Competitive Programming in Computational Thinking and Problem Solving Education

Kevin Kam Fung Yuen, Dennis Y. W. Liu, Hong Va Leong
Department of Computing
The Hong Kong Polytechnic University
Hong Kong SAR, China
kevinkf.yuen@gmail.com
{kevin.yuen, ywliu, hong.va.leong}@polyu.edu.hk

***Abstract*— Building practical programming competency requires a long-lasting journey of discovery, trial and error, learning and improvement. This paper presents essential findings of a case study of a Python programming contest with an automatic judgement system for Competitive Programming training extending the learning experiences for students in an introductory course, computational thinking and problem solving. The benefits and challenges are discussed. Due to the COVID-19 epidemic, a hybrid model of the contest was adopted, i.e., some students participated in the contest on-site, while the others participated remotely. To alleviate human effort in judging the submissions, the DOMjudge platform, a web-based automatic judgement system, has been deployed as an online automatic judgement system and contest management in competitive programming. The implementation roadmap and framework were provided. The contest problems and contestants' performance were discussed. Not many junior contestants could solve at least one problem(s), and competitive computing training should be offered if the students are keen on open competitions. An empirical study was conducted to evaluate the student feedback after the contest. Preliminary results revealed that the contest offering the chance to stimulate student learning interests could enhance their independent learning, innovative thinking, and problem-solving skills, and could thus lead to the overall benefits of the learning experiences, which further encourage them to participate in future contests to improve their learning and therefore enhance their employability. Employers often treasure student experiences in competitive programming events, like ACM programming contest, Google Code Jam, or Microsoft Imagine Cup. The sharp vision requiring skills to tackle unseen problems within a short period is also instrumental to students planning for graduate school.**

Keywords*— *Competitive programming, learning technologies, Contest Education, Computational Thinking Education

I. INTRODUCTION

Competitive programming is a coding competition in which the contestants not only solve coding problems concerning the strict requirements specified in the questions but also compete with the others measured by the scores based on time efficiency, correctness, and penalty. [1] listed guides and examples on how to implement and evaluate competitive algorithms in view of efficiency and design.

Competitive programming typically adopts a contest management platform, including the automatic judgement system to evaluate contestant solutions immediately and automatically. Competitive programming with the online judgement systems can be used for education training, organising contests and developer recruitment. [2] surveyed online judge systems and their applications, including competitive programming contests, education and recruitment processes, data mining services, online compilers and development platforms. [3] described the generic architecture design, including clients and servers, as well as implementation based on their institution for the online judge systems.

Competitive programming can be incorporated into teaching practice. For example, [4],[5],[6], and [7] introduced competitive programming with online judge systems to the programming and algorithms courses. [8] used a competitive programming platform as a mechanism for auto-grading assignments for an introductory course on algorithm design and analysis. [9] proposed the use of badges for personalised competitive programming training.

This paper presents a real-life case study with findings for encouraging competitive programming via a Python Programming Contest with an automatic judgement system at the Department of Computing, The Hong Kong Polytechnic University. To summarise, we have made the following contributions:

- Section II presents a literature review for the related study. Section III offers the motivations, limitations and merits to extend the learning experience of the Python Programming Contest into the Computational Thinking and Problem-Solving course.

- The roadmap and framework for creating a Python Programming Contest based on the experiences at the Department of Computing, The Hong Kong Polytechnic University, are offered in Sections IV and V respectively.
- Evaluation and relevant findings are depicted and explained in Section VI. Section VII provides concluding remarks with limitations and future research suggestions.

II. Related Work

A. Contest Formats

Examples of popular programming contests and formats can be referred to as CodeChef [10], HackerRank [11], Facebook Meta Hacker Cup [12], Google Code Jam [13], and International Collegiate Programming Contest (ICPC) [14], previously called ACM-ICPC (1977-2017]). [15] compared several online programming platforms to support teaching computer science disciplines concerning different learning outcomes, and presented an example using HackerRank[11] for the assignments of their programming courses. [16] introduced an educational programming judge system, Jutge.org, where instructors can administer their courses, manage their roaster of students and tutors, add problems, attach documents, as well as create lists of problems, assignments, contests and exams.

In particular, DOMjudge[17] has been used in ACM-ICPC. ACM-ICPC is a team-based competition in which each team is given ten or more problems to be solved within five hours, sharing a single computer. Besides strong programming skills, participants must possess an excellent algorithmic ability to transform the problems into more well-known problems for generating quick solutions. Team members must cooperate closely and make a good division of labour to effectively utilise the single shared computer. Strategies are also required to maximise the number of problems solved within the minimum aggregate elapsed time.

As the format of the contest presented in this paper is similar to that of ACM-ICPC, DOMjudge has been adopted as the contest platform as the primary purpose of the course contest is to arouse the interests of and encourage the students to participate in external contests like ICPC in their further programming and problem solving skill development. Future extension of this study would consider more platforms offering more features to meet different requirements of different courses.

B. Online Judgement Systems

Whilst the limitation of the classical online judgement systems is that grading programs are merely based on the number of test cases passed, regardless of the code quality of programs, [18] presented the framework of the OJ system should contain personalised feedback for students, code quality checking, code similarity checking, and advising on teaching adjustment, but actual implementation was not presented. According to the recent review of automatic feedback in online learning [19], there was evidence that automatic feedback increased student performance in activities, but few papers attempted to analyse the quality of feedback, and there was no strong evidence that automatic feedback could ease instructors' workload. [20] utilised plagiarism detection function in the OJ system as the traditional ones usually do not support. Several approaches, such as the Markov model [21], fuzzy logic [22], Bidirectional Encoder Representations from Transformers (BERT) [23], collaborative filtering [24], clustering and associate rule mining [25], have been proposed to evaluate the difficulty of programming problems, as well as to recommend programming problems in online judge system.

This study did not initially apply the online judgement in general courses but instead focused on the provision of a contest for the first-year students, major in computer science, with an interest to experience, as this study found that only a portion of students are interested in the contest (registration rate is not high). However, some students were highly motivated by this competition and requested us to nominate them to join the university's ACM programming team, to participate in the local contest in Hong Kong and then regional contests in Asia. Recently, the Student Society of Computer Science also shows its strong interest in conducting its own programming contest, to be open to other students in the university. Generally, we believe that the functionalities of the judgement system could be enhanced to support activities other than competitions. As the industrial environment is very competitive, an extension of this study would consider supporting functions in OJ systems to facilitate students to learn programming to gain a competitive edge.

C. Education Impacts

[26] reviewed the most relevant developments of automated assessment for CS assignments, and offered a case study of implementing a web-based automated judgement system to evaluate the educational effectiveness. [27] found that the most frequently cited skills necessary for learning programming were related to problem solving and mathematical ability. Problem-solving was also cited as a learning challenge, followed by motivation and engagement, and difficulties in learning the syntax of programming languages. The main teaching challenges concern the lack of appropriate methods and tools, as well as scaling and personalised teaching. [28] presented an empirical study using Structural Equation modelling to show the results that 'hedonic

motivation', 'self-efficacy', and 'social influence', but academic majors, had the most significant positive effects on students' intention to use the online judge system.

