An Applied C Programming Exercise with Card Game Strategy and Analysis of Codes by a Grouping of Score and Code Metrics

Takeru Shimizu
Graduate School of Engineering
Kagawa University
Takamatsu, Japan
s19g460@stu.kagawa-u.ac.jp

Reo Ishii
Graduate School of Engineering
Kagawa University
Takamatsu, Japan
s18g455@stu.kagawa-u.ac.jp

Hiroyuki Tominaga
Faculty of Engineering and Design
Kagawa University
Takamatsu, Japan
tominaga@eng.kagawa-u.ac.jp

***Abstract*—We have proposed an applied C exercise with the subjects of card game Poker strategy. We have operated contest management server WinT and provided execution environment. Learners implement their strategies aiming efficient poker's hands of cards with estimating of remaining deck and weighted rate. In this paper, we consider a method of personalizing advice for students. For this purpose, we use the K-means method to group students, analyze the feature of each group, and consider feedback strategy. We aim to reduce the burden on teachers and TAs and provide higher quality advice by the support system automatically performing this method.**

***Keywords*— C programming exercise, card game strategy, contest style, code metrics, K-means clustering**

I. Introduction

On the programming exercise in the university's information engineering field, there are two matters: a programming skill as a means of problem solving with an algorithmic approach and an attitude for programming as a task of continuous improvement based on software development methods. As the subject of the exercises to learn these, the development of an information system that simulates actual work is often mentioned. However, it is difficult for students to set a problem due to the lack of knowledge and social experience of students. Because of this, the programming exercise have difficulty making progress in problem solving practice.

In our laboratory, we practice programming exercises on game strategy from the field of knowledge information processing. We have proposed two exercises with board games and card games. We provide execution environments and operate contest management servers. Students submit the created game strategy program to the contest management server. We set the period of preliminary league and accept the submission by learners at many times. We publish the learner's score and ranking. This encourages improvements through strategy review and continual code modifications, allowing them to learn their attitude towards programming. The strategy code in these exercises is likely to be unique to each student. Therefore, it is desirable that advice for students be personalized according to code metrics, execution performance, and the like.

In this paper, we focus on WinT that is an exercise on card game strategy [1]. We group strategies in terms of code metrics and execution performance and analyze the feature of the code by each group. From these, we will consider the feedback strategy for each group.

II. Programming Exercise with Poker Strategy

We have provided a score-based exercise based on the card game poker. This exercise has been practiced since 2010. The player makes one of nine types of hand while exchanging five cards in player's hand with a deck. The score of hands is defined according to the degree of difficulty of implementation as a program. Also define the number of changes in a take. The strategy is to repeat the take with a deck fully shuffled.

We have introduced weighted rates in each take. The weighted total score of the hand made in each take is taken as the raw score in that deck. The score of the strategy is the average of the score by the deck of a random considerable number. The regulation for the number of changes, takes, and the rate of weighted are set annually. The strategy needs to aim for a hand that takes into account the status of the remaining decks and the weighted rate (Fig. 1) [2].

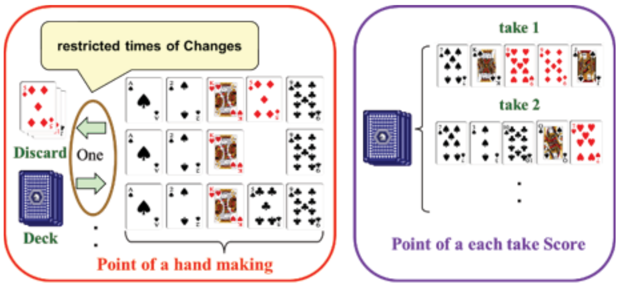

Fig 1. Outline of WinT

III. Contest Management and Support System

We hold a contest where all students compete for scores. For the contest support, we have developed the contest management server WinT. Since 2013, we have operated in the actual exercise.

Learners download the execution environment and create a program of strategy on their own local PC. WinT accepts submission of the created strategy program, compiles it on the server side, and saves the execution result in the strategy DB. For all the strategies, the poker is executed using 10000 decks that prepared in advance.

WinT publishes score, rankings and code metrics such as Q-ABC to individuals and everyone. Learners will use this information to review their strategies and modify them accordingly. WinT motivates continuous code correction by always giving learners the opportunity to evaluate their strategies. In past practice, presentation of Q-ABC size has been found to be effective for student's code improvement activities [3].

