Assessing ChatGPT's Proficiency in CS1-Level Problem Solving

Mario Sánchez(✉) and Andrea Herrera

Universidad de los Andes, Bogotá, Colombia
{mar-san1,a-herrer}@uniandes.edu.co

Abstract. ChatGPT is an advanced large language model (LLM) capable of generating code to solve specific problems when presented with carefully designed prompts, among other capabilities. The existence of ChatGPT raises signifi-cant questions regarding teaching practices and evaluations within the dis-cipline. If ChatGPT can effectively solve exercises assigned to students, it prompts a reevaluation of the skills and knowledge that we teach and eval-uate. The objective of this paper is to assess the proficiency of ChatGPT in solving exercises commonly encountered in a CS1 course. This serves as an initial step in exploring the implications of ChatGPT for computer science education. By examining ChatGPT's performance and comparing it with real students, we aim to gain insights into its capabilities and limitations. Our evaluation encompasses a comprehensive examination of 125 problems specifically designed for CS1-level learners. The experiment revealed that ChatGPT successfully solved approximately 60% of the provided prob-lems. Subsequently, we conducted a detailed analysis of the characteristics of the problems that ChatGPT could not solve, aiming to gain a deeper understanding of the nuances that make them challenging for LLMs. This study contributes to the ongoing discourse surrounding the integration of AI-based tools, such as ChatGPT, in computer science education, and high-lights the need for a reevaluation of educational objectives and methods employed in traditional educational institutions.

Keywords: CS1 · Large language model · Computer science education · Generative AI

1 Introduction

In recent years, the field known as Artificial Intelligence in Education (AIED), which focuses on leveraging AI technologies for educational purposes, has experienced a surge of interest among researchers, practitioners, and educators. The ultimate goal of the field is to improve the effectiveness of educational practices by means of AI-powered tools [1]. Much like previous technological advancements introduced in the education sector, "learning how to embrace new technologies in teaching is not easy" [2] and presents both unique challenges and promising opportunities. Over time, curricula and teaching practices should adapt and

M. Tabares et al. (Eds.): CCC 2023, CCIS 1924, pp. 71–81, 2024.
https://doi.org/10.1007/978-3-031-47372-2_7

incorporate these new technologies to enable the development of even more advanced competences [2]. For example, AI-based tools are being used today in science curricula to improve the assessment - feedback loop that has typically been too long and not specific enough [3].

Among all of the available technologies that fall within the scope of AIED, Natu-ral Language Processing (NLP), Large Language Models (LLMs), and tools based on them such as OpenAI's ChatGPT [4,5] are creating the greater disruptions. It is no secret that students all over the world have been extensively using ChatGPT since it was released as a public (and free) service at the end of 2022, a reality that has ignited considerable controversy in the educational sphere, even prompting calls for prohibitions [6]. Similar to other advances from the past, ChatGPT makes it possible for a machine to perform activities that were previously restricted to humans, such as writ-ing a paragraph or essay about a given topic, with proper structure, internal coherence, and advanced use of the language to pass as written by humans. Given that these characteristics are typically central to the teaching and assessment processes [2], there exists a widespread apprehension that students may excessively depend on this tool, which markedly eases information acquisition, thereby engaging less with course materials, and failing to develop skills to investigate, synthetize, and critically ana-lyze information to come to their own conclusions and solutions [6,7]. Ultimately, overreliance on AI-based tools and models may led to increased laziness, less creativi-ty, less appreciation of human-generated content, and diminished communication abilities [4,6,7]. Rather than adopting reactionary measures, such as outright bans on these technologies, what is required is a strategy with educational systems and curricula to adopt these technologies and use them to enable students to understand and solve even more advanced problems than they do today [2,6].

Among ChatGPT's most touted capabilities is its ability to interact with code [4]. Specifically, it can generate code based on user prompts [8], identify inefficiencies in code to recommend improvements and optimize algorithms [9], and help developers to identify and resolve errors faster than with traditional debugging tools [10]. The abilities that humans require to perform this kind of activities are precisely a subset of the abilities typically targeted in a CS-1 course [11], i.e., an introductory course to programming. Consequently, the advent of ChatGPT has given rise to the same con-cerns regarding AIED in these courses, as we discussed earlier.