ACM-ICPC competition formulates typical competition problems into computer software design, and tackling competition problems can help train students' practical ability, and cultivate students' creative thinking and teamwork awareness [29]. A study [30] found that student motivations to adopt programming contests like ACM-ICPC are strongly associated with relative advantage, compatibility, ease of use, peer influence, perceived enjoyment and perceived usefulness. Overall, students expressed a positive attitude towards adopting programming contests as it helped improve their problem-solving and programming skills leading to overall employability. In a large-scale introduction programming course, classes adopting automated assessment can help to achieve a higher pass rate than those that did not [31]. One primary reason is the virtually unlimited judging ability of the system to provide timely feedback towards student submissions, especially from those technically weaker but diligent students. [32] analysed the data from the Online Judge system to predict the early dropout for Programming Courses and found that online judges also help in reducing dropout, although there is still a high level of dropout noticeable in introductory programming classes. The findings of educational impacts are presented in Section VII.

III. Competitive Programming for Computational Thinking and Problem Solving

Computational Thinking and Problem Solving (COMP1002) [33] is an introductory course for students with no prior experience in computer programming. It aims to equip students with fundamental computational skills. In particular, the students will learn how to abstract and solve problems and realise the solutions using high-level programming language. The course syllabus is listed below.

- Formulating problems for computers to solve; logically organising and analysing data; representing data through abstractions; automating solutions through algorithmic thinking; implementing efficient solutions; generalising the problem-solving process.
- Computing with numbers and strings; lists and files; functions; decision structures; loop structures and Booleans; sets and dictionaries.
- Problem analysis and design; function abstraction and modularisation; bottom-up and top-down approaches
- Application of computational techniques in different domains.

To offer the opportunity for the students to gain a new learning experience to advance their programming skills in computational thinking and problem solving, a Python programming contest has been organised in the recent two years. However, the registration rate was lower than our expectation, as not all students felt interested in the competition. We observed that the high-performing students tended to join the contest, and the weaker students were already struggling with their daily assignments, lest to spare their time in the competition, for which they did not feel confident.

The internal contest is mainly intended for the extension of the study for those freshmen feeling interested to advance themselves. In addition, through the contest, the elite and interested students can continue to be trained to participate in the open contests like ICPC in the future. Actually, it was one of the goals of this programming contest to nurture future ICPC participants, hence the reason for adopting the automatic judgement system in ICPC.

Initially, we did not choose the online judge systems for our assignment marking due to the reasons that the evaluation is result-oriented, but programming styles cannot be checked, and thus the system cannot check similarity and plagiarism. In addition, very little difference from the fixed output requirements can cause entire errors; for example, a letter case, decimal place, or a space in the output can cause a zero score. This will further lead to students' learning dissatisfaction. However, in the scope of the competition, it demands students to generate the correct program irrespective of the coding style since our contest is patterned behind ICPC.

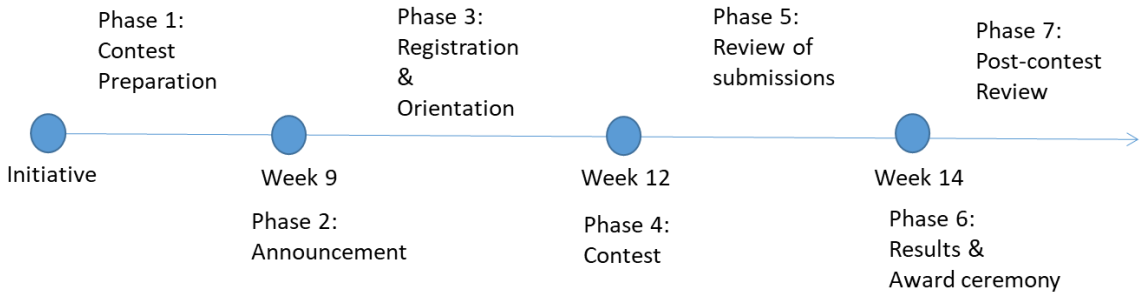

Fig. 1. Key phases in contest Roadmap

Based on the experience of organizing the contest, we plan to introduce more functions of competitive programming into the formal assessments. In particular, a similarity check from student program outputs and correct official outputs could be performed instead of a binary comparison in the automatic judgement mode. For instance, removing blanks, comparing floating-point results within tolerated thresholds, and checking for edit distance differences, could help to produce a partial score to indicate the degree of correctness. Human judgement is made when there is a discrepancy between the student program output and official output, especially when there are multiple possible correct output results, due to difference in tie-breaking criteria. As a competition tool, the judging system will not give the correct answer or the hidden test data to the participants. However, as an educational tool, there is no such a constraint. Students could run the test data on their programs and compare their outputs with the official outputs to gain more insight into the bugs of their program.

IV. Roadmap of Python Programming Contest

A Python programming contest has been organised in the last two years (2020 and 2021). In the first year, students submitted the codes in Blackboard, which were manually marked by the tutors. The marking workload was high since the tutors had to repeatedly run every submission to verify the answer. The most influential factor that affected participants was the lack of instant feedback on program correctness so that they could retry upon a failed attempt. Such a situation occurred in the early days of ICPC, for which teams submitted their solutions on floppy disks to be transported by helpers from teams to judges for evaluation. In the second year, an automatic marking system, DOMjudge [17], was deployed for competitive programming management, with which students could submit their codes and the system would automatically judge and return the verdict whether the results were correct or not.

There are 13 teaching weeks in the computational thinking and problem-solving course. The roadmap for organising the contest during teaching weeks is shown in Fig. 1. The details are provided below.

- Phase 1 - Contest Preparation: before week 9, the initiative for the contest was proposed and approved. The technical team took about three weeks to learn, set up and fully test the DOMjudge server ready for production use.
- Phase 2 - Announcement: in week 9, the announcement of the contest including registration details and prize rewards was released. To encourage students to participate, cash coupons were offered to the Top 10 finalists.
- Phase 3 - Registration and Orientation: in weeks 10-11, each contestant received the Domjudge account information by email. After their registrations were confirmed, a trial run with a tutorial guide for using the system was provided. Student queries and feedback for using the contest system were responded and attended. In addition, the teaching team provided nine problems which were further set up and tested in the system.
- Phase 4 - Contest: the contest commenced in Week 12. During the 1.5 hours of the contest, contestants could submit their codes to and get feedback results from the system. After the contest, the contestants were asked to fill out a survey anonymously for feedback on their contest experiences. The survey results are discussed in Section VII.
- Phase 5 - Review of submissions: although the system can grade the code, it cannot check the similarity and the potential plagiarism. The source codes of the awardees were reviewed to ensure plagiarism-free, and there was no abnormal case. Of course, under the hybrid mode of the contest, we could not verify whether participants had sought help from elsewhere or worked together in a team instead of working on the questions individually. We gave them the benefit of the doubt upon the honour agreement. We believe that a pure face-to-face and laboratory-based competition with access to the Internet disabled will be free of potential irregularities when the pandemic subsides in the future.
- Phase 6 - Results & Award ceremony: in week 14, results were announced. The certificates and prizes ceremony was presented.
- Phase 7 - Post-contest review: the teaching team has reviewed the students' performance and feedback. Whilst experiences can be reused, improvements will be made for the next time this contest is held. Especially, some programming exercises will be installed in the automatic judgement system for the students' learning and training.

