Routledge
Taylor & Francis Group

Interactive Learning Environments

ISSN: 1049-4820 (Print) 1744-5191 (Online) Journal homepage: http://www.tandfonline.com/loi/nile20

# Exploring children's perceptions of developing twenty-first century skills through computational thinking and programming

Gary Ka-Wai Wong & Ho-Yin Cheung

**To cite this article:** Gary Ka-Wai Wong & Ho-Yin Cheung (2018): Exploring children's perceptions of developing twenty-first century skills through computational thinking and programming, Interactive Learning Environments, DOI: 10.1080/10494820.2018.1534245

**To link to this article:** https://doi.org/10.1080/10494820.2018.1534245


Published online: 25 Oct 2018.


Submit your article to this journal


Article views: 2


View Crossmark data

Full Terms & Conditions of access and use can be found at
http://www.tandfonline.com/action/journalInformation?journalCode=nile20

INTERACTIVE LEARNING ENVIRONMENTS
https://doi.org/10.1080/10494820.2018.1534245

Routledge
Taylor & Francis Group

Check for updates

# Exploring children's perceptions of developing twenty-first century skills through computational thinking and programming

Gary Ka-Wai Wong and Ho-Yin Cheung

Faculty of Education, The University of Hong Kong, Pokfulam, Hong Kong, People's Republic of China

**ABSTRACT**

The role of programming in computing education for children has grown rapidly in recent years with the proliferation of specially designed programming tools, which is grounded on Seymour Papert's theoretical work in Constructionism. Studies show that some children can develop computational thinking skills and practices with programming activities when learning with the tools through a well-design curriculum in elementary education (or primary education). However, existing studies may not completely address whether programming skills and computational thinking can be connected to the development of other generic skills, which are considered important to the learning and cognitive development of children. This study investigates the impact of programming on three learning competencies (creative thinking, critical thinking and problem solving) known as twenty-first century skills. The conceptual mapping between programming, computational thinking and the three learning competencies is presented. In a one-year intervention in a primary school, students were taught how to build interactive games through programming, and thus explored some basic computational thinking concepts in class. Our results show that children perceived a significant impact of programming on their learning competencies. Yet, the transferability of twenty-first century skills developed through computational thinking and programming may require a further study. Our study provides insights from children as primary respondents to help direct future research in the field of programming and computational thinking education and its potential impact on twenty-first century skills.

**ARTICLE HISTORY**
Received 4 April 2018
Accepted 7 October 2018

**KEYWORDS**
Programming education; computational thinking; primary curriculum; twenty-first century skills; constructionism

## Introduction

Computer programming is considered a major digital literacy competence for the twenty-first century. In the process of learning how to programme, learners can develop computational thinking (CT). CT, which was popularised by Wing (2006), is defined as the problem-solving and thinking processes of formulating computational problems and their solutions, which can be effectively solved by a computer through programming (Grover & Pea, 2013). Creativity is a possible attribute of the process of developing computational artefacts and solving problems (Gretter & Yadav, 2016). People also use their critical thinking skills (i.e. effective reasoning, systematic thinking and evaluation of evidence (Binkley et al., 2012)) to approach computational problems, which basically involves dealing with programme structure, sequence of logical instructions, computational complexity and social impact when developing a computational solution to the problem. Therefore, programming

**CONTACT** Gary Ka-Wai Wong wongkwg@hku.hk Faculty of Education, The University of Hong Kong, Pokfulam, Hong Kong, China

© 2018 Informa UK Limited, trading as Taylor & Francis Group

and other learning competences (i.e. problem solving, creativity, critical thinking) are closely associated, and are important for future compulsory education.

Educators believe that children can acquire these twenty-first century skills as complementary skills. Various countries have begun to consider introducing programming activities in schools. For example, primary and secondary schools in Hong Kong are developing a curriculum that features programming activities with different tools (Wong, Cheung, Ching, & Huen, 2015). Similar initiatives are also reported in different countries (e.g. Israel, Pearson, Tapia, Wherfel, & Reese, 2015; Kucuk & Sisman, 2017; Manches & Plowman, 2017; Mannila et al., 2014; Webb et al., 2017; Wong et al., 2015).

In addition, the "participatory culture" advocated by Jenkins, Purushotma, Weigel, Clinton, and Robison (2009) can be linked to programming activities because children become content creators in their digital culture through programming activities (Gretter & Yadav, 2016). This digital engagement will become more essential in today's globalised and hyper-connected digital society with social media.

Some studies have noted the commonality between twenty-first century skills and CT through programming (Kong, 2016; Lye & Koh, 2014; Mannila et al., 2014; Webb et al., 2017). However, there is a need for more empirical evidence to assess this relationship. Particularly in the context of primary education, where generic learning competences (i.e. problem solving, creativity and critical thinking) that are taught are fundamental to future learning, studies have mostly focused on the programming performance of primary school students without making an explicit connection between these learning competences and programming (e.g. Lye & Koh, 2014; Shute, Sun, & Asbell-Clarke, 2017). Indeed, connecting CT to twenty-first century skills seems to be trivial, and yet our understanding of the learner's perspective about whether these skills can actually be linked or related to other subjects remains limited (Tran, 2018).