Given that the AIED field is relatively new, there are still large uncertainties and avenues for research. The open problems that we need to solve are varied and will probably require the collaborative efforts of several disciplines such as computer-science and psychology. In this paper, our intent is to contribute to the discourse by examining a very specific problem we have encountered firsthand: is it possible to write programming exercises for a CS-1 course that are solvable by students but are not easily solved by ChatGPT? Our motivation for this inquiry stems from the ob-servation that our students in a large CS-1 course have been resorting to ChatGPT for assistance, in contravention of course rules and arguably against their best interests. Since exercises in this course are mainly a formative tool to practice and strengthen abilities, delegating this work to ChatGPT is counterproductive.

The rest of this paper is organized as follows. Section 2 presents the context for our experiment, which is fully described Sect. 3. In this experiment we evaluated ChatGPT's capabilities with 125 exercises from our CS-1 course and compared its results with actual students' performance. Notably, all of our experiments were per-formed in Spanish, the language of instruction of the course. Next, Sect. 4 presents the analysis of the experiment's results and tries to understand what are the common characteristics among the 48 problems that ChatGPT failed to solve. Finally, Sect. 5 concludes the paper and discusses potential avenues for research.

2 Experimental Context: A CS-1 Course

This research was conducted in the context of a large CS1 course at the Universidad de los Andes in Colombia. This course teaches basic programming concepts using Python and serves approximately 1000 students per semester from different programs and schools (Engineering, Sciences, Economics, Design) in groups of at most 24 students. The course is taught in Spanish, and it is divided into four modules: (1) discovering the world of programming, (2) making decisions, (3) repeating actions and handling one-dimensional data structures, and (4) solving problems with arrays and libraries [12].

A key characteristic of this course is the usage of Senecode, an automated grading tool to support deliberate practice [13]. When using Senecode, a student gets the de-scription of a problem that he must solve by writing a Python function with a specif-ic signature (function name and parameters). After he submits a possible solution, the tool runs it using predefined inputs and compares the produced output with the known correct answers. For several problems, Senecode is also capable of generating synthetic inputs and outputs to use during this evaluation phase which concludes when the student receives feedback on his submission. Instead of rendering a binary verdict of'correct' or'incorrect', this platform aims to provide constructive feedback that could assist students in refining their solutions, thereby supporting their educational process.

There are currently 125 problems in the platform classified by course module, with different degrees of difficulty: problem authors assign an intended difficulty (a number from 0 to 50) but students may perceive difficulty in a different way. Figure 1 shows, for each problem, the assigned difficulty compared to the percentage of submissions that have been successful, and the percentage of students that have tried to solve a problem and have been able to do it after one or many attempts. The latter may be considered the real difficulties of the problems since they are grounded on student behavior. The figure shows that 1) the assigned and the real difficulty typically do not match; 2) that there are "hard" problems where most submissions are wrong; 3) and that for the majority of problems, most students eventually solve the problems that they attempt.

The final characteristic of the Senecode platform is its ability to reject submissions that employ language features that are above the course level at the

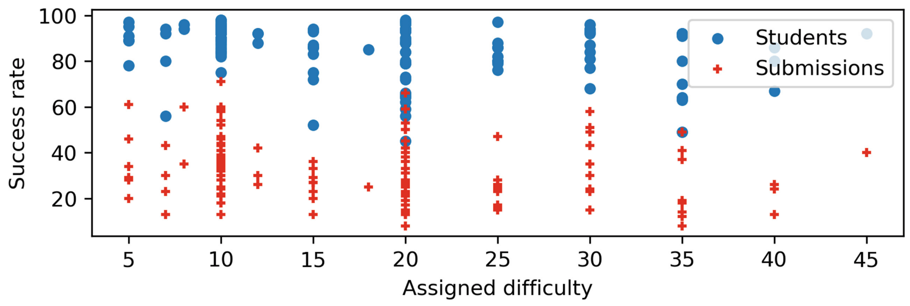

Fig. 1. Assigned difficulty vs. Student success rate and submission success rate, for each problem.

module. For example, problems intended to reinforce conditionals -in the second module of the course- are configured to disallow the usage of loops and lists. Another example is a module 1 problem where tree numbers have to be sorted without using conditional statements. These restrictions are automatically enforced and are clearly announced as part of problem statements. Finally, some constructs that are not studied on the course, such as lambdas and list comprehensions, are also disallowed.