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

Authorized licensed use limited to: Norges Teknisk-Naturvitenskapelige Universitet. Downloaded on March 20,2022 at 16:16:06 UTC from IEEE Xplore. Restrictions apply.

At the end of the contest period, the final result will be the best scoring strategy among the submitted strategies. In the grade evaluation, the final result is targeted.

IV. Learning Objective and Evaluation

Table1 shows the expected learning outcomes from the WinT exercises. As a C language learning outcome, learners can expect to learn how to use array operations, array element pattern matching, and random number functions. Specifically, these are learned by managing and rearranging the cards. In terms of algorithm, it is expected that the learner learns a simulation method using sorting, recursion, and random numbers. Grammatically, even beginners of C language can challenge. As software development methods, it is expected that learners will learn functional design with good outlook, software testing, debugging, parameter adjustment efficiency, and version control. In addition, learners can expect to acquire refactoring skills by responding to the enlargement of code by expanding strategies.

TABLE I. Summary of Features Evaluated by System

Matters	Items	In Exercise
C language	Array operation	Card management
	Pattern Matching	Recognition of hand
	Random Number	Simulation of hand
Algorithm	Sort	Arrangement of hand
	Recursion	Look ahead of hand
	Simulation	Simulation of hand (ex. Monte-Carlo simulation)
Software Development Method	Function Design	Using auxiliary functions, Consider reusability
	Test	Verification of strategies using decks
	Debug	Confirming the effectiveness of strategy improvements
	Parameter Adjustment	Parameter adjustment to improve strategy performance
	Version Control	Management of strategy enhancement and revision
	Refactoring	Responding to code enlargement due to strategy enhancement

V. Practice Result of Exercises of Each Year

We discuss the class practice of each year in WinT. In this paper, we focus on 2016 to 2018. Table 2 shows the regulations for each year. Table 3 shows the outline and results of class practice in each year. The exercises were carried out approximately 6 to 7 weeks each year. The number of participants increased in 2018 compared with the average year. This is because there were a lot of students that repeating the same class. Next, we will describe the results of the practice especially on the number of strategies. The total number of strategies in 2018 is larger than in other years. This is because one student of the relevant year submitted a large number of submissions. This student submitted 824 submissions during the contest. This student's submission status is unusual and does not match the general submission pattern. Therefore, in this paper, this student was excluded from the analysis. The average year is about 100 submissions at most. The median number of strategies is relatively high in 2017.

TABLE II. The Guideline of Exercise

Year	Change	Take	Weighted Rate					
2016	6	5	1.5	1.5	1.0	2.0	2.0	-
2017	5	6	1.0	1.5	2.0	1.0	1.5	2.0
2018	6	5	1.5	1.5	1.0	2.0	2.0	-

TABLE III. Summary and Result of Practice of Each Year

		2016	2017	2018
Contest	Term	7 weeks	6 weeks	5 weeks
Participants	Learners	45	44	52
Strategy	Total	1215	1272	1912
	Max	128	84	824

VI. Evaluation of Strategy Codes by Code Metrics

We have proposed Q-ABC size to evaluate the quality of WinT strategy code. Q-ABC size is a code metric that integrates the ABC size obtained for each function. ABC size was developed to overcome the weakness of line of code (LOC) and represent a more generalized function size indicator. This size is a value calculated by the equation of Fig. 2 for each function for assignment to variables (Assignment), function call (Branch), and conditional statement (Condition). ABC stands for the initials of these three metrics [4].

In LOC, even programs with the same function show very different values depending on the coding style, so it was not enough to measure the scale of the software. On the other hand, the ABC size can show the same value for programs with the same function. For this reason, it is suitable for measuring the scale of a program. Also, in general, a program that is too large is likely to have a responsibility that is too large. Therefore, by observing the ABC size, it is possible to estimate a program with too much responsibility. It is also suggested that programs with too much responsibility will have a lower cohesion. In general, programs with low cohesion are difficult to understand and maintain. These programs have low module reusability and are not desirable from a software development perspective. In this way, programs with high ABC sizes often show an undesirable trend from the perspective of program development and maintenance. On the other hand, a program that shows a suitably low ABC size is highly cohesive, and is more likely to be communicative, sequential, and functional. These are preferred for development and maintenance.