As such, our study addresses the following research questions:

- **RQ1:** What is the perceived change in problem solving, creativity and critical thinking competences across different primary students' ages after learning programming with CT?
- **RQ2:** How do primary students relate the learning of programming to other school-based subjects?

## Theoretical frameworks

### *Constructionism and CT*

Research on CT with programming for children can be traced back to Papert (1980), who coined the term "CT" in his LOGO programming project for children. Papert (1980) emphasised the importance of *constructionism*, whereby learners construct their knowledge by building artefacts through active engagement in an interactive learning environment with technological tools (Kucuk & Sisman, 2017). The composition of concepts and rules happens in the process of learning, where one is actively doing and consciously thinking within the process of engaging in the learning activities (Ackermann, 2001; Alimisis, 2013). On top of this learning by doing-and-thinking approach, Papert (1980) believed that technological tools are the key to empowering children with the ability to deal with situations by themselves. Through building the technological artefacts, children can gain knowledge during the process of thinking about the problems and learning how to solve them. Voogt, Fisser, Good, Mishra, and Yadav (2015) provided a summary of research since Papert's first work introducing LOGO programming to children, which studied children's programming skills and its impact on their learning, including the development of CT skills. They concluded that the benefit of developing CT through programming is clear, but we still need to know more about how it can be embedded in school curricula and other disciplines, and how it should be taught in the K-12 context (Lye & Koh, 2014; Tran, 2018).

In the context of CT through programming, Brennan and Resnick (2012) proposed a well-known conceptual framework consisting of three dimensions to structure the different aspects of CT:

**Table 1.** Conceptual framework of computational thinking.

| Dimensions | Descriptions | Examples | Mapped twenty-first Century Skills |
|---|---|---|---|
| Computational concepts | Syntactic, semantic and schematic knowledge commonly used in programming | Variables, loops (e.g. repeat, do … while), conditionals (e.g. if … then … else), operators (e.g. arithmetic and comparison) | • Critical thinking<br>• Problem solving |
| Computational practices | Strategic knowledge to solve programming problem during the process of thinking and practices | Being incremental and iterative, testing and debugging, reusing and remixing and abstracting and modularising. | • Creativity<br>• Critical thinking<br>• Problem solving |
| Computational perspectives | Understandings of personal, social and technological relationships around them in connection to programming | Expressing and questioning the technology world through computation (e.g. Create their own digital stories with Scratch) | • Creativity<br>• Critical thinking |

*computational concepts*, *computational practices* and *computational perspectives*. Table 1 summarises the basic descriptions with some examples to illustrate the ideas, and how they can be mapped to our three twenty-first century skills. This framework forms a structural understanding of how students might approach programming learning.

## *Creativity and CT*

According to Binkley et al. (2012), the operational definitions of creativity and innovation cover how learners deal with new and worthwhile ideas. These include an open attitude to the ideas, the knowledge and skills to create these ideas, as well as to work on the ideas and innovate, tackling failure in the process. In the context of computer programming and technology education, Manches and Plowman (2017) defined creativity as "generating ideas and strategies as an individual or community, reasoning critically between these and producing plausible explanations and strategies consistent with the available evidence" (Agogi, Rossis, & Stylianidou, 2014, p. 8). Indeed, when children engage in programming activities, they have opportunities to solve computational problems in creative ways and develop a computer program (Mishra & Yadav, 2013) Debugging activities in programming also allow children to think creatively about strategies to refine their ideas.

## *Critical thinking and CT*

Binkley et al. (2012) classify critical thinking and problem thinking in the same category as decision making. According to their definition, these behaviours involve the knowledge, skills and attitudes to

**Table 2.** Mapping the five stages programming into a critical thinking approach.

| Dimensions (Wong et al., 2018) | Examples of questions provoking critical thinking |
|---|---|
| Algorithmic Design of Program | • How should an algorithm be designed to form a feasible solution in the program?<br>• Why should social constraints be considered when developing the algorithm? |
| Building Structure of Program | • How should I form a good program structure?<br>• What is the difference between looping modules and separate instructions? |
| Coding the Program | • What programming blocks should be used to ask for multiple inputs at once?<br>• Where could I find out more reference examples for this programming problem? |
| Debugging the Codes | • Why is the code not giving the correct output?<br>• If I change this programming block, would it affect the whole structure of my program? |
| Evaluating the Program | • How could I revise the program and improve user experience?<br>• What will happen to this program if I add more requirements to the original problem? |

think systematically and evaluate evidence, make reasoned judgements and decisions, and articulate these through clear explanation and justification of the process. Critical thinking is part of programming through CT when children participate in the programming process (Lye & Koh, 2014; Wright, Rich, & Leatham, 2012), because solving any computational problem requires constant judgement and justification of the process, from algorithm design to testing and debugging. Decision making throughout the different stages of programming requires critical thinking skills. Based on the A-E programming conceptual framework by Wong, Jiang, and Kong (2018), the five stages of programming can be linked to their associated critical questions during the processes (See Table 2). If children can learn how to think more critically, they will create more feasible and comprehensive computational solutions to programming problems (DeJarnette, 2012; Moomaw, 2012).

