**Table 2.** Clustering in incorrect solution codes of Problem 1

| Cluster | Detected Errors | Descriptions |
|---|---|---|
| 1 | [ 20 ], [ 21 ], [ 10 ], [ i ], +2 ] | Wrong array size, array reference |
| 4 | printf ( " % d \n " , 1 ) ; printf ( " % s \n " , str ) ; printf ( " % d \n " , num ) ; % d " , i ) ; printf ( " % c | Error around printf statement |
| 8 | for ( i = 0 ; i < 20 ; i ++ ) in [ i ] = ' \0 ' ; scanf for ( i = 0 ; i < 22 ; ++ i ) str [ i ] = ' \0 ' ; fgets | Error related with assignment process |
| 12 | = 21 ; = 18 ; = 20 ; j = 0 ; = - 1 | Assignment operation |

**Table 3.** Clustering in accepted solution codes of Problem 1

| Cluster | Changes in accepted codes | Descriptions |
|---|---|---|
| 0 | } printf ( " \n " ) ; return, } printf ( " \n " ) ; }, ; printf ( " \n " ) ; }, } printf ( " \n " ) ; return 0 ; }, } fprintf ( fout , " \n " ) ; return, } printf ( " % s " , " \n " ) ; return | Add or changes in printf function or output |
| 5 | [ 100 ], [ 21 ], [ 22 ], [ j ], [ k ], + 1 ] | Changes around array sizes and value |
| 14 | s « endl ; t « endl ; str « endl ; )) « endl ; ] « endl ; ) « endl ; « str « endl; c « endl ; a « endl ;; return | Added new line for output |
| 17 | } cout « " \n " ; return ; cout « " \n " ; } ; cout « " \n " ; return } cout « " \n " ; } ; std :: cout « " \n " ; return | Added new line for output |

cluster 16 shows miscellaneous miscalculations in incorrect codes.

**Table 4.** Clustering in both incorrect and accepted solution codes

| Clust. | Errors in incorrect codes and Changes in accepted codes | Descriptions |
|---|---|---|
| 3 | WA: } return AC: } std :: cout « std :: endl ; return WA: ; } AC: ; std :: cout « std :: endl ; } WA: } return AC: } std :: cout « "" « std :: endl ; return WA: } } AC: } std :: cout « std :: endl ; } WA: ; return AC: ; std :: cout « std :: endl ; return | Users made mistake to add new line feed as output. |
| 13 | WA: [ 20 ], AC: [ 100 ] WA: [ 20 ], AC: [ 21 ] WA: [ 21 ], AC: [ 22 ] WA: [ i ], AC: [ j ] WA: [ i ], AC: [ k ] WA: + 2 ], AC: + 1 ] | Wrong array size, array reference, etc. |
| 16 | WA: i > 10, AC: i - 10 WA: i > 0, AC: i ≥ 0 WA: f > 1, AC: f ≥ 1 WA: i > 10, AC: i ∧ 10 WA: length + 1, AC: length - 1 WA: 2 > I, AC: 2 < i WA: i > 10, AC: i ≠ 10 | Miscalculation of constants and variables |
| 27 | WA: = 21 ; AC: = 20 ; WA: = 18 ; AC: = 19 ; WA: = 20 ; AC: = 19 ; WA: - 2 ; AC: - 1 ; WA: - 3 ; AC: - 2 ; WA: - 1 ; AC: - 2 ; WA: + 19 ; AC: + 41 ; WA: + 40 ; AC: + 41 ; | Incorrect number value |

**Problem 2:**
**Print a Rectangle:** Draw a rectangle which has a height of $H$ cm and a width of $W$ cm. Draw a 1-cm square by single '#'.
**Input:** The input consists of multiple datasets. Each dataset consists of two integers $H$ and $W$ separated by a single space. The input ends with two 0 (when both $H$ and $W$ are zero).
**Output:** For each dataset, print the rectangle made of $H \times W$ '#'. Print a blank line after each dataset.
**Solution Codes:** A total of 1173 paired solution codes have been used.