V. Framework of Python Programming Contest

The contest is similar to the prestigious ACM International Collegiate Programming Contest (ICPC) [14]. However, intending to strengthen our first-year students' coding ability in Python and cultivate a sense of independent learning among them, our contest is individually based, in which contestants are not allowed to contact or communicate with others. The contestants may use their computers or those in our computer lab, which are loaded with Python 3.6 or above and their choices of program editors. They can pre-load any packages or tools on their computers.

Contestants were given nine programming tasks to solve within 1 hour and 30 minutes. Nine corresponding submission entries, namely T1 to T9, were available in DOMjudge. The contestants were able to attempt the tasks according to their preferred sequence. For each task, the contestants were given a set of sample input/output test data (Fig. 2). The programs were evaluated based on the sample testing data plus some unseen testing data. A program was considered as correct if it produced correct answers to all the testing data sets. Programming styles and comments inside the programs were not considered, and therefore the submitted programs were evaluated solely based on the correctness of the results. In other words, contestants were to program fast but it was not necessary to produce concise or efficient programs.

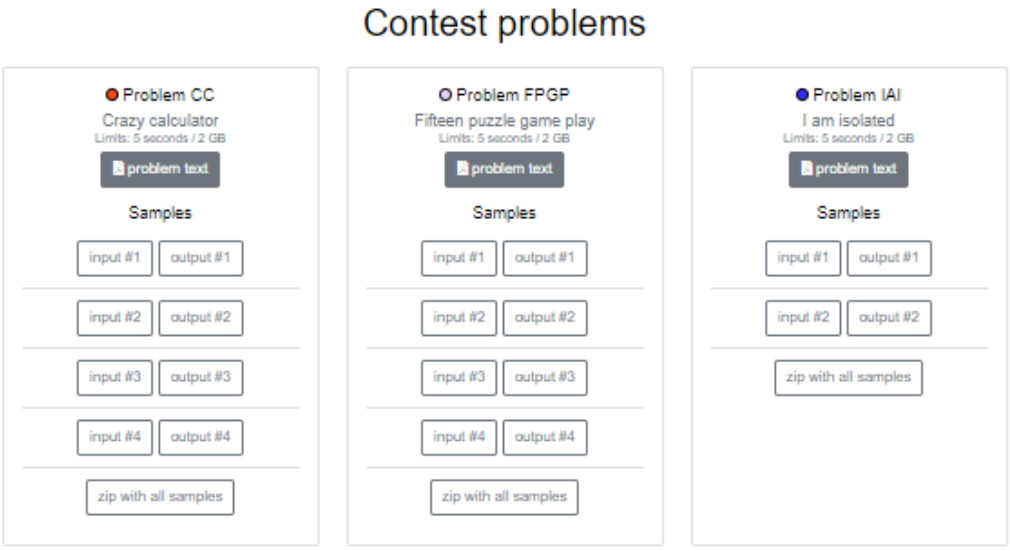

Fig. 2. Screenshot of web UI in the contest system

The champion is the one who could solve the most number of problems. The tie-breaking rule was based on the total time spent on solving the problems correctly, i.e., the sum of the time consumed for each problem solved. The time consumed for a solved problem was the time elapsed from the beginning of the contest to the submittal of the first correct program (called an accepted run in ACM ICPC terminology) plus 20 penalty minutes for every previously "incorrect" program (called a rejected run) for that problem. For example, if a contestant submitted three times for problem T_1, with submission time at 15, 28, and 46 minutes respectively, and answered it correctly on the third version, the total time would be 46 minutes + 20 penalty minutes x 2 = 86 minutes. If a contestant submitted two times for T_2, with submission time at 24 and 31 minutes respectively but did not get it correct, the total time would remain 0 minutes (not solved). If a contestant submitted five times for T_3, with submission times at 28, 43, 87, 88, and 89 minutes respectively, and the fourth version is answered correctly, the total time would be 88 minutes + 20 penalty minutes x 3 = 148 minutes. The fifth version would not be considered since the fourth attempt already solved the problem. Assuming that the contestant made the three sets of submissions to T_1, T_2 and T_3 as above, the total time counted for this contestant for the two solved problems, T_1 and T_3, is 86 +148 = 234 minutes.

VI. Contest Programming Tasks

The programming tasks were set to test the all-aroundness of the fundamental programming techniques of the students, which include standard input/output, control structures utilization, basic data structure manipulation, string processing, and data type conversion. Contestants were asked to write Python programs for nine contest problems in 1.5 hours, briefly introduced below.

A. Contest Problems

1) Phone Bill

The "Phone Bill" (PB) problem requires a program solution that accepts a date, *X*, in *dd/mm/yy* format, and a list of bill entries, until the sentinel *-1* is entered. Next, the total phone bill cost of all the dates with the same month and year of *X* needs to be calculated and all the individual costs to be displayed in 2 decimal places with a given format. Contestants needed to use the loop structure to process the string inputs, parse the data into variables and perform basic mathematical calculations. The difficulty level of this task is medium.

2) I Am Isolated

The "I Am Isolated" (IAI) problem requires a program solution that determines the number of persons isolated within an input of a 10 x 10 map. Here the input map contains only '.' and 'X', where 'X' represents a person and '.' indicates an empty space. The task is to count the total number of 'X' and also the total number of isolated 'X'. 'X' is said to be isolated if all of its left, right, up, down, and diagonal neighbours are empty. Contestants needed to understand the concepts of the 2D array and handle the indexing and boundary cases properly. The difficulty level of this task is low.

3) Points of Triangle

The "Points of Triangle" (POT) problem requires a program solution that computes the maximum area of a triangle that could be formed from an input list of n points, expressed as x, y coordinates. The program should read the list of points sequentially, display the maximum area and the corresponding 3 points forming that triangle. If there are more than one triangles having the same maximum area, only the first one found needs to be reported. Contestants needed to use the loop structure to process the string inputs, perform basic mathematical calculations and make comparisons on the results. The difficulty level of this task is medium.

4) Crazy Calculator

The "Crazy Calculator" (CC) problem requires a program solution that determines the output of a "special" calculator that evaluates an arithmetic expression with the meanings of the four basic arithmetic operators defined by the user. The definition is essentially a change of the arithmetic operations. For example, the "+" operator can be redefined to perform multiplication. The first input line to the program indicates the definitions, and the second input line contains the expression to be evaluated. To align the level of difficulty to freshmen, the four operators are assumed to be of equal precedence, i.e., it is not required to perform multiplication before addition. Furthermore, all the "tokens" are separated by a space. The output is the result of the evaluation of the expression. Contestants needed to parse the input properly and evaluate different parts of an expression sequentially. The difficulty level of this task is medium.

5) Secret Words

The "Secret Words" (SW) problem requires a program solution that inputs a string, S, of random characters (with only lowercase alphabets and then a string of a sentence or sequence of *secret words*, and outputs the starting and ending positions of each *secret word* found in S. A *secret word* is said to be hidden inside S in its order, and it does not necessarily require all the characters of the *secret word* to be immediate neighbours. It can be considered a simplified form of finding a given subsequence in the classical LCS problem. Contestants needed to be handy in string processing, particularly in accessing individual characters. The difficulty level of this task is high.