### *Problem solving and CT*

In the context of computer programming, the approach to problem solving is often described as computational thinking (Grover & Pea, 2013; Israel et al., 2015). Indeed, the CT process is triggered if and only if there is a computational problem to be solved. Through a series of thinking processes and skills, the ultimate goal is to solve the programming problem by developing a computer program as a digital artefact. As Fessakis, Gouli, and Mavroudi (2013) described, programming environments are a powerful reconstructible media for CT because they involve a problem-solving process, requiring the participants to analyse, organise, express and evaluate their thoughts clearly and concisely.

In fact, metacognitive skills and problem-solving skills can be developed because the "solver" can "teach" the computer how to solve the computational problem, while articulating their thoughts and observing the outcomes for improvements by programming the computer (Clements & Gullo, 1984). Children can learn how to solve problem by themselves, and collaborative problem solving can also occur during the coding process (Israel et al., 2015). This unique method of algorithmic problem solving through CT enables individuals to solve programming problems efficiently and effectively (Shute et al., 2017). We can evidently categorise CT as a way of thinking and a set of problem-solving skills for computational problems.

## Method

This study introduces a computer programming curriculum in the participating primary school as an intervention to evaluate the impacts of computer programming education on students' CT. Both quantitative and qualitative data were collected through questionnaires and interviews at the beginning and end of the intervention respectively.

### *Participants*

A total of 358 students from a primary school in Hong Kong consented to participate in this study. The students came from three different grade levels: Primary 4, 5 and 6. These students had prior experience of learning programming in the same school during the previous year, as summarised in Table 3.

**Table 3.** Students' prior programming experience and the topics covered during the study.

| Grade in 2015–16 | 2014–15 (prior experience) | 2015–16 (this study) |
|---|---|---|
| Primary 4 | 2DIY[a] (making games) | Kodu (beginners) |
| Primary 5 | 2DIY (advanced) | Kodu (advanced) |
| Primary 6 | Kodu (beginners) | Scratch |

[a]The 2Do It Yourself (2DIY) is a self-developed Flash-based activity building tool in which the students are able to build their own games and give a series of instructions to control the motion of an avatar on the screen. See https://www.2simple.com/product/2do-it-yourself.

### The intervention of programming curriculum

The intervention in this study consisted of a semester-long, school-based programming curriculum designed by the participating teachers. It consisted of 14 programming lessons, each of 35 minutes in duration, adding up to approximately 8 hours of contact time for each student. The curriculum for Primary 4 and Primary 5 focused on Kodu at the beginner and advanced levels respectively. Developed by Microsoft, Kodu is a graphical programming environment designed for children to create their own simple games, and can also help children to learn the concepts of computer science during the game design process (MacLaurin, 2009; Stolee & Fristoe, 2011). Children can place programming tiles in a meaningful order to form the condition and action on each rule, i.e. *Rule -> Condition Action* (Stolee & Fristoe, 2011). Students in Primary 5 also learned programming with Kodu, but at a more advanced level. Students in Primary 6 were taught programming with Scratch (Resnick et al., 2009). This programming language allows students to give instructions to a "sprite" to create a broader range of artefacts such as games, interactive stories and animations. Scratch offers ready-to-use templates and cartoon-type characters/backdrop stages, allowing children to contextualise the outcomes of programmes in different scenarios.

Drawing on the theory of constructionism, the students were given the opportunity to create digital artefacts in the form of a game-based project. This was intended to foster the students' problem solving and critical thinking skills. To encourage creativity, the students were freely allowed and encouraged to remix their programmes and further personalise them.

For the Primary 4 group, the goal was to develop a shooting game in Kodu with one target, a scoreboard and winning conditions using Kodu. For the Primary 5 group, the goal was to develop a war game with multiple targets as villains, health indicators, a scoreboard and winning conditions similar to the group in Primary 4. For Primary 6, the goal was to develop a Scratch-based game involving object searching and escaping from the scene.

Using pre-class videos and in-class instructions, the teacher guided the students through the essential steps to build the basic game progressively before they could remix it. While the students worked on their tasks, the teachers walked around the classroom and provided necessary help. The students were allowed to communicate, move around and discuss with their peers. The programming lab was renovated for this learning approach. Desktop computers were placed around the walls and more available spaces were provided in the middle. Towards the end of each lesson, the teachers invited students to present their outcomes and explain how they solved the programming tasks.

### Sampling

All of the participating students were invited to take part in our questionnaire, while the interview samples were selected by the teachers. The purposive sample consisted of an even distribution of students across low, median and high levels of academic performance. The purposive sampling scheme is shown in Table 4.