However, ABC size is a value calculated for each function in C language. We have proposed a Q-ABC size in WinT to show more summarized values to students. This is a value that indicates the overall trend of the submitted strategy code. Q-ABC size is calculated by the equation of Fig. 3. The parameter N is the number of functions. The parameter x is the ABC size of each function. A high Q-ABC size indicates that there is a function with a high ABC size. In other words, it suggests that there is a function with low cohesion. Therefore, it is often

Authorized licensed use limited to: Norges Teknisk-Naturvitenskapelige Universitet. Downloaded on March 20,2022 at 16:16:06 UTC from IEEE Xplore. Restrictions apply.

necessary to improve the program. On the other hand, if the Q-ABC size shows a reasonably low value, the cohesion of the strategy code is likely to remain high. As a reasonably low value, we are referring to the 15 or less adopted by Rubocop [5]. Here, as an example of a program that showed a high Q-ABC size, the snippet of the strategy with the highest Q-ABC size in the 2018 exercise is shown in Fig. 4. This snippet is for one function of the strategy code. Many functions are described in one function, and cohesion is low. Other high Q-ABC size programs also show signs of code smell, such as duplicate code and methods that are too long. On the other hand, these signs are not seen for programs with a Q-ABC size of 15 or less. For this reason, strategy codes that exhibit high Q-ABC sizes need improvement.

We provide instruction from human teachers for strategy codes that indicate a particularly high Q-ABC size. As for the content of feedback, we are teaching from the method of modularization of duplicate processing. In particular, it focuses on organizing duplicate processes in the strategy code. However, some students may need to review how to use and declare functions. Generally in WinT, students whose Q-ABC size were high but exceeded the average score seemed to improved their cohesiveness by modularizing the overlapping processes. Students who write these strategies often have a good understanding of algorithms and programming grammar. However, there are many cases where proper design, which is one of the software development methods, is not possible. If the score is low, there are many students who have insufficient understanding of the basics of programming, especially grammar. Therefore, the feedback will be a review of how to use and declare functions.

$$ABC\ Size = \sqrt{A^2 + B^2 + C^2}$$

Fig 2. Calculation formula of ABC size

$$Q\text{-}ABC = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

Fig 3. Calculation formula of Q-ABC size

VII. A Clustering Method of Strategies by Code Metrics and Score

In previous researches, scatter plot SMP and recursive index RCM have been proposed, focusing on the correlation between scores and code metrics. In this way, strange code detection is realized by correlation with a relative index to the program of strategy [6]. On the other hand, in this research, students are considered as a group by the feature of strategy. This is intended to give appropriate advice to each group. Therefore, the strategies to be taught include those that are not strange code.

We discuss clustering method by score and Q-ABC size. First, standardize the scores and Q-ABC size of each strategy as pre-processing. This time, we use the deviation value with the target contest to be analyzed as the population. Second, we divide each strategy into clusters. In this paper, we apply the K-means method [7]. The K-means method is a clustering method that tries to minimize the mean square distance between points in the same cluster. Generally used widely. In particular, it is also used to analyze the behavior of programming learners. Lahtinen is trying to identify students with different skill levels as a group and identify student groups with skills problems [8]. Norwawi et al. try to recognize patterns in programming learning styles using classification algorithms [9].

```
int myhd[HNUM];
int num[13] = {0};
int sut[4] = {0};
int suteta_num[13] = {0};
int suteta_sut[4] = {0};
int k, k2,t;
for ( k = 0; k < HNUM; k++ ) {
   myhd[k] = hd[k];
}
for ( k = 0; k < HNUM; k++ ) {
  t = hd[k] % 13; num[t]++;
}
for ( k = 0; k < HNUM; k++ ) {
  t = hd[k] / 13; sut[t]++;
}
for ( k = 0; k < us; k++ ) {
  t = ud[k] % 13; suteta_num[t]++;
}
for ( k = 0; k < us; k++ ) {
   t = ud[k] / 13; suteta_sut[t]++;
}
int c2 = 0, c3 = 0;
for ( k = 0; k < 13; k++ ) {
  if ( num[k] == 3 ) { c3++; }
  if ( num[k] == 2 ) { c2++; }
}
int len = 0;
for ( k = 0; k < 13; k++ ) {
  if ( num[k] > 0 ) {
    len++;
    if ( len == 5 ) { break; }
  } else { len = 0; }
}
if ( len == 4 && num[0] == 1 ) {
  return -1;
}
if ( len == 5 ) {
  for ( k = 0; k < 4; k++ ) {
    if ( sut[k] == 5 ) {
       return -1;
    }
  }
}

// (After here, there are 52 lines)
```