6) Fifteen Puzzle Game Play

The "Fifteen Puzzle Game Play" (FPGP) problem requires a program solution that verifies an input solution to the classical "fifteen puzzle" game. To solve the puzzle, one needs to move the tiles up, down, left or right from a given random configuration to restore the original configuration. A move is valid only if it involves the movement of a tile next to the empty space to the empty space. The configuration is input as a list of 16 numbers (1-15 represent the tile numbers and 0 represents the empty space) representing the four rows in that order. The program has to verify and print out the validity of each move in a given sequence starting from a given configuration. Also, the program has to print out the starting and final configurations. Contestants needed to logically determine valid moves of a tile and be handy to manipulate a 2D array or other similar data structure to represent the puzzle. The difficulty level of this task is high.

7) Shortest Distance

The "Shortest Distance" (SD) problem requires a program solution that inputs a set of locations, each with an ID, and outputs a pair of adjacent locations of the shortest distance measured by Euclidean distance given in the problem. Contestants needed to understand the Euclidean distance equation and the loop structure to process a list of locations in numbers and strings. The difficulty of this task is low.

8) Symmetric Decryption

The "Symmetric Decryption" (SYD) problem requires a program solution that decrypts the message encrypted by the given function with the symmetric key. An encryption equation is provided. Students needed to understand the encryption equation, formulate the corresponding decryption equation, understand the UTF codes, manipulate strings, and implement the decryption function. The difficulty level is high.

9) Standardization

The "Standardization" (S) problem requires a program solution that gets the numbers in a set X and returns the corresponding standardised values, Z, rounded to 3 digits. Contestants needed to understand the standardisation equations provided, understand the loop structure to manipulate a list of numbers. The difficulty level of this task is low.

B. Contestants Performance

The Freshman Python Programming Contest was organised mainly for around 200 freshmen students taking COMP1002 in 2021. Due to the diverse backgrounds of the students, students were encouraged to register for the contest on the basis of their interests. The contest with an automatic judgement system is mainly used to extend the learning outcomes of COMP1002. In response, 48 contestants registered. As the participation rate is just 23.4%, it may be the right decision that we did not immediately introduce automatic judgement system for assignment grading in this stage, since we can cope with the judging load with experience drawn from our past involvement in ACM as ICPC judge. However, with anticipated growth in a number of participants in future, we plan to enable the automatic judgement system for part of assignment grading to enhance the students' learning experiences in the next phase.

Fig. 3. Contestants' Performance with at least one problem(s) solved

After the contest, a review of the source codes of the proposed awardees was performed and no abnormal case was found. Fig. 3 shows contestants' performance with at least one problem(s) solved. Although a trial run with some past contest problems was offered for the contestants to learn and experience, only 13 contestants (27%) could solve at least one problem(s). The top one could solve six problems and the second one could solve four problems. From those who could solve three problems, the contestant rank was further based on the score measured by the submission times and the number of failed attempts, following the ICPC rules mentioned in Section V.

There were 29 correct solutions (27%) from 106 submissions. Fig. 4 shows the total number of submissions over time during the contest. Complier-error and run-error submissions are not the major problems, but the wrong-answer submissions, i.e., the output results are not perfectly matched with the correct results in the system. For example, errors can be due to a slight incorrect output format such as the number of decimal places, a space, a break line and an upper or lower case. Fig. 5 shows that the CC problem had most attempts, which is the left-most problem shown on the web UI in Fig. 2.

It is realized that the first problem attempted by the contestants had a high fail rate (77%), and it is less predictable on their choice of problem as the second one. Since different problems are of different degrees of difficulty levels as mentioned in Section VI, contestants should browse all the problems and choose the most comfortable one to tackle. For example, the difficulty level of the Standardization (S) problem is considered the lowest, but it received a low attempt rate and a low acceptance rate, possibly due to insufficient time to digest the output format requirement, or due to its being placed as the nineth (last) question, or perhaps misjudging it as a hard problem with the mathematical notations. In contrast, the crazy calculator looked similar to what they experienced in daily life and tended to attempt it.