We used both incorrect and accepted solution codes of Problem 2 for clustering, the Elbow method shows the optimal number of clusters is 2 as shown in Figure 8. The mean value of Cluster 3 is much higher than other clusters.

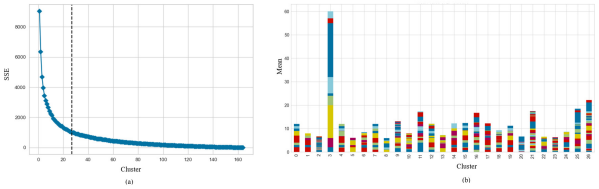

**Figure 8.** Incorrect and Accepted codes of Problem 2 (a) Optimal number of clusters using Elbow method (b) Mean value of each cluster

Clustering operation has been performed using both incorrect and accepted solution codes of Problem 2. Some cluster data is presented in Table 5. After analyzing the cluster data, we found that many users made common mistakes in the solution codes.

## 5 Conclusion

In this paper, we have proposed a method for the categorization of frequent errors in solution codes. We used

**Table 5.** Clustering in both incorrect and accepted solution codes of Problem 2

| Cluster | Errors in incorrect codes and Changes in accepted codes | Descriptions |
|---|---|---|
| 0 | WA: } } AC: } printf ( " \n " ) ; } WA: } } AC: } printf ( " % c " , ' \n ' ) ; } | Missing to print new lines in incorrect codes that are modified into accepted codes |
| 4 | WA: } } AC: } putchar ( ' \n ' ) ; } WA: } scanf AC: } putchar ( ' \n ' ) ; scanf WA: } return AC: } putchar ( ' \n ' ) ; return WA: } } AC: } cout « ' n ' ; } | Missing to print new lines in incorrect codes that are changed into accepted codes |
| 16 | WA: { for AC: { if ( a == 0 && b == 0 ) break ; for WA: { for AC: { if ( H == 0 && W == 0 ) break ; for WA: { for AC: { if ( H == 0 && W == 0 ) break ; for WA: { for AC: { if ( x == 0 \|\| y == 0 ) break ; for WA: { for AC: { if ( H == 0 & W == 0 ) break ; for WA: { REP AC: { if ( H == 0 && W == 0 ) break ; REP WA: { drawRect AC: { if ( H == 0 && W == 0 ) break ; drawRect | "Break" statement is missing in incorrect codes that are corrected into accepted codes. |
| 20 | WA: h ≤ H AC: h < H WA: l ≤ k AC: l < k WA: count ≤ z AC: count < z WA: d , b AC: d < b WA: x > w AC: x < w WA: i = H AC: i < H | The comparison operation was incorrect in the wrong codes that changed to accepted codes. |
| 10 | WA: } cout « endl ; return AC: } return<br>WA: ; cout « endl ; } AC: ; }<br>WA: } cout « endl ; for AC: } for<br>WA: } cout « endl ; if AC: } if<br>WA: { cout « endl ; return AC: { return<br>WA: } cout « endl ; } AC: } } | Wrong output line ("endl") was added to incorrect codes that deleted in accepted codes. |

paired source codes (incorrect → accepted) for our experiments. We have analyzed the reasons behind the errors, we find the most common errors in the solution codes made by the users. We also analyzed the correction process in the accepted solution codes. Experimental results show that the model can detect common errors made by users and cluster errors on the basis of similarity. The outcome of this research paper can be useful for students as well as teachers and instructors. Teachers can prepare accurate lecture plans to help students, so that they can solve common mistakes in solution codes. In future work, automatic correction of incorrect codes can be interesting.

## Acknowledgement

This research work was supported by the JSPS Kakenhi grant number 19K12252.

## References

[1] Andrew Ettles, Andrew Luxton-Reilly, and Paul Denny. 2018. Common logic errors made by novice programmers. In Proceedings of the *20th Australasian Computing Education Conference (ACE '18)* . Association for Computing Machinery, New York, NY, USA, 83–89.