Fig 4. A Snippet of Strategy with Highest Q-ABC Size in 2018

VIII. Clustering Strategies in WinT Practice

We apply a clustering method proposed in this paper to WinT's final league strategies in 2018 practice. The strategy at the final league is one for one student. Therefore, the students are divided into groups using the final league results. This time, we set the number of clusters to 4 and the tolerance in convergence judgment to be 0.0001. The initial value is random. The results are shown in Fig 5. Circles are Group 1, squares are Group 2, upper triangles are Group 3, and lower

Authorized licensed use limited to: Norges Teknisk-Naturvitenskapelige Universitet. Downloaded on March 20,2022 at 16:16:06 UTC from IEEE Xplore. Restrictions apply.

triangles are Group 4. These labeling were performed according to the following tendency.

- Group 1: Score is high
- Group 2: Average score and low Q-ABC size
- Group 3: Average score and high Q-ABC size
- Group 4: Score is low

As a result of clustering, 44 students were divided as shown in Table 4 as a result of the final league strategy. The score in each group and the average of Q-ABC size are also shown simultaneously. Group 1 has the highest average score. On the other hand, Group 4 has the lowest average score. Q-ABC size is extremely large in Group 3.

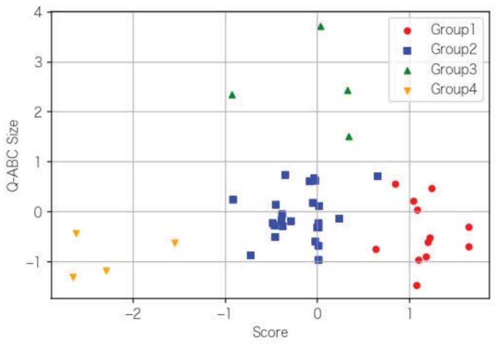

Fig 5. Result of Clustering by the K-means method

TABLE IV. SUMMARY OF GROUPS BY CLUSTERING

	Number of Students	Average of Score	Average of Q-ABC
Group 1	12	97.8	7.9
Group 2	24	70.0	8.9
Group 3	4	75.2	22.3
Group 4	4	14.1	5.2

IX. TENDENCY AND FEATURE OF STUDENTS IN EACH GROUP

First, we analyze the number of submissions in each group. The statistics for the number of submissions are shown in Table 5. As for the median, Group 1 is the largest. On the other hand, the Group 4 is the smallest. As for the minimum value, all groups are close to each other. On the other hand, the maximum value is extremely large in Group 1 compared to the other groups.

Second, we analyze the number of functions and the number of states. The statistics of the number of functions in each group are shown in Table 6. The median is around 6.0 without Group 1. The number of states is the total value of the number of assignments to variables, function calls, and conditional statements. The statistics of the number of states are shown in Table 7. Groups 1 and 3 have relatively large values. On the other hand, the two groups have smaller values compared to them.

TABLE V. STATISTICAL VALUE OF NUMBER OF SUBMISSIONS

	Median	SD	Min	Max
Group 1	39.5	30.4	9	116
Group 2	13.5	12.4	6	57
Group 3	16.0	9.0	7	27
Group 4	9.0	3.0	7	15

TABLE VI. STATISTICAL VALUE OF NUMBER OF FUNCTIONS

	Median	SD	Min	Max
Group 1	13.0	11.78	6	39
Group 2	6.5	4.3	3	18
Group 3	6.0	7.7	2	22
Group 4	5.5	7.9	2	22

TABLE VII. STATISTICAL VALUE OF NUMBER OF STATES

	Median	SD	Min	Max
Group 1	86.0	191.0	44	606
Group 2	71.5	55.5	31	217
Group 3	158.0	137.8	57	428
Group 4	32.5	65.3	9	172

X. DISCUSSION ON THE TENDENCY AND FEATURE OF STRATEGY

We consider the relationship between tendency and feature of each group and strategy. First, we consider the number of submissions. Basically, the number of submissions tends to be higher in the group with higher grades. In particular, the difference between Group 1 and Group 4 is remarkable. From this, it is likely that the Group 4 will need a mechanism or advice to increase the number of submissions. There is also a possibility that intensive instruction by teachers and teaching assistants (TAs) will be required.