### Instrumentation for questionnaire

A questionnaire instrument named "Assessment Program for Affective and Social Outcomes (2nd version) (APASO-II)" was used in this study (EDB, n.d.). It was developed by the Education Bureau

**Table 4.** Sampling scheme for interview.

| | Pre-interview | | | Post-interview | | |
|---|---|---|---|---|---|---|
| Grade Level | Low | Median | High | Low | Median | High |
| Primary 4 | 5 | 5 | 5 | 5 | 5 | 5 |
| Primary 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Primary 6 | 5 | 5 | 5 | 5 | 5 | 5 |

of Hong Kong (EDB) as an online self-evaluation tool for local schools to assess their students' development and needs in the social and affective domains. The Chinese version of APASO-II was used in this study because it is the students' first language.

Various constructs can be assessed under APASO-II, including creative thinking, critical thinking and problem solving, on a four-point Likert scale to indicate how much the respondents agree with the corresponding statements. Creative thinking consists of eight items developed by EDB to assess how well the students are able to do or view things in ways different from others. The critical thinking construct was also developed by the EDB, and it consists of five items concerning how well the students were able to make their own reasoned judgements in complex situations. Finally, the problem-solving construct, consisting of eight items, was adapted from the study by Chow et al. (2009). This construct covers whether the students were able to consider various perspectives, pinpoint the key issues, pick the best method, and then solve the problem effectively. All of the questions were self-assessed and therefore represent the respondents' perceived self-efficacy in these areas. All three constructs were consistent with the corresponding definitions given in the previous sections. Due to copyright restrictions, only local schools in Hong Kong have direct access to the instruments online.

APASO-II is widely used in local schools. This is important because although children from age 8 to 11 (age range of our informants) can cope with self-administered surveys, questionnaires have to be specially developed for them as they still interpret the wordings of questionnaires very literally and have issues with indirect questions and negatively phrased terms (Borgers, De Leeuw, & Hox, 2000). Therefore, the APASO-II instrument, which were designed for and were already familiar to the children in our case, was more appropriate than other instruments designed for adults. To ensure that students understood the items, the teachers also explained the items to their students during the survey. According to EDB, all of the scales and subscales in the APASO-II instrument have been statistically validated and are reliable (EDB, n.d.).

### *Interview protocol*

The interviews were structured and conducted in groups of three. Group interviews were used so that young children might be more comfortable and familiar with the setting (Darbyshire, Macdougall, & Schiller, 2005). Moreover, although researchers suggest that semi-structured interviews can be used for this age group (Borgers et al., 2000), in our case the interview questions (see Appendix) were explicitly framed to collect the knowledge required to answer the research questions, and structured interviews were more appropriate (Cohen, Manion, & Morrison, 2011).

The interviewers asked how students perceived the relationship between programming and other subjects they learned at school, as well as the skills involved in learning these subjects. These two questions were preceded by two warm-up questions, which were slightly different between the pre- and post-interviews. The interview protocols are outlined in the Appendix. Finally, each interview session lasted for 15–30 min. The interviews were conducted in Cantonese, the first language of the students and the teachers.

### *Data analysis*

APASO-II returned a complete dataset of each student's responses to each of the 21 items being tested. The scores for the three constructs were then calculated by averaging the responses under the corresponding items. Two-sample unpaired t-tests were used to test the statistical significance of the pre–post difference, while effect sizes were also calculated for each of the three constructs.

The interviews were audio-taped and transcribed into text for coding. Occurrences of the codes were counted and reported. Miles, Huberman, and Saldana (2014) suggest that although numbers

tend to be ignored in qualitative analysis, sometimes we want to identify a theme or pattern in qualitative data. This aligns with our purpose in this study, as we intended to identify the subjects and knowledge/skills related to programming.

## Findings

### *Demographic data*

The total student population in the study year was 369, and 358 students (97.0%) consented to take part in the pre-test, while 350 (94.9%) completed the questionnaire in the post-test, all uniformly distributed across grade levels (Table 5). The demographic distribution for the interviews followed the purposive sampling scheme stated in Table 4 above.

### *Results from questionnaire*

Our study shows a statistically significant increase in all three constructs after the intervention, in terms of perceived attitude (see Table 6). The effect size values for the pre–post comparisons are also included (Table 7). According to Cohen et al. (2011), a Cohen's *d* in the range of 0.21–0.50 is considered a modest effect, while a range of 0.51–1.00 is considered a moderate effect. In our data, the intervention brought a modest/moderate effect in all of the other pre–post comparisons shown in Tables 8–10. The significant gains in creative thinking are particularly higher across three grade levels than the other two constructs, but the gains become smaller when students have more