[2] Brett A. Becker. 2016. An Effective Approach to Enhancing Compiler Error Messages. In Proceedings of the *47th ACM Technical Symposium on Computer Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 126–131.

[3] Paul Denny, Andrew Luxton-Reilly, and Ewan Tempero. 2012. All Syntax Errors Are Not Equal. In Proceedings of the *17th ACM Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '12)*. ACM, New York, NY, USA, 75–80.

[4] Raymond S. Pettit, John Homer, and Roger Gee. 2017. Do Enhanced Compiler Error Messages Help Students? Results Inconclusive. In Proceedings of the *2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 465–470.

[5] James Prather, Raymond Pettit, Kayla Holcomb McMurry, Alani Peters, John Homer, Nevan Simone, and Maxine Cohen. 2017. On Novices' Interaction with Compiler Error Messages: A Human Factors Approach. In Proceedings of the *2017 ACM Conference on International Computing Education Research (ICER '17)*. ACM,New York, NY, USA, 74–82.

[6] Md. Mostafizer Rahman, Yutaka Watanobe, and Keita Nakamura. 2020. A Neural Network Based Intelligent Support Model for Program Code Completion. *Scientific Programming*, vol. 2020, Article ID 7426461, 18 pages.

[7] Md. Mostafizer Rahman, Yutaka Watanobe, and Keita Nakamura. 2020. Source Code Assessment and Classification Based on Estimated Error Probability Using Attentive LSTM Language Model and Its Application in Programming Education. *Applied Sciences*, vol. 10, no. 8: 2973.

[8] Md. Mostafizer Rahman, Yutaka Watanobe, and Keita Nakamura. 2021. A Bidirectional LSTM Language Model for Code Evaluation and Repair. *Symmetry*, 13(2):247.

[9] Marzieh Ahmadzadeh, Dave Elliman, and Colin Higgins. 2005. An Analysis of Patterns of Debugging Among Novice Computer Science Students. SIGCSE Bull. 37, 3 (June 2005), 84–88.

[10] Sue C Fitzgerald, Gary Lewandowski, Renée Mccauley, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, June 2008, 93–16.

[11] Michael McCracken, Vicki Almstrum, Danny Diaz, Mark Guzdial, Dianne Hagan, Yifat Ben-David Kolikant, Cary Laxer, Lynda Thomas, Ian Utting, and Tadeusz Wilusz. 2001. A Multi-national, Multi-institutional Study of Assessment of Programming Skills of First-year CS Students. SIGCSE Bull. 33, 4 (Dec. 2001), 125–180.

[12] James C. Spohrer and Elliot Soloway. 1986. Novice Mistakes: Are the Folk Wisdoms Correct? Commun. ACM 29, 7 (July 1986), 624–632.

[13] Yutaka Watanobe. Aizu online judge. https://onlinejudge.u-aizu.ac.jp. (2018)

[14] Davin McCall and Michael Kölling. 2014. Meaningful categorisation of novice programmer errors. In the Proceedings of the *2014 IEEE Frontiers in Education Conference (FIE)*, Madrid, 2014, pp. 1-8.

[15] Maria Hristova, Ananya Misra, Megan Rutter, and Rebecca Mercuri. 2003. Identifying and Correcting Java Programming Errors for Introductory Computer Science Students. SIGCSE Bull. 35, 1 (Jan. 2003), 153–156.

[16] Amjad Altadmri and Neil C.C. Brown. 2015. 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. In Proceedings of the *46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. Association for Computing Machinery, New York, NY, USA, 522–527.

[17] Md. Mostafizer Rahman and Yutaka Watanobe. 2019. An efficient approach for selecting initial centroid and outlier detection of data clustering. In Advancing Technology Industrialization Through Intelligent Software Methodologies, Tools and Techniques; IOS Press: Amsterdam, The Netherlands, 2019; vol. 318, 616–628.