Integrating Parsons problems in a CS1 programming course autograder

Corentin Lengelé

UCLouvain University
Place Sainte corentin.lengele@student.uclouvain.be

Abstract. Parsons problems are a type of exercises in computer science education where students have to assemble a correct program from a set of scrambled or disordered program elements, by arranging the relevant program elements into the correct order. This work discusses how to integrate such Parsons problems into an existing autograder platform, and to observe the effectiveness of first-year bachelor students receiving such problems as compared to how they solve traditional coding tasks.

Introduction This poster will present our ongoing master thesis on integrating interactive learning methods for first-year bachelor programming courses through the integration of Parsons problems into an educational programming platform. A Parsons problem is an interactive exercise that requires rearranging code blocks rather than writing code. [1] The main purpose of integrating such exercises into an exsting autograder is to give students access to a different learning method and to observe the effectiveness of it.

Approach We aim to implement a Parsons problems plugin for the INGInious [1] autograder platform [2], based on some existing interactive Parsons problem examples[2]. The plugin will make this new problem type available to students, tutors, and teaching staff on the platform, allowing them to use or define such problems in the context of an existing course. An alternative programming course utilising Parsons problems will be developed for students to explore. This will provide valuable insights into the efficacy of this learning approach for first-year bachelor students.

Parson problems A Parsons problem is a coding challenge where learners must rearrange predefined blocks in the correct order. It is an intermediate exercise between a worked example where a solution is given and a classic problem solving exercise where the student needs to write code from scratch. This approach has shown comparable learning outcomes to traditional coding tasks while requiring less time. [3] This type of exercise assesses the understanding of multiple

[1] https://inginious.org/
[2] https://js-parsons.github.io
https://runestone.academy/ns/books/published/overview/Assessments/parsons.html

skills such as knowledge of syntax, problem decomposition, algorithm development, and debugging [4]. It can be structured as a drag-and-drop activity, where learners select code blocks from a provided list and assemble a solution within a designated area, as shown in Fig. 1.

Numerous additional options can be incorporated into this type of problem. [4] For instance, the code's indentation could either be pre-set or left for the student to determine, adding an extra layer of complexity. Introducing distractor blocks is another possibility, which are blocks that shouldn't be included in the solution. Alternatively, a pair can be presented with either a valid code block or a distractor that closely resembles it, further enhancing the problem-solving aspect.

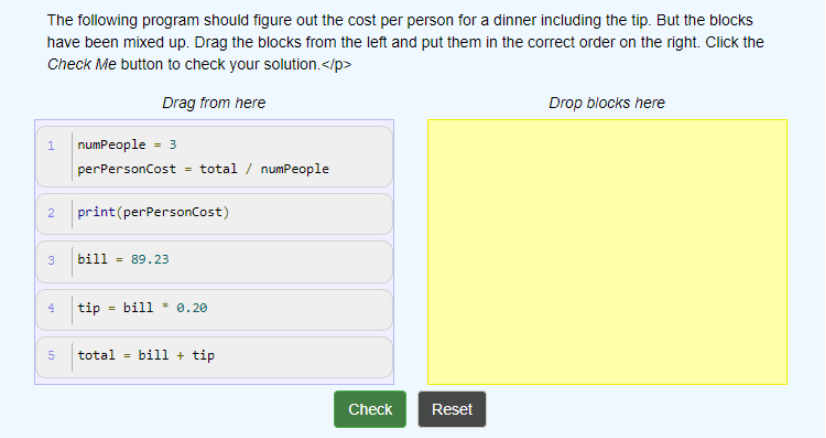

Fig. 1. Example of a Parsons problem structured as a drag-and-drop activity (from https://runestone.academy/ns/books/published/overview/Assessments/parsons.html).

INGInious INGInious [2, 5] is an intelligent autograder platform that allows secured and automated testing of code made by students. It is written in Python and uses Docker to run students' code inside a secured environment. It provides a backend which manages interaction with Docker to grade code, and a frontend which allows students to submit their code in an ease-to-use interface. The frontend also includes an administration interface for teachers to check the progress of their students and to modify exercices in a simple way. Various types of problems such as coding exercises, multiple-choice questions or mathemical problems are already integrated into the platform. Integrationg new problem types is q quite easy process that does not need modifications to the codebase.

References

1. P. Ihantola and V. Karavirta, "Two-dimensional parson's puzzles: The concept, tools, and first observations," *Journal of Information Technology Education. Inno-*

vations in Practice, vol. 10, p. 119, 2011.
2. G. Derval, A. Gego, P. Reinbold, B. Frantzen, and P. Van Roy, "Automatic grading of programming exercises in a mooc using the inginious platform," *European Stakeholder Summit on experiences and best practices in and around MOOCs (EMOOCS'15)*, pp. 86–91, 2015.
3. B. J. Ericson, L. E. Margulieux, and J. Rick, "Solving parsons problems versus fixing and writing code," in *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, ser. Koli Calling '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 20–29. [Online]. Available: https://doi.org/10.1145/3141880.3141895
4. B. J. Ericson, P. Denny, J. Prather, R. Duran, A. Hellas, J. Leinonen, C. S. Miller, B. B. Morrison, J. L. Pearce, and S. H. Rodger, "Parsons problems and beyond: Systematic literature review and empirical study designs," in *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education*, ser. ITiCSE-WGR '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 191–234. [Online]. Available: https://doi.org/10.1145/3571785.3574127
5. "Inginious grading platform for students," https://github.com/UCL-INGI/INGInious.