**Table 5.** Demographic data for questionnaire.

| | Pre-test ($N = 358$) | | Post-test ($N = 350$) | |
|---|---|---|---|---|
| Items | Frequency | Percentage | Frequency | Percentage |
| Gender | | | | |
| Male | 190 | 53.1% | 185 | 52.9% |
| Female | 168 | 46.9% | 165 | 47.1% |
| Grade Level | | | | |
| Primary 4 | 123 | 34.4% | 122 | 34.9% |
| Primary 5 | 120 | 33.5% | 112 | 32.0% |
| Primary 6 | 115 | 32.1% | 116 | 33.1% |

**Table 6.** Comparison of mean scores of three constructs in the samples.

| | Sample Mean (SD) | | | | |
|---|---|---|---|---|---|
| Tests | Pre-test | Post-test | Construct | *p*-value | Effect size (Cohen's *d*) |
| (i) | 2.78 (0.52) | 3.05 (0.60) | Creative thinking | <0.001*** | 0.47 |
| (ii) | 2.91 (0.60) | 3.10 (0.58) | Critical thinking | <0.001*** | 0.33 |
| (iii) | 2.99 (0.47) | 3.18 (0.58) | Problem solving | <0.001*** | 0.36 |

***$p < 0.001$.

**Table 7.** Baseline comparison with the norm of our study.

| | Sample Mean (SD) | | | |
|---|---|---|---|---|
| Tests | This study ($N \sim 350$) | Norm ($N \sim 9900$) | Construct | *p*-value |
| (i) Before intervention | 2.78 (0.52) | 2.88 (0.65) | Creative thinking | <0.001*** |
| (ii) After intervention | 3.05 (0.60) | | | <0.001*** |
| (iii) Before intervention | 2.91 (0.52) | 2.99 (0.65) | Critical thinking | 0.005** |
| (iv) After intervention | 3.10 (0.58) | | | 0.001** |
| (v) Before intervention | 2.99 (0.47) | 3.06 (0.62) | Problem solving | 0.007** |
| (vi) After intervention | 3.18 (0.58) | | | <0.001*** |

**$p < 0.01$, ***$p < 0.001$.

**Table 8.** Comparison of pre–post difference for primary 4 sample.

| | Sample Mean (SD) | | | | |
|---|---|---|---|---|---|
| Tests | Pre-test ($N = 123$) | Post-test ($N = 122$) | Construct | $p$-value | Effect size (Cohen's $d$) |
| (i) | 2.88 (0.52) | 3.20 (0.57) | Creative thinking | <0.001*** | 0.60 |
| (ii) | 2.92 (0.56) | 3.17 (0.59) | Critical thinking | 0.001** | 0.44 |
| (iii) | 3.04 (0.54) | 3.29 (0.57) | Problem solving | 0.001** | 0.45 |

**$p < 0.01$, ***$p < 0.001$.

**Table 9.** Comparison of pre–post difference for primary 5 sample.

| | Sample Mean (SD) | | | | |
|---|---|---|---|---|---|
| Tests | Pre-test ($N = 120$) | Post-test ($N = 112$) | Construct | $p$-value | Effect size (Cohen's $d$) |
| (i) | 2.76 (0.53) | 3.03 (0.64) | Creative thinking | 0.001** | 0.47 |
| (ii) | 2.94 (0.55) | 3.10 (0.60) | Critical thinking | 0.036* | 0.28 |
| (iii) | 2.98 (0.45) | 3.18 (0.58) | Problem solving | 0.010* | 0.35 |

*$p < 0.05$, **$p < 0.01$.

**Table 10.** Comparison of pre–post difference for primary 6 sample.

| | Sample Mean (SD) | | | | |
|---|---|---|---|---|---|
| Tests | Pre-test ($N = 115$) | Post-test ($N = 116$) | Construct | $p$-value | Effect size (Cohen's $d$) |
| (i) | 2.69 (0.50) | 2.89 (0.56) | Creative thinking | 0.004** | 0.38 |
| (ii) | 2.88 (0.46) | 3.02 (0.54) | Critical thinking | 0.046* | 0.26 |
| (iii) | 2.93 (0.41) | 3.07 (0.56) | Problem solving | 0.032* | 0.28 |

*$p < 0.05$, **$p < 0.01$.

exposure to programming activities prior to this study, e.g. Primary 6 students show a lower gain in their perceived impact of programming in their creative thinking.

Our sampled study results are significantly different from the regional norm data (N ~ 9900) of primary schools in Hong Kong in three constructs. Our students perceived a higher effect on these skills after the intervention, which is at least valid for all grades in our samples.

### *Results from student interviews*

When asked what subjects are related to programming, the students provided a range of answers. The frequencies of these different subjects are tabulated in Table 11. Note that because some students suggested more than one subject, the column totals can be larger than the sample size (which is $N = 45$ in both pre- and post-interviews).

