in simple computer simulations and games.

## Pupil objectives

- I can say what is important and I must include
- I can say what is unimportant and I can ignore
- I can say how a computer program (for example, a computer simulation or game) includes what is important.

## Introduction - (5 mins)

- Invite two pupils to the front of the class to play a short 'guess what' game. Give one pupil a 'guess what' card and they must either sketch it, or make it using modelling dough so that their partner guesses what it is. The quicker their partner guesses correctly the better they have done.
- Lead a discussion to consider what enabled the guesser to work out what item was being drawn or modelled. Lead to the idea that the maker had to work out what was most important about the item, and what could be ignored, which helped the guesser work out what the item was.
- Model how to then think about and record what was included and what was ignored, creating a class example. You could use the table on slide 2 of the presentation or use a recording sheet displayed on the IWB using a visualiser. Ask pupils to help you think what was included and what was ignored for the item that was just guessed and show pupils how to add to the table. Ask pupils what other aspects might have been better to include and add to the table.
- Show slide 4 of the presentation to introduce the learning objectives, if you wish.

## Main activity (15 mins)



- Group pupils in pairs. Give each pair a few 'guess what' cards, a whiteboard and pen, modelling dough and paper (or recording sheets) to note their include, ignore notes. (Alternatively group pupils in threes, and one pupil thinks of the 'guess what' item.)
- Pairs should now have time to play the game – with 1 player selecting a card and drawing or modelling the object and the other pupil guessing what has been created or drawn. They should then work collaboratively to work out what was included and what was ignored, adding notes about what might have been better. Ensure pupils swap roles during this time.
- As pupils are playing:
  - Circulate around groups to ensure pupils are thinking carefully about what was included and what was ignored and why.
  - Stop the whole class on a couple of occasions and compare objects or drawings being created for the same 'guess what' card to discuss what common aspects are being included or excluded.

### Plenary (5 mins)

- Select two or three pairs to share examples and discuss as a class any similarities and differences in what was included or ignored. Or select any notable examples that showed surprising or interesting approaches to abstraction, e.g. examples that everyone found easy or hard.
- Ask pupils in pairs to think of computer simulation they know about (e.g a flight, driving, simulator at a theme park, planets simulator, fossil formation etc) or a computer game they know about and to think about what is included and what is not included. (Use slide 5/6/7 if needed). *See resources for a selection of simulations that could be displayed if needed.*
- Share ideas as a class and lead a discussion to explain that designers of simulations and games need to decide what to include or ignore when creating programs and that the skill of working out what is important and not important to include is essential. You could introduce the term abstraction and explain that this is an important area of study in computing.

### Differentiation

#### Support:

Use additional targeted questions during main task to check basic understanding of what is important, not important.

#### Stretch & Challenge:

- Challenge pupils to think about any common themes of what is and is not included across several computer games or simulations they are familiar with e.g Angry Birds and Candy Crush both have simple characters, scores, levels, bright colour but do not have complex story lines like some questing games. Looking for patterns and generalised aspects is another important computational thinking skill.
- Challenge pupils to describe a game in a minimum number of words. For example, Candy Crush is a timeline game. The player moves along the line solving problems in a step by step order. They can't move on without succeeding.



## Assessment Opportunities

- Informal teacher assessment of pupils during main task and plenary. Focus on understanding of:
  - Thinking what is important to include.
  - Thinking what can be ignored.
  - Being aware that thinking about what is ignored or included in computer simulations and games is an important aspect of design

# Teaching notes

## Concepts and approaches

### Abstraction

Abstraction is about simplifying things; identifying what is important without worrying too much about the detail. Abstraction allows us to manage complexity. In this activity, pupils abstract as they identify what can be ignored and what is important about the items they are drawing or modelling. They also consider what is ignored or included in computer simulations and games they know.

### Curriculum links

#### Computing

- Abstraction is part of the overarching aims of the computing curriculum which seeks to ensure that all pupils: ‘can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation.’

#### Art and Design: Pupils should be taught:

- to improve their mastery of art and design techniques, including drawing, painting and sculpture with a range of materials [for example, pencil, charcoal, paint, clay]

### Prior knowledge

None, although having done [Fossil Formation](#) or another abstraction activity is an advantage.

### Resources (see downloads below)

- Guess what cards (or create your own topic based words)
- Modelling material e.g. playdough
- Individual whiteboards and whiteboard pens
- Paper or include/Ignore recording sheet
- Lesson presentation
- Access to the internet and/or scratch to display simulations if needed (See slide 6 of the presentation for examples.)