Second, we considered the number of functions. From the results in Groups 2, 3 and 4, no extreme difference was found in the number of functions. From this, it is likely that it is difficult to give advice from the relationship between score and Q-ABC size regarding function design. Therefore, analysis focusing on different indicators is required.

Finally, we considered the number of states. Group 1 has a large number of states but a large number of functions. Therefore, Q-ABC becomes smaller. When the code was actually visually observed, it was a tendency that appropriate function division was performed. Group 3 has a particularly large number of states. Visual inspection of these codes revealed that the same process was repeatedly described, the description of a specific function was too long, and the for statement was not used in scanning of an array. Fig. 6 is a snippet of an example in which the for sentence is not used in scanning the array. The same process like these continues for four times. These students need to be aware of the

Authorized licensed use limited to: Norges Teknisk-Naturvitenskapelige Universitet. Downloaded on March 20,2022 at 16:16:06 UTC from IEEE Xplore. Restrictions apply.

improvement in code quality. Also, as a prelude to this, it may be necessary to teach the basic syntax of C language.

```
if ( hand == 0 ) {
    shorthand[0] = 1;
    shorthand[1] = 2;
    shorthand[2] = 2;
    shorthand[3] = 5 - s_turn(myhd, buff);
    shorthand[4] = 5 - f_turn(myhd);
    shorthand[5] = 3;
    shorthand[6] = 3;
    shorthand[7] = 5 - sf_turn(myhd, buf);
    shorthand[8] = 5 - rsf_turn(myhd, buff);
} else if ( hand == 1 ) {
    shorthand[1] = 1;
    shorthand[2] = 1;
    shorthand[3] = 5 - s_turn(myhd, buff);
    shorthand[4] = 5 - f_turn(myhd);
    shorthand[5] = 2;
    shorthand[6] = 2;
    shorthand[7] = 5 - sf_turn(myhd, buf);
    shorthand[8] = 5 - rsf_turn(myhd, buff);
} else if ( hand == 2 ) {
  ...
```

Fig 6. An Example Code in Group 3

XI. Discussion of Advice based on Tendency of Strategies

We discuss the feedback strategy for each group. In actual application, we will consider how to use past practice data as a population. We use past data to predict which group the strategy submitted during the preliminary league belongs to.

We describe a methodology for converting the classification results into clusters into feedback. Keuning et al. analyzed the trends in code quality problems that novice tends to commit and whether issues are recognized by novice and corrections are made. Analysis of more than 450,000 source codes submitted by novice found that most grammatical problems were corrected, but there was little correction for factors that reduce cohesion (God class, Loose coupling, Too many methods etc.) [10].

In fact, even in WinT, there are many factors that reduce the cohesiveness of strategies with too high Q-ABC size. In particular, functions that have too much responsibility, such as the so-called God class, are frequently observed. These factors are generally recognized as Code smell [11]. Our feedback strategy is based on improving the cohesiveness for strategies with a particularly high Q-ABC size and providing algorithm guidance for strategies with low scores. For guidance to improve cohesiveness, aim to eliminate Code smell. For example, make them aware of the following removal of smell. : Duplicate code, too long methods, God objects, duplicate methods, etc.

Based on this methodology, in this paper, we discuss the strategy in the case where the final league of 2018 is a population. For Group 1 is more aware of refactoring than score improvement. It is also conceivable to advise software testing and version control methods.

For Group 2, aim for transition to Group 1. In particular, we make them aware of the improvement in their scores. We advise on effective algorithms such as efficiency improvement and automation of parameter adjustment, random number simulation of a deck, and look ahead by recursion. In addition, in the function expansion stage, we make students aware that they do not embed factors that correspond to Code smell. In particular, teachers need to be careful because adding code to improve scoring increases the possibility of generating methods that are too long.

For Group 3, aim for transition to Group 1 or Group 2. We make them aware of the improvement in code quality. For that purpose, we mainly advise on the code arrangement by auxiliary functions. In addition, they may not be good at learning basic items of C language such as array scanning and usage of functions. Therefore, we have them reconfirm the grammar and basic syntax of C language. In addition, this group is presumed to be a group where Code smell has already occurred. For this reason, teachers need guidance to solve these problems.