Interestingly, after the intervention only four students suggested that mathematics is related to programming. Although programming is often connected to mathematics and mathematical thinking, this finding shows that the function of the pictorial/block-based interface of the programming environments might succeed in abstracting the mathematical elements behind the computing

**Table 11.** Students' views on subjects related to programming.

| | Pre-interview | Post-interview |
|---|---|---|
| Computer/information technology | 32 | 32 |
| English | 6 | 8 |
| General education | 4 | 5 |
| Mathematics | 23 | 4 |
| Chinese | 1 | 2 |
| Design | 0 | 1 |
| Thinking | 0 | 1 |
| Science | 1 | 0 |

**Table 12.** Skills learned through programming.

| | Pre-interview | Post-interview |
|---|---|---|
| Thinking skills/problem solving | 7 | 23 |
| Creativity | 26 | 19 |
| Controlling/operating the computer | 10 | 10 |
| Programming/creating software/game | 16 | 9 |
| Independent learning | 2 | 5 |
| Communication, collaboration, sharing | 3 | 4 |
| English | 0 | 3 |
| Self-management | 1 | 2 |

technology. In addition, the nature of programming tasks might not be particularly mathematical, and is more instructional and logical (e.g. setting up the conditional statements). Shute et al. (2017) pointed out that CT and mathematical thinking do have overlapping skills, but they cannot be considered as a subset of each other. Some students related programming to English and General Education.[1] The students explained that English is related because students had to do their programming in an English user interface, and that General Education is related because the process of programming involves creativity and common sense.

Another question asked what the students had learned through programming. The frequencies of answers are provided in Table 12.

Many students indicated that they learnt how to think, solve problems and be creative through programming. Due to the nature of programming, which involves solving computational problems, it is understandable that most students have an impression that problem solving techniques can be learned through programming. These findings roughly correspond to the three constructs used in our quantitative analysis above. The students explained that their creativity was enhanced as they designed and created the game. However, they mentioned creativity more in the pre-interview than the post-interview.

The students also suggested that they learned thinking and problem-solving skills when they tried to fix errors in their code and other technical problems. In particular, thinking and problem solving were much more frequently mentioned in the post-interview than the pre-interview. However, only a few students explicitly used the term ''critical thinking'' (rather than ''thinking'' in general) in their answers. Some skills mentioned by the students are not included due to insignificant pre–post differences. However, the skills learned through CT might go beyond what the students mentioned, because they have limited knowledge of the terminology.

## Discussion

Our quantitative results showed an improvement in the students' creative thinking, critical thinking and problem-solving skills after the intervention. This finding was verified by our qualitative study, which found that students perceived that the programming curriculum not only equipped them with knowledge in programming and computer operation, but also enhanced their thinking skills, problem solving skills and creativity through the process of designing and creating their own games.

Note that the level of creativity expressed by students might depend on the freedom provided in the programming requirements. By relaxing the requirements, students can demonstrate their creative thinking in developing the digital artefacts and programmes to achieve the goals (Denner, Werner, & Ortiz, 2012). However, excessive freedom and flexibility can affect the creativity of students, because they might not be able to achieve the expected outcomes. Considering the results of Denner et al. (2012), the design of programming curricula in primary schools should take this into account to help students to develop even higher creative skills.

Our findings can also be compared to those of Falloon (2016), who studied how junior primary students (age 5 and 6) learned to programme with Scratch Junior on iPad to create subject-

related artefacts (e.g. basic shapes and letters related to mathematics and language). The study was similar to our project in terms of the age group and visual programming tools. Fallon showed that students can develop higher-order general thinking skills in analysis, predictive thinking and evaluation, which are highly related to problem solving and critical thinking in our results.

Falloon's (2016) study also took the researcher interpretation approach to assess the relationship between thinking skills and other subject learning to programming. Our study found similar results by assessing students' perception. In the interview, the students explained that their game construction process involved the use of mathematical concepts. They also required creativity and common sense, which they perceived as related to general education. Finally, some students recognised that the need to work in an English user interface provided a side benefit: they could enhance their English.

We therefore agree with Falloon (2016) that programming should be included in the core curriculum in primary education because it builds cognitive dimensions and have benefits that are not only restricted to programming and computer skills but also other learning competences across a variety of subjects.

Although many scholars have assumed that once learned, twenty-first century skills are transferable to other domains, some authors have challenged this trivial account of transferability. For example, Binkley et al. (2012) comment that in all of the frameworks they reviewed that assess twenty-first century skills, the teaching of these skills has been embedded in the subjects that make up the school curriculum. They question whether these subject-embedded skills (e.g. critical thinking and creativity) have sufficient common features to be transferable from one subject to another.

In our study, the students showed an awareness of the relationship between CT, programming and twenty-first century skills. This could be considered the first step of skills transfer. However, these skills might be specific to the context of solving computational problems and only partially transferable to other subject contexts. For instance, it is uncertain whether primary school students can develop problem solving and creative thinking by acquiring CT and then transfer the skills to solve a geometry problem. A classical work by Halpern (1998) on transferability of critical thinking argues that thinking skills need to be taught in an explicit and conscious approach with examples linked to a specific context. Merely teaching such skills with an assumption that students can recall the skills in different domains is ineffective.