3 Experimental Design and Results

It is not surprising that students are already using ChatGPT to help them when solving code problems. Regardless of the ethical considerations and the fact that it does not make sense to use external help to solve problems designed for training, this generates interesting questions for instructors and problem designers. For this paper, we have summarized these concerns into one research question:

How can we design problems so that
ChatGPT will not be able to naively solve them. (RQ.1)

We use the "naively" qualifier because we expect students to be using simple prompts based on the problem statements - even just copy-pasted text - instead of very elaborated reinterpretations. After all, if students can write precise prompts, they are probably capable of solving every CS1 problem and do not need ChatGPT help at all.

To answer this question, we planned an experiment based on asking ChatGPT to solve each problem in the Senecode platform and comparing its results with those of actual students. For this purpose, we created an automated pipeline that i) gets each problem statement from Senecode, ii) generates a complete prompt, iii) uses OpenAI's API to query GPT 3.5-turbo model in a new session per problem, iv) gets the response and extracts the meaningful answer (a Python function), v) and submits the solution to Senecode so that it can be automatically graded. After this pipeline was run for each problem, we first used Python scripts to analyze the results and compare them with actual students, and then performed a manual analysis - problem by problem - to try to understand the characteristics of those that remained unsolved by ChatGPT.

For illustration purposes, we now present an example of a prompt used for one of the simplest problems and the solution that ChatGPT offered for this problem. Since the original experiment used a prompt completely written in Spanish, ChatGPT provided a solution also written in Spanish. The following is a translation as close to the original as possible.

Problem Prompt. I need a Python function to help me solve the following problem.

"Create a function that receives the cost in pesos of a restaurant bill, and then calculates the associated VAT tax and the tip for the waiter. The VAT rate is 19%, and the tip in the restaurant is 10% of the bill value (without tax).

The function must return a string showing the VAT, tip and total as follows: "'X,Y,Z"', where **X** is the VAT, **Y** the tip and **Z** the total.

Do not forget to approximate your result to the nearest integer.

Note: The approximation must be done only on the return string (not before)"

The function to solve the problem must be in Python, must be named `calculate_vat_tip_total_bill` and must comply with the following restrictions:

The function must receive only the following parameter:

* `bill_cost` (parameter of type '`int`'): cost of the restaurant bill, without tax or tip.

The return of the function must be of type '`str`'. The function returns a string with the vat, tip and total bill, separated by commas.

In addition, the function must respect the following restrictions:

* It should not use the '`if`' primitive to solve this problem.
* It should not use the '`for`' primitive to solve this problem.
* It should not use the '`while`' primitive to solve this problem.

ChatGPT's Solution. The following is the solution that ChatGPT proposed for the above prompt.

```python
def calculate_ vat_ tip_total_bill bill_cost: int) -> str:
    vat = int(round(bill_cost * 0.19))
    tip = int(round(bill_cost * 0.1))
    total = bill_cost + vat + tip
    return "{},{},{}".format(vat, tip, total)
```

As shown in this example, problem statements frequently have some context and are not direct questions: they require some level of interpretation to get to the correct answer. Also, some of the problems are known problems that are typically found in any introduction to programming book, but most are somehow original. Every prompt used for the experiment was structured in the same way to be as clear as possible. Before the whole experiment was run, we tested several prompt structures, and we selected the one that got the best results. In ChatGPT's answer we see that it has understood the general request and has provided a function with the right signature. However, in this case it has provided a faulty solution for the problem: it does not consider that rounding must be done only in the return string (not before) and thus in some cases it provides a wrong answer.