### Related activities

[Fossil Formation](#)

[Solar System Simulation](#)

[Modelling the Internet Activity](#)



## KS1/2 Introduction to Decomposition Unplugged: Tut, clap or jive

**Recommended Year Group:** Any Key Stage 1 or 2

**Activity Duration:** 30 minutes

### Concepts and approaches



#### Decomposition

#### Overview

This is an unplugged activity in which pupils create hand clapping, hand tutting or hand jive sequences of movements. Pupils break the sequence of actions down into parts and in so doing are decomposing. Pupils link this idea to breaking problems down when creating computer programs such as animations or games.

#### Pupil Objectives

- I can break a sequence of moves down into its parts (KS1)
- I can decompose a sequence (KS2)
- I can say why this is useful (KS1/2)
- I can say how decomposition is used when creating computer programs like animations or games (KS2 – optional)

#### Introduction (5 minutes)

- Show pupils a sequence of hand movements, this could be a hand jive or tutting moves or clapping sequence. (See resources for ideas.) *The sequence needs to be relatively complex or long so that pupils will find it difficult to remember the parts without it being broken down into parts.*
- Ask pupils to now recreate the sequence without showing it to them again, or explaining the parts.
- Lead a discussion around how you could teach them it in a more effective way, leading to the idea of breaking the sequence down into parts.
- Explain that breaking something down into parts is called decomposition. (KS2)
- Model how to break the sequence into parts (decompose it). Record your parts using either slides 4 or 5 of the presentation or using the [decomposition design sheet](#).
- Model how to number and/or name each part and draw an image of each part. In KS2 you might also add notes. (Perhaps model just breaking down the first 2 or 3 parts.)
- Show slide 2 or 3 of the presentation to introduce the learning objectives, if you wish.



### Main Activity (10 minutes)

- Group pupils in pairs. Give each pair paper or the decomposition design sheet on which to record their sequence.
- Pairs should now have time to work out their sequence of movements and re- cord their decomposition. Pupils should work collaboratively discussing moves, recording each part, testing it out and debugging it.
- As pupils are working:
  - Circulate around the pairs to ensure pupils are breaking down their sequence into parts.
  - Stop the whole class on a couple of occasions and ask selected pairs to demonstrate their sequence so far.
  - Ask pairs of pupils to swap their designs and try them out – to debug them.

### Sharing sequences and decompositions (10 minutes)

- Ask a selection of pairs to teach the class their finished sequence, using their decomposition to help them. If possible show their design at the same time, per- haps using a visualiser or mobile device to mirror to an IWB. (Note: This broken down set of steps is an algorithm.)

### Plenary (5 minutes)

- Lead a class discussion about how breaking down the sequence into parts helped their design process and sharing of the sequence. Points for discussion might include, being able to see the overall sequence of parts, being able to spot repetition, being able to focus on one part at a time.
- If time, or as homework, ask pupils in pairs to think how computer programs are created by computer scientists, for example if they were creating a new computer game, say one like Angry Birds. Some designers might work on the first level of the game others on the next. Some programmers might work on the backgrounds, some on the sound, others on the action. Explain that decomposition is a fundamental skill when working with computers as it helps us break down complicated problems, focus on one part at a time and share the work with others.

### Differentiation

**Support:** Some pupils could work in a group with an adult, perhaps photographing each part of their sequence.

**Stretch & challenge:** Challenge pupils to spot any repeated moves and how they could use a 'repeat command' or replace the repeated parts with a summary name e.g replace two steps open hand, close hand with one step open/close hand. Decomposing the summarised part just once. Pupils could add a selection (an if... then... else...). For example, if wearing a blue jumper do the actions low, else do them high.

### Assessment opportunities

- Informal teacher assessment of pupils during main task and plenary. Focus on understanding of decomposition e.g. are the parts in the right order, do you have all the parts, where do you start/end, do you repeat any parts, can you further



decompose a particular part? Also can pupils relate this to creating computer programs.

## Teaching Notes:

### Concepts and approaches

#### Decomposition

The process of breaking down a problem into smaller manageable parts is known as decomposition. Decomposition helps us solve complex problems and manage large projects.

Decomposition is breaking a problem or system down into its parts. Sometimes we break those parts down further. As we decompose something we learn more about it. If we decompose a problem it becomes more manageable as we can deal with the parts separately and more easily.