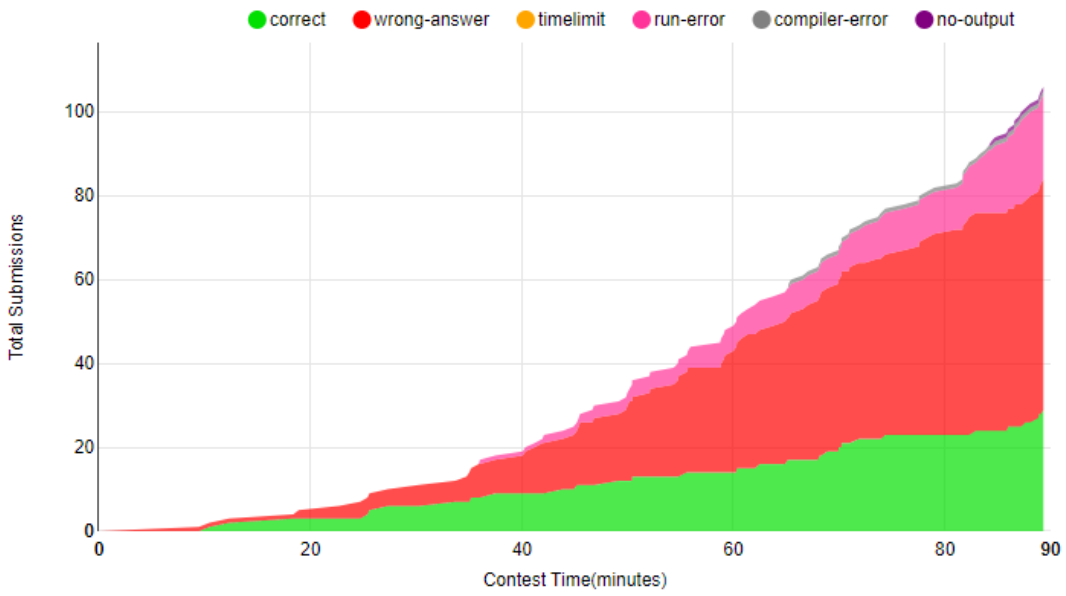

Fig. 4. Total submissions over time during the contest

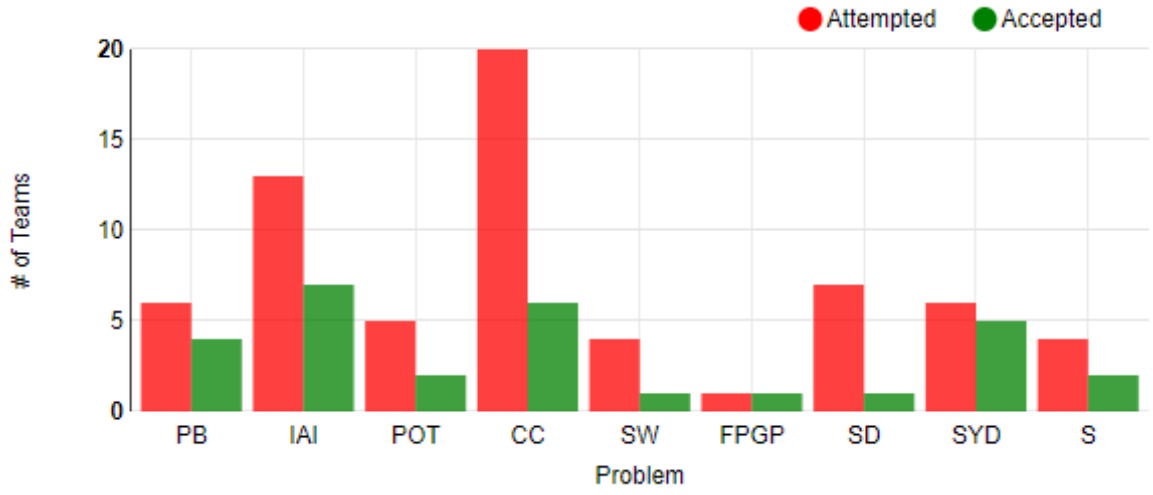

Fig. 5. Number of problems attempted and solved in the contest

VII. Evaluation and Discussion

An online anonymous survey after the contest was distributed to all 48 contestants, and 19 responses were received. The survey investigates the background of students' programming education and contest experiences, familiarity with the use of the system, learning experiences and motivation for the contest. The survey is based on a 5-point Likert scale, where 1 means strongly disagree and 5 means strongly agree. The summary of the survey questions is shown in TABLE I. As only 13 contestants (27%) could solve at least one problem, some anonymous responses might include those students who could not solve at least one problem and might feel dismayed. That is why some items in the survey received portions of not very positive results. The survey findings are further analysed and discussed in the sections below.

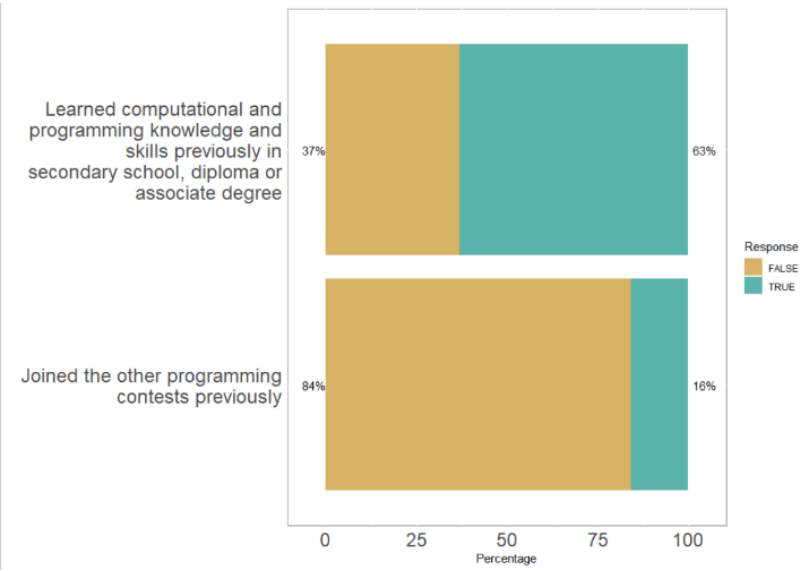

Fig. 6. Responses for education background and contest experience

A. Programming education background and Contest experiences

Before taking this course, more than half of the contestants (63%) expressed that they had learned some basic computational and programming knowledge and skills in (post-)secondary school, but the majority (84%) did not participate in programming contests previously. The responses for programming education background and contest experience are shown in Fig. 6.

TABLE I. SUMMARY OF THE SURVEY QUESTIONS

Sections (scale)	Questions		Mean	*SD*
Background (No(0) or Yes(1))	bg1	I have learned computational and programming knowledge and skills previously in secondary school, diploma or associate degree.	63%	-
	bg2	I joined the other programming contests previously.	16%	-
System Familiarity (1-5)	sf1	The user guide is clear for me to be familiar with the contest system.	3.63	1.3
	sf2	The description of the contest problems is clear.	3.47	1.26
	sf3	I can understand the contest problems.	3.47	1.31
Learning Experiences (1-5)	le1	I have learned something new and interesting from this contest.	3.74	1.19
	le2	For this python programming contest, I can apply what I have learned from COMP1002.	3.68	1.34
	le3	For this python programming contest, I can apply what I have learned from the other courses.	3.42	1.54
	le4	The contest is a good opportunity for my independent learning.	4.21	1.08
	le5	The contest provides good exercises to improve my problem-solving skills.	4.21	1.08
	le6	The contest provides good exercises to practise innovative thinking.	4.16	1.07
	le7	The contest stimulates my learning interest.	4	1.05
	le8	Overall, I think this contest is beneficial to my learning.	4.16	1.07
	le9	After this contest, I will continue to join the other programming contests.	4.05	1.08
Motivations (1-5)	mv1	Improving my learning	4.11	1.05
	mv2	Prize	3.53	1.35
	mv3	Peer influence	3.00	1.29
	mv4	Lecturer encouragement	3.47	1.17
	mv5	Preparation for my future open contests	3.53	1.26
	mv6	Enhancement of my future employability	3.95	1.08

To provide some background, students taking this course are quite diversified. About 80% of them studied the high school diploma, called DSE (similar to GCE, IB, or AP), and about half of them selected the Information and Communication Technology subject in their DSE. The remaining 20% mostly are students possessing a sub-degree (e.g., associate degree or higher diploma). The sub-degree students with excellent academic results are admitted into year 3 directly of the curriculum and will not take this course, whereas those not-as-good-ones and those not studying IT in their sub-degree have to take this course. While local students form the majority, there are also students coming from overseas, e.g. Korea, Indonesia, Kazakhstan, and Pakistan. Students could apply for course exemption to the department if they are able to provide sufficient proof of competency.

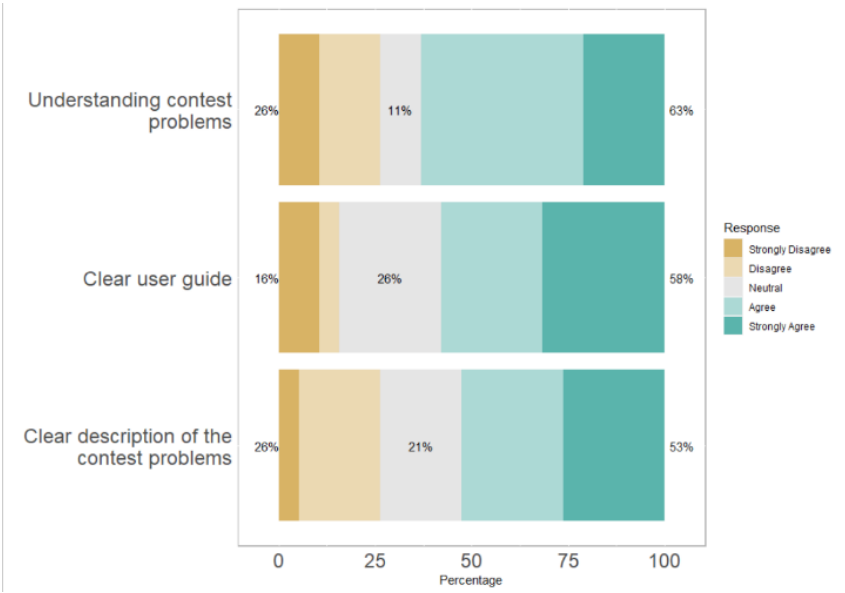

Fig. 7. Responses for System Familiarity and problems understanding

B. Orientation and System Familiarity

The responses for the familiarity of the system and understanding of the contest problems are shown in Fig. 7. Before the contest, a trial run was organised for the contestants to be familiar with the system. A user guide was distributed to the students. More than half of the students (58%) said the user guide was clear. After the contest, more than half of the students agreed that they could understand the contest problems (63%) and that the description of the contest problems was clear (53%). Those contestants feeling neutral or disagree may have carried over their feeling upon their inability to solve the problems correctly, even though they could understand the problem description.