In a more specific context, Kurland, Pea, Clement, and Mawby (1986) found that at the end of a one-year study of high school students learning computer programming, the students' programming experience was not transferred to other subject domains. The authors argued that the students, who were novices in programming, were only able to recognise the surface characteristics of the tasks rather than focusing on the underlying conceptual properties as an expert would. They were unable to transfer the skills to other areas. Therefore, the self-reported questionnaire in our study can be assumed to focus more on programming and CT related problems, rather than perfectly transferable twenty-first century skills. We suggest that a further study could strengthen the concept of transferability of twenty-first century skills developed through CT and programming. For instance, an experimental study could involve control groups focused on a specific domain (e.g. science education) for transfer.

## Limitations and conclusions

A small number of students did not take part in the post-test. This could impose two limitations on the pre–post comparison in this paper. First, this could have introduced bias into the test sample because the absent students might have been less academically oriented. Second, because the questionnaire responses were anonymous, it was impossible to eliminate data from the students who did not take part in both tests. Nevertheless, the response rate was high for both the pre-test and post-test, so the impact on the data should be negligible.

Educators around the world have not yet reached a consensus on how CT should be defined, whether it should be taught through programming in early compulsory education, and how it should be taught. The benefits of CT are yet to be completely realised until purpose-built assessment tools are developed. In summary, this study shows how CT and programming can be linked to relevant soft-skills and their transferability. Future studies should consider developing assessment tools to evaluate how effectively learners develop these skills through CT and programming compared to other non-programming activities.

## Note

1. See http://www.edb.gov.hk/en/curriculum-development/kla/general-studies-for-primary/index.html.

## Disclosure statement

No potential conflict of interest was reported by the authors.

## Funding

This work was supported by Research Grants Council, University Grants Committee [Grant Number General Research Fund, No.#18615216].

## Notes on contributors

*Gary Ka-Wai Wong* is an Assistant Professor of Information and Technology Studies in the Faculty of Education at the University of Hong Kong since 2016. He is currently interested in computational thinking for children, computer-mediated reality for education and integrated learning in STEM (Science, Technology, Engineering, Mathematics) education. He has a BSc in Computer Science and Mathematics (Double Major) from Brigham Young University-Hawaii, MPhil in Electronic and Computer Engineering from the Hong Kong University of Science and Technology, and PhD in Computer Science from City University of Hong Kong. He also received a EdM in Learning Design and Leadership and completing his EdD from University of Illinois at Urbana Champaign. He is the Chair of IEEE Hong Kong Section (Education Chapter) since 2014. His research has been published in journals including IEEE Transactions on Wireless Communications, Australasian Journal of Educational Technology, Educational Technology Research and Development, British Journal of Educational Technology, and Interactive Learning Environments.

*Ho-Yin Cheung* is a researcher at the Faculty of Education, University of Hong Kong. He holds BSc and MPhil degrees in Physics from the Hong Kong University of Science and Technology, and an Ed.D. degree from the University of Bristol, specializing in flipped classroom and active learning approaches. His current research interest includes flipped classroom, programming education, learning approaches and educational technologies. Besides, he is also an experienced teacher and software developer of online educational platforms.

## ORCID

*Gary Ka-Wai Wong* http://orcid.org/0000-0003-1269-0734
*Ho-Yin Cheung* http://orcid.org/0000-0001-8230-5171

## References

Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of Learning Group Publication*, *5*(3), 438. https://www.tandfonline.com/doi/ref/10.1080/00131881.2014.965569

Agogi, E., Rossis, D., & Stylianidou, F. (2014). *Creative little scientist: D 6.6 set of recommendations to policy makers and stakeholders* (p. 64). Retrieved from http://www.creative-little-scienstists.eu

Alimisis, D. (2013). Educational robotics: Open questions and new challenges. *Themes in Science & Technology Education*, *6* (1), 63–71.

Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-Ricci, M., & Rumble, M. (2012). Defining twenty-first century skills. In P. Griffin, B. McGaw, & E. Care (Eds.), *Assessment and teaching of 21st century skills* (pp. 17–66). Cham, Switzerland: Springer International Publishing.

Borgers, N., De Leeuw, E., & Hox, J. (2000). Children as respondents in survey research: Cognitive development and response quality 1. *Bulletin of Sociological Methodology/Bulletin de Méthodologie Sociologique*, *66*(1), 60–75.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Annual American educational research association meeting (pp. 1–25). Vancouver, BC. Retrieved from http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Chow, G. C. C., Chin, M. K., Mok, M. M. C., Edginton, C. R., Li, X. Y., Wong, W. W. S., & Tang, M. S. (2009). Generic skills promotion and the influence of participation of the life-wide learning model: 2008 camp adventure youth services program in Hong Kong. *World Leisure Journal*, *51*(4), 237–251.

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology*, *76*(6), 1051–1058.

Cohen, L., Manion, L., & Morrison, K. (2011). *Research methods in education* (7th ed.). Oxford , UK: Routledge.

Darbyshire, P., Macdougall, C., & Schiller, W. (2005). Multiple methods in qualitative research with children: More insight or just more? *Qualitative Research*, *5*(4), 417–436.