For Group 4, aim for transition to Group 2. They are likely to need advice on how to think of strategy ideas and how to use auxiliary functions. This group may require intensive instruction by teachers and TAs.

In addition, it is also conceivable to automatically give the above-mentioned advice from the support system. By automating advice, the burden on teachers and TAs can be reduced. In this way, the teacher can give more solid advice.

XII. Conclusion

We have proposed programming exercises on card game strategies. We provide students with an execution environment and operate the contest management server WinT. We set the duration of the preliminary league and accept submissions of the strategy code created by the students any number of times. The support system publishes scores and ranks, promotes improvement by re-examining strategies, and promotes continuous modification of code.

In this study, we group strategies in terms of code metrics and execution performance and analyze the feature of the code by each group. In addition, we analyzed the group feature of the code. As a result, grouping of strategies by score and Q-ABC size is likely to be applicable to advice personalization. In addition, applying this method mechanically and giving advice automatically from the support system may reduce the burden on teachers and TAs.

In this paper, we used the k-means method for the clustering method. This method has the disadvantage that it does not allow membership in multiple clusters and cannot flexibly express individual feature. Therefore, we are also focusing on the Fuzzy c-means method [12].

In addition, we need to analyze the relationship between regulation and student progress, which changes from year to year. In the future, we will consider details of clustering techniques and advice. These may be achieved by detecting Code smell by focusing on various code metrics other than Q-ABC size. Through these studies, we aim for more effective advice.

Authorized licensed use limited to: Norges Teknisk-Naturvitenskapelige Universitet. Downloaded on March 20,2022 at 16:16:06 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] F. Gemba, , N. Hanakawa, and H. Tominaga, “Educational Approach and Practices for an Applied C Programming Exercise with Poker Card Game Strategy and a Contest Style”, Proceedings of ICIA 2016, pp.97-104, 2016.

[2] T. Shimizu, N. Hanakawa, and H. Tominaga, “An Applied C Programming Exercise with Card Game Strategy and Analysis of Score Tendency of Hand Distribution in the Final League,” 2018 IEEE 7th Global Conference on Consumer Electronics (GCCE), pp. 634–635, 2018.

[3] N. Hanakawa, and H. Tominaga, “Code Metrics Analysis of Board Game Strategy Programs in the Final League Tournament for a Java Programming Exercise”, 2015 Shikoku-Section Joint Convention Record of the Institutes of Electrical and Related Engineers (SJCIEE2015), pp. 245, 2015.

[4] Fitzpatrick, J. “Applying the ABC metric to C, C++, and Java,” C++ Report, 1997.

[5] “Home - RuboCop: The Ruby Linter that Serves and Protects,” RuboCop. [Online]. Available: http://www.rubocop.org/. [Accessed: 19-Oct-2019].

[6] F. Gemba, and H. Tominaga, “ Programming Exercise for Problem Solving with Card-Game Strategy -Functions of Submission Situation and Code Comparison in Contest Management Server-,” IPSJ SIG Technical Reports, Vol. 2015-CE-128, No. 9, pp. 1-6, 2015.

[7] J. A. Hartigan and M. A. Wong, “Algorithm AS 136: A K-Means Clustering Algorithm,” Applied Statistics, vol. 28, no. 1, p. 100-108, 1979.

[8] E. Lahtinen, “A Categorization of Novice Programmers: A Cluster Analysis Study,” PPIG, Vol. 16, pp.32-41, 2007.

[9] N. M. Norwawi, S. F. Abdusalam, C. F. Hibadullah, and B. M. Shuaibu, “Classification of students performance in computer programming course according to learning style,” 2009 2nd Conference on Data Mining and Optimization, pp.37-41, 2009.

[10] H. Keuning, B. Heeren, and J. Jeuring, “Code Quality Issues in Student Programs,” Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE 17, pp.110-115, 2017.

[11] M. Fowler and K. Beck, Refactoring improving the design of existing code. Boston: Addison-Wesley, 2013.

[12] J. C. Bezdek, “Pattern recognition with fuzzy objective function algorithms”, Springer Science & Business Media, 2013.

Authorized licensed use limited to: Norges Teknisk-Naturvitenskapelige Universitet. Downloaded on March 20,2022 at 16:16:06 UTC from IEEE Xplore. Restrictions apply.