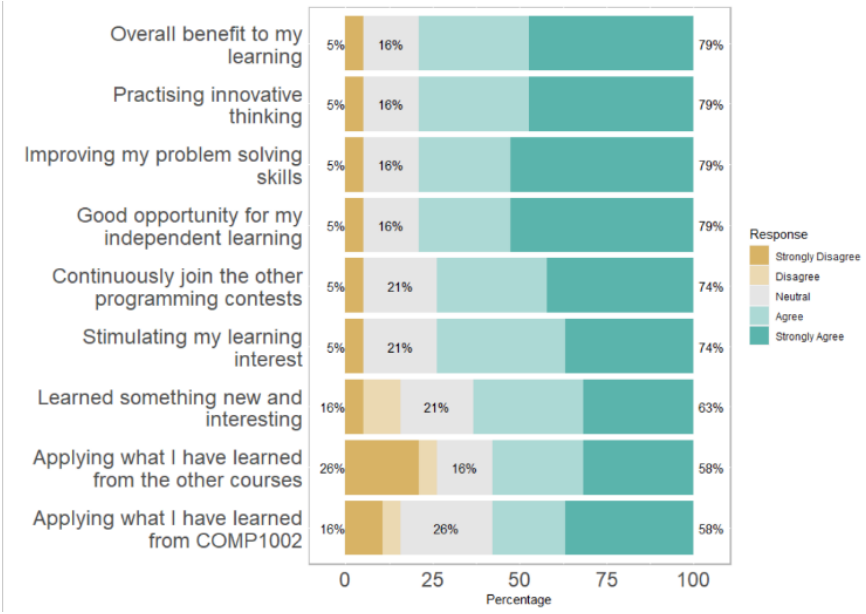

Fig. 8. Responses for students' learning experiences through the contest

C. Learning Experiences

Fig. 8 shows the responses for students' learning experiences through the contest. The majority of contestants (79%) agreed that the contest offered overall benefits for their learning. In particular, the majority agreed that the contest could provide a good opportunity for their independent learning (79%), practise their innovative thinking (79%), improve their problem-solving skills (79%), stimulate their learning interests (74%), learn something new and interesting from this contest (63%). More than half agreed that they could apply what they had learned from this course to the other courses (58%), as well as they will continue to join the other programming contests (74%).

D. Motivations

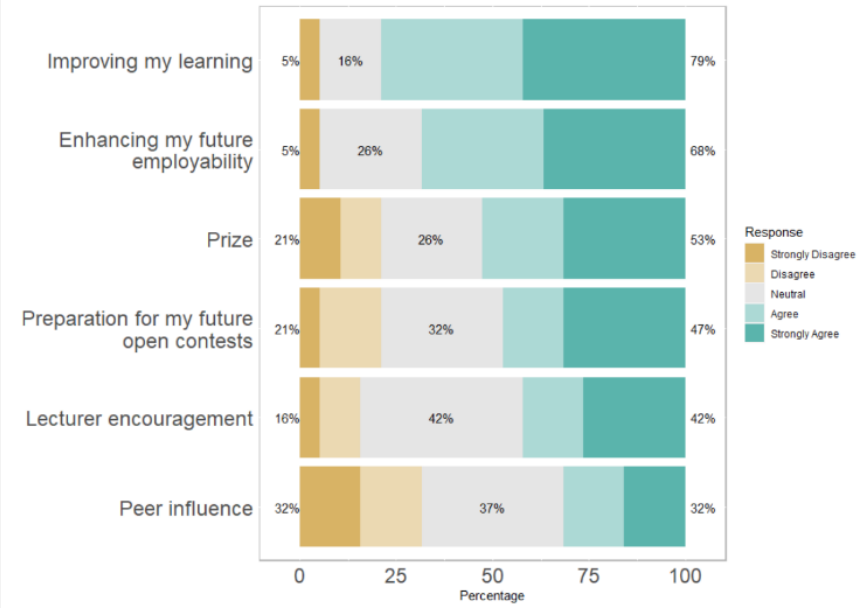

Fig. 9. Responses for students' motivations for the contest

Fig. 9 shows the responses for students' motivations to the contest. The top motivation for the students to join the competition is to improve their learning (79%), followed by enhancement of their future employability (68%), prize (53%), preparation for upcoming open contests (47%), lecturer encouragement (42%), and finally pear influence (32%). Surprisingly, the prize is not the top factor for the contestants, so the organiser may not require high cash value for the prize to attract the contestants (winners may feel very happy even with only award certificates), and the operation cost of the contest could be lower.

TABLE II. SUMMARY OF MODEL FITS FOR THREE SEM MODELS

	Cut-offs	Fig. 10	Fig. 11	Fig. 12
p(Chi2)	>=0.5	-	0.54	0.37
P(Baseline)	<=0.05	0.00	0.00	0.00
CFI	>0.9	1.00	1.00	1.00
NNFI/TLI	>=0.9	1.00	1.06	0.99
RMSEA	<=0.08	0.00	0.00	0.07
P(RMSEA)	>=0.05	-	0.56	0.42
GFI	>=0.9	1.00	0.97	1.00
AGFI	>=0.9	1.00	0.86	0.99
SRMR	<=0.8	0.00	0.03	0.02

E. Path Analyses

On the basis of the descriptive analysis from the sections above, path analysis based on Structural Equation Modelling (SEM) is conducted to analyse the relationship between different learning factors. Three SEM models are constructed, and their model fits are shown in TABLE II. All models have passed the cut-offs of diverse fit indices of SEM. All estimates of latent variables, regressions, and variances shown in the path diagram are statistically significant.

1) Prize and COMP1002 knowledge for contest motivations

Fig. 10 shows the SEM paths for the prize and applying COMP1002 toward two contest motivations of improving learning and employability, i.e., *mv1* + *mv6* ~ *mv2* + *le2*. The prize (*mv2*) and COMP1002 knowledge (*le2*) have a statistically significant regression relationship ($p < 0.05$) to the participating motivations with learning improvement (*mv1*) and employability enhancement (*mv6*).