DeJarnette, N. K. (2012). America's children: Providing early exposure to STEM (science, technology, engineering and math) initiatives. *Education*, *133*(1), 77–84.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers and Education*, *58*(1), 240–249. doi:10.1016/j.compedu.2011.08.006

EDB. (n.d.). *Assessment program for affective and social outcomes* (2nd Version) (APASO-II). Retrieved from https://www.edb.gov.hk/en/sch-admin/sch-quality-assurance/performance-indicators/apaso2/index.html

Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. on the iPad. *Journal of Computer Assisted Learning*, *32*(6), 576–593.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers and Education*, *63*, 87–97. doi:10.1016/j.compedu.2012.11.016

Gretter, S., & Yadav, A. (2016). Computational thinking and media & information literacy: An integrated approach to teaching twenty-first century skills. *TechTrends*, *60*(5), 510–516. doi:10.1007/s11528-016-0098-4

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, *42*(1), 38–43. doi:10.3102/0013189X12463051

Halpern, D. F. (1998). Teaching critical thinking for transfer across domains: Disposition, skills, structure training, and metacognitive monitoring. *American Psychologist*, *53*(4), 449–455.

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers and Education*, *82*, 263–279. doi:10.1016/j.compedu.2014.11.022

Jenkins, H., Purushotma, W. R., Weigel, M., Clinton, K., & Robison, A. J. (2009). *Confronting the challenges of participatory culture: Media education for the 21st century*. Cambridge, Massachusetts: MIT Press.

Kong, S. C. (2016). A framework of curriculum design for computational thinking development in K-12 education. *Journal of Computers in Education*, *3*(4), 377–394. doi:10.1007/s40692-016-0076-z.

Kucuk, S., & Sisman, B. (2017). Behavioral patterns of elementary students and teachers in one-to-one robotics instruction. *Computers & Education*, *111*, 31–43. doi:10.1016/j.compedu.2017.04.002

Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research*, *2*(4), 429–458.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. doi:10.1016/j.chb.2014.09.012

MacLaurin, M. (2009). *Kodu: End-user programming and design for games*. Proceedings of the 4th international conference on foundations of digital games (p. 2). ACM.

Manches, A., & Plowman, L. (2017). Computing education in children's early years: A call for debate. *British Journal of Educational Technology*, *48*(1), 191–201.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). *Computational thinking in K-9 education*. Proceedings of the working group reports of the 2014 on innovation 38; Technology in computer science education conference (pp. 1–29). doi:10.1145/2713609.2713610

Miles, M. B., Huberman, A. M., & Saldana, J. (2014). *Drawing and verifying conclusions. Qualitative data analysis: A methods sourcebook* (3rd ed.). Thousand Oaks, CA: Sage.

Mishra, P., & Yadav, A. (2013). Of art and algorithms: Rethinking technology & creativity in the 21st century. *TechTrends*, *57*(3), 10–14.

Moomaw, S. (2012). STEM begins in the early years. *School Science and Mathematics*, *112*(2), 57–58.

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas*. New York: Basic Books.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Silver, J. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142–158.

Stolee, K. T., & Fristoe, T. (2011, March). *Expressing computer science concepts through Kodu game lab*. Proceedings of the 42nd ACM technical symposium on Computer science education (pp. 99–104). ACM.

Tran, Y. (2018). Computational thinking equity in elementary classrooms: What third-grade students know and can do. *Journal of Educational Computing Research*. doi:10.1177/0735633117743918

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, *20*(4), 715–728. doi:10.1007/s10639-015-9412-6

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies*, *22*(2), 445–468. doi:10.1007/s10639-016-9493-x

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. doi:10.1145/1118178.1118215

Wong, G. K., Cheung, H. Y., Ching, E. C., & Huen, J. M. (2015). *School perceptions of coding education in K-12: A large scale quantitative study to inform innovative practices*. Teaching, Assessment, and Learning for Engineering (TALE), 2015 IEEE International Conference on (pp. 5–10). IEEE.

Wong, G., Jiang, S., & Kong, R. (2018). Computational thinking and multifaceted skills: A qualitative study in primary schools. In *Teaching computational thinking in primary education* (pp. 78–101). Hershey, PA: IGI Global.

Wright, G. A., Rich, P., & Leatham, K. R. (2012). How programming fits with technology education curriculum. *Technology and Teaching Engineer*, *71*(7), 3–9.

## Appendix: Structured interview guide

Pre-interview questions

(1) Do you know any software tools for programming?
(2) Have you used Kodu or Scratch before?
(3) What subjects do you think programming is closely related to? Why?
(4) What have you learned from programming and what skills have been enhanced?

Post-interview questions

(1) Do you know any software tools for programming?
(2) How interesting do you think the programming lesson was (0 = Very boring; 10 = Very interesting)
(3) Do you think that the subject and programming are closely related? Why?
(4) What have you learned from programming and what skills have been enhanced?