In this activity pupils create a sequence of hand actions, they break the sequence down into parts and in so doing are decomposing. They also think about how decomposition is used when creating computer games.

**Note:** The resultant design pupils create after decomposing their sequence of hand actions is a set of instructions to perform a task, it is an algorithm.

### Curriculum links

#### Computing:

- **Key Stage 1:** Although decomposition is not explicitly mentioned in the key stage 1 programme of study, it is used when pupils break a task down to work out the steps in a simple algorithm or when they choose a part of a program to work on when they 'create and debug programs'.
- **Key Stage 2:** Solve problems by decomposing them into smaller parts

#### PE:

- perform dances using simple movement patterns (KS1)
- perform dances using a range of movement patterns (KS2)

### Resources (downloadable from webpage)

- Paper or [decomposition design sheet](#) (**Note:** there is a KS1 page and a KS2 page)
- Lesson presentation (includes examples)
- Access to YouTube to display tutting, clapping or jive examples if needed (see presentation slide 9)

### Related activities

#### [KS1 Crazy Characters Algorithms Activity](#)

Appendix C  
Self-reporting questionnaire

## Vragenlijst

Naam:.....

Naam school:.....

Datum:..... ☐ jongen ☐ meisje

Geboortedatum:..... (Bijv. 22 januari 2008) Leeftijd:.....

Groep:.....

### Instructie

Dit is een lijst met 25 vragen. Achter elke vraag staan een paar bolletjes.

Bijvoorbeeld:

	Klopt niet			Klopt helemaal
1 Ik vind de opdrachten leuk	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Je kiest het antwoord dat het best bij jou past en kleurt dit bolletje in, of je omcirkelt het best passende antwoord.

Dus als je de opdrachten heel leuk vind, kleur je het meest rechtse bolletje in. Vind je de opdrachten helemaal niet leuk, dan kleur je het meest linkse bolletje in. Zit het er een beetje tussenin, kies dan een ander bolletje wat beter past.

Er zijn geen goede of foute antwoorden. Elk antwoord is goed als het maar je eigen mening is. Werk vlot door en denk niet te lang over het antwoord na, want het gaat om je eerste indruk. Sla geen enkele vraag over. Ook al is een vraag moeilijk te beantwoorden, probeer dan toch bij elke vraag je iets voor te stellen en een antwoord te geven.

Wat vond je van de				
1 Moeilijkheid	Heel moeilijk	Moeilijk	Makkelijk	Heel makkelijk
2 Lengte	Heel lang	Lang	Kort	Heel kort
3 Duidelijkheid	Heel onduidelijk	Onduidelijk	Duidelijk	Heel duidelijk

Vraag	Helemaal niet				Heel erg
4 Hoe diep moest je nadenken tijdens de taak?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
5 Hoe erg moest je je haasten tijdens de taak?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
6 Hoe goed lukte de taak?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
7 Hoe hard moest je werken voor de taak?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
8 Hoe onzeker, boos, of vermoeid was je?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Vraag



 Klopt niet

Klopt helemaal 

## Appendix C

### Self-reporting questionnaire

9	Ik vind de opdrachten leuk	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
10	Ik vind het fijn dat je bij deze opdrachten nieuwe dingen leert	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
11	Deze opdrachten vind ik nuttig	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
12	Ik hoef geen beloning. De opdrachten gaven me plezier genoeg!	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
13	Deze opdrachten vond ik erg interessant	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
14	Denken ging makkelijk	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
15	De juiste gedachten kwamen vanzelf	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
16	Bij iedere opdracht wist ik wat ik moest doen	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
17	Ik had het gevoel dat ik alles onder controle had	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Vraag		 Klopt niet		Klopt helemaal 	
18	Ik vond de opdracht leuk	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
19	Ik snapte goed wat ik moest doen	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
A	Ik vind het leuk om te tekenen	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
D	Ik vind het leuk om te dansen	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
A	Ik teken vaak als hobby	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
D	Ik heb dansles gehad	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

22 Als je deze les een cijfer tussen de 1 (heel slecht) en 10 (heel goed) moet geven, welk cijfer geef je dan? \_\_\_\_\_

23 Hoe vond je de les in één woord?

24 Heb je ooit eens zoiets gedaan zoals vandaag? Ja Nee  
(omcirkel het juiste antwoord)

25 Zo ja, wat heb je dan gedaan?

---



---

**BEDANKT!**