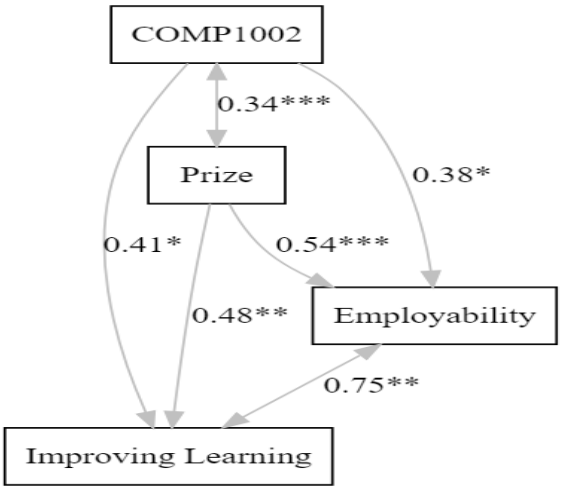

Fig. 10. Prize and COMP1002 for contest motivations based on employability and improving learning

2) Factors on understanding the contest problems

Fig. 11 shows the regression relationship of a construct of their measurable variables on understanding the contest problems. The contestants, who are with improving learning motivation (*mv1*) and can apply the courses knowledge and skills learned from COMP1002 (*le2*) and other courses (*le3*) to the contest problems, exhibit statistical regression relationship on positively understanding the contest problems (*sf3*).

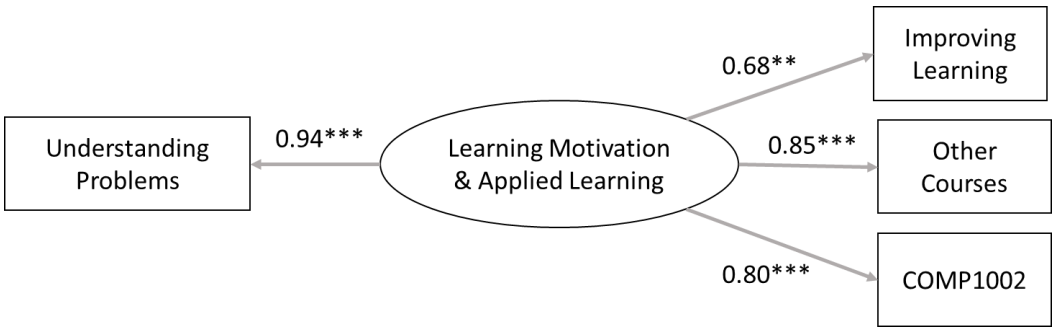

Fig. 11. Factors for understanding contest problems

3) Motivations and learnings

Fig. 12 shows the path analysis for motivations and learning factors. All relations established in the SEM diagram are statistically significant. The contest offering the chance to stimulate their learning interests (*le7*) has a strong correlation to their independent learning (*le4*) and significant relationship to innovation thinking (*le6*), leading to significant relationship to Problem-solving skills (*le5*), which is significantly related to the overall learning benefits (le8). When students find overall learning benefits, they normally are keen on participating in contests (le9) to improve their learning (*mv1*) and finally enhance their employability (*mv6*). The attitude of independent learning (*le4*) has a positive effect on the innovative thinking (*le4*), problem-solving (*le6*) and overall learning benefits (*le8*).

4) Implications and future studies

On the basis of the path analyses, several implications may be concluded as follows.

Fig. 10 shows that prize motivation and the course COMP1002 is significant to the motivations of enhancing employability and improving learning. The course offers a contest with prize. Increasing the prize categories and values for the contest should attract more students of the motivations of enhancing employability and improving learning to join. As students think that the

skills and knowledge learned from the course can be used for contest, students may think that the contest can help them to practice and prepare to get better results in the examination. Future study may investigate the examination factor as a motivation.

Fig. 11 shows that the students showing learning motivation and applied learning have a positive relationship with understanding the contest problems. However, only 13 contestants (27%) could solve at least one problem(s). The future study should investigate the strategies and difficulties of the students to solve the problems. For example, how many questions are viewed by the students? How do students choose the questions to start? What are the gaps between the students and teachers to perceive the difficulty levels?

Fig. 12 shows that running the contest can stimulate students' learning and help improve students' independent learning, innovative thinking, and problem-solving skills, which further leads to overall benefit, which encourages students to participate in the other contests and improving learning and finally benefits their employability. Obviously, incorporating contest with automated judgement system into the course curriculum can enhance student learning engagement. However, currently we only applied the automated judgement system for one contest during the course. It is possible to enhance the system for non-contest purposes, for example, self-practise based on problem recommendations, and course assessments, to have the better learning environment to simulate student learning. Future study may address these issues.

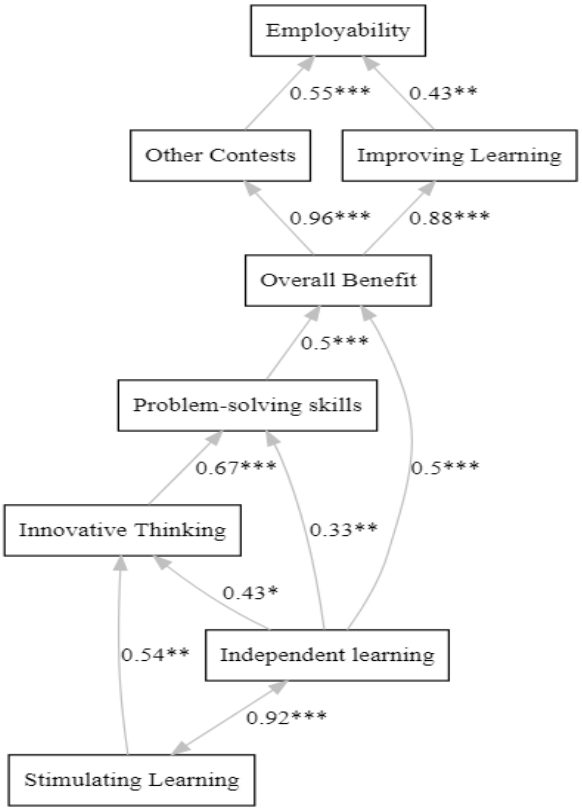

Fig. 12. Path analysis for motivations and learnings

VIII. CONCLUSION

This paper provides a structured framework and recommends practical principles and guidelines for incorporating programming contests into computational thinking education. A real-life case study has been demonstrated. The roadmap shows how the contest was prepared and organised along with a 13-week course, computational thinking and problem-solving. The framework contains the contest deployment and judgement rules.

Contest problems are introduced, and contestant performance is analysed and discussed. It is found that the freshmen contestants very likely attempt the first problem at the beginning without examining the difficulty level of each problem in advance. They may be under an assumption that the first problem is often the easiest, much like that the first lecture is the easiest within a course. In fact, some problems considered as of low difficulty level receive fewer attempts than those with higher

difficulty level, implying imperfect judgement and strategy in competition preparation. Anonymous responses from the online survey were received and analysed. Several statistically significant patterns have been observed as below.

- The course and the prize are incentives to encourage the student to participate in the contest.
- The contestants, who are motivated by self-improvement and proficient in applying the knowledge and skills learned from the computational thinking and problem-solving course and other courses, usually show a positive relationship with understanding the contest problems.
- The contest can positively stimulate students' learning and help improve students' independent learning, innovative thinking, and problem-solving skills, which further leads to enhancing their employability.

Regarding the limitation of this study, as the participation rate among the freshmen was relatively low, it is suggested that the automatic judgement system is used for part of assignment grading in the course to enhance the students' learning experiences to promote the competitive computing culture before the contest starts. As automated judgement systems are used in the recruitment and training in many enterprises, students should be well prepared for the competitive business environment. It is necessary to introduce competitive computing education. However, it appears that there are insufficient supporting case studies for the introduction and promotion of competitive computing culture in computational thinking and problem-solving courses in the tertiary education sector. The provision of this study could address this issue. As the contest conducted at Department of Computing, Hong Kong Polytechnic University resembles the prestigious ACM-ICPC contest, the experiences shared in this paper can be used for the consideration of the other universities in the other countries to integrate the contest into their course curricula to enhance student learning experiences.

On the basis of this research, possible future studies are suggested as follows. As not many junior students could answer the questions well, the behaviour about how the students choose the questions and the perception of how students and teachers estimate the difficulties of the questions will be studied. Therefore, the guidelines on how to produce more appropriate questions in some preferred ordering could be given. The future studies may include an investigation of how the addressed relationships in this study can be strengthened by integrating automated judgement system, with more features, including plagiarism detection and problems recommendation, into course assessments and exercises, and the programming contest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] Laaksonen, A., *Guide to competitive programming*. 2020: Springer.

[2] Wasik, S., et al., *A Survey on Online Judge Systems and Their Applications.* ACM Comput. Surv., 2018. **51**(1): p. Article 3.

[3] Watanobe, Y., et al., *Next-Generation Programming Learning Platform: Architecture and Challenges.* SHS Web Conf., 2020. **77**: p. 01004.

[4] Bandeira, I.N., et al. *Competitive programming: A teaching methodology analysis applied to first-year programming classes*. in *2019 IEEE Frontiers in Education Conference (FIE)*. 2019.

[5] Zheng, Y. and M. Sarem. *A New ACM/ICPC-Based Teaching Reform and Exploration of "Design and Analysis of Algorithms"*. 2014. Dordrecht: Springer Netherlands.

[6] Zheng, Y. and M. Sarem. *A Novel C++ Teaching Method Based on Game Mode and ACM/ICPC*. in *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*. 2018.

[7] Wang, G.P., et al., *OJPOT: online judge & practice oriented teaching idea in programming courses.* European Journal of Engineering Education, 2016. **41**(3): p. 304-319.

[8] Coore, D. and D. Fokum, *Facilitating Course Assessment with a Competitive Programming Platform*, in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 2019, Association for Computing Machinery: Minneapolis, MN, USA. p. 449–455.

[9] Mascio, T.D., L. Laura, and M. Temperini. *A Framework for Personalized Competitive Programming Training*. in *2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET)*. 2018.

[10] Codechef. *https://www.codechef.com.* Accessed on July 2022.

[11] Hackerrank. Available from: *https://www.hackerrank.com/*. Accessed on July 2022.

[12] Facebook *Meta Hacker Cup*. Available from: https://www.facebook.com/codingcompetitions/hacker-cup. Accessed on July 2022.

[13] Google. *Code Jam*. Available from: https://codingcompetitions.withgoogle.com/codejam. Accessed on July 2022.

[14] *ACM/ICPC*, *Association for Computing Machinery – International Collegiate Programming Contest.* Available from: *https://icpc.global.* Accessed on July 2022.

[15] Zinovieva, I.S., et al., *The use of online coding platforms as additional distance tools in programming education.* Journal of Physics: Conference Series, 2021. **1840**(1): p. 012029.

[16] Petit, J., O. Giménez, and S. Roura, *Jutge.org: an educational programming judge*, in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. 2012, Association for Computing Machinery: Raleigh, North Carolina, USA. p. 445–450.

[17] Domjudge. *Domjudge Programming Contest Jury System*. Available from: Accessed on July 2022.

[18] Zhou, W., et al., *The framework of a new online judge system for programming education*, in *Proceedings of ACM Turing Celebration Conference - China*. 2018, Association for Computing Machinery: Shanghai, China. p. 9–14.

[19] Cavalcanti, A.P., et al., *Automatic feedback in online learning environments: A systematic literature review.* Computers and Education: Artificial Intelligence, 2021. **2**: p. 100027.

[20] Iffath, F., et al., *Online Judging Platform Utilizing Dynamic Plagiarism Detection Facilities.* Computers, 2021. **10**(4): p. 47.

[21] Zhao, W.X., et al., *Automatically Learning Topics and Difficulty Levels of Problems in Online Judge Systems.* ACM Trans. Inf. Syst., 2018. **36**(3): p. Article 27.

[22] Intisar, C.M. and Y. Watanobe, *Cluster Analysis to Estimate the Difficulty of Programming Problems*, in *Proceedings of the 3rd International Conference on Applications in Information Technology*. 2018, Association for Computing Machinery: Aizu-Wakamatsu, Japan. p. 23–28.

[23] Pereira, F.D., et al., *Towards a Human-AI Hybrid System for Categorising Programming Problems*, in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 2021, Association for Computing Machinery: Virtual Event, USA. p. 94–100.

[24] Yera, R. and L. Martínez, *A recommendation approach for programming online judges supported by data preprocessing techniques.* Applied Intelligence, 2017. **47**(2): p. 277-290.

[25] Rahman, M.M., et al. *A Novel Rule-Based Online Judge Recommender System to Promote Computer Programming Education*. 2021. Cham: Springer International Publishing.

[26] Paiva, J.C., J.P. Leal, and Á. Figueira, *Automated Assessment in Computer Science Education: A State-of-the-Art Review.* ACM Trans. Comput. Educ., 2022. **22**(3): p. Article 34.

[27] Medeiros, R.P., G.L. Ramalho, and T.P. Falcão, *A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education.* IEEE Transactions on Education, 2019. **62**(2): p. 77-90.

[28] Arifin, J. and R.S. Perdana. *UGrade: Autograder for Competitive Programming Using Contestant PC as Worker*. in *2019 International Conference on Data and Software Engineering (ICoDSE)*. 2019.

[29] Luo, Y. and H. Zheng. *An Innovative Teaching Mode Based on Programming Contest*. in *Application of Intelligent Systems in Multi-modal Information Analytics*. 2021. Cham: Springer International Publishing.

[30] Raman, R., H. Vachharajani, and K. Achuthan, *Students motivation for adopting programming contests: Innovation-diffusion perspective.* Education and Information Technologies, 2018. **23**(5): p. 1919-1932.

[31] Zampirolli, F.A., et al., *An experience of automated assessment in a large-scale introduction programming course.* Computer Applications in Engineering Education, 2021. **29**(5): p. 1284-1299.

[32] Pereira, F.D., et al. *Early Dropout Prediction for Programming Courses Supported by Online Judges*. 2019. Cham: Springer International Publishing.

[33] *COMP1002:Computational Thinking and Problem Solving* Available from: https://www.polyu.edu.hk/comp/docdrive/ug/subject/COMP1002.pdf. Accessed on July 2022.