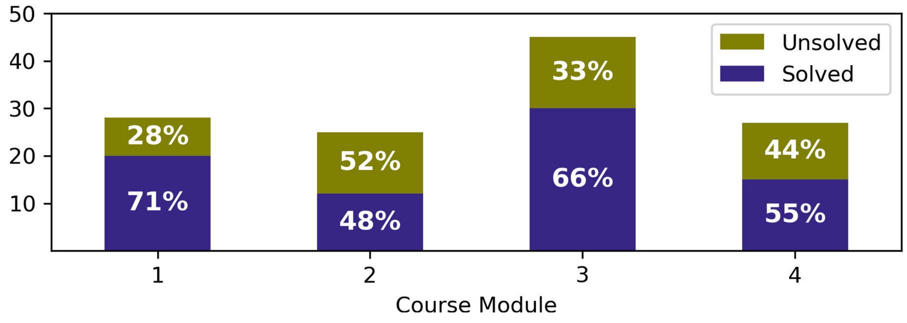

Fig. 2. Number and percentage of problems solved by ChatGPT per course module.

After running the whole experiment, we found that ChatGPT was able to solve only 62% of the problems, which came as a surprise since we were expecting a number close to 90%. Figure 2 shows the number of problems successfully solved by ChatGPT for each of the four modules in the course.

4 Analysis and Discussion

In order to try to answer our research question (RQ1: How can we design problems so that ChatGPT will not be able to naively solve them) we have to analyze the results obtained in the experiment, from a number of perspectives.

Figure 2 already showed that the course module is not a definitive factor. For example, we initially expected problems from module 2, which do not require loops and linear data structures, to be easier than problems in module 3 but that was not the case. ChatGPT was unable to solve 52% of problems in module 2 and 33% of problems in module 3. This suggest that the content of the problems, at least in the context of a CS-1 course, is not a factor that makes a problem easier of harder for ChatGPT. It is also worth noting that ChatGPT did not solve all the problems of each module: it should be possible, in principle, to write additional problems for each module that are "unsolvable".

Our second analysis considered the difficulty of the problems. Figure 3 shows the behavior of ChatGPT in each problem compared to their assigned difficulty, the student success rate, and the submission success rate: each dot in the left side of the diagram represents a problem that ChatGPT solved while dots in the right side are represent those that remained unsolved. What we can see from the first diagram is that ChatGPT success in each problem does not appear to be related to the assigned difficulty. The second diagram shows that ChatGPT was able to solve both problems that are hard and easy for students; moreover, all the problems that ChatGPT failed to solve also have a success rate among students that is less than 70%. Finally, the submission success rate appears to be related to ChatGPT effectiveness. This means that problems where students typically struggle and have to make more submissions to get to the correct answer, are also harder for ChatGPT.

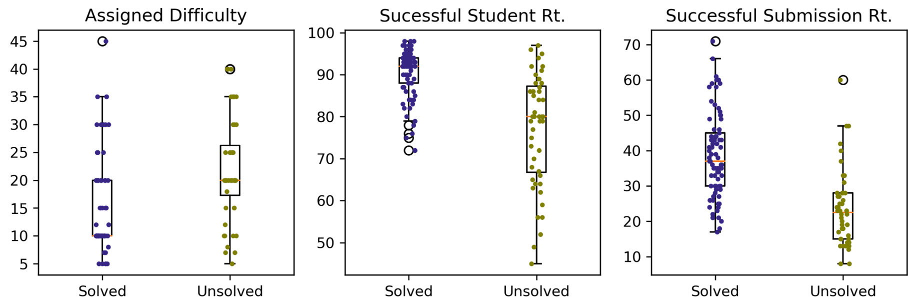

Fig. 3. ChatGPT results compared to a) assigned difficulty, b) student success rate, and c) submission success rate.

These results led us into a problem-by-problem analysis of the problem statements to understand what makes a problem hard to solve for ChatGPT (and for students!). The first analysis considered the length of the prompts, measured by the number of words. As shown in Sect. 3, the length of a prompt depends on the amount of information on the problem statement, in the function description, and in the restrictions. We found a correlation that was the contrary to the one we were expecting: problems with longer prompts are more likely to be solved than problems with shorter prompts. Our interpretation of this phenomenon considers two aspects. First, that giving ChatGPT more information about a problem steers him into the right direction. Secondly, that prompts for these problems are not long enough to make it loose attention and start forgetting the initial parts of each one. The lesson learned from this is that problem statements should be succinct, but not too much, in order to confuse ChatGPT (Fig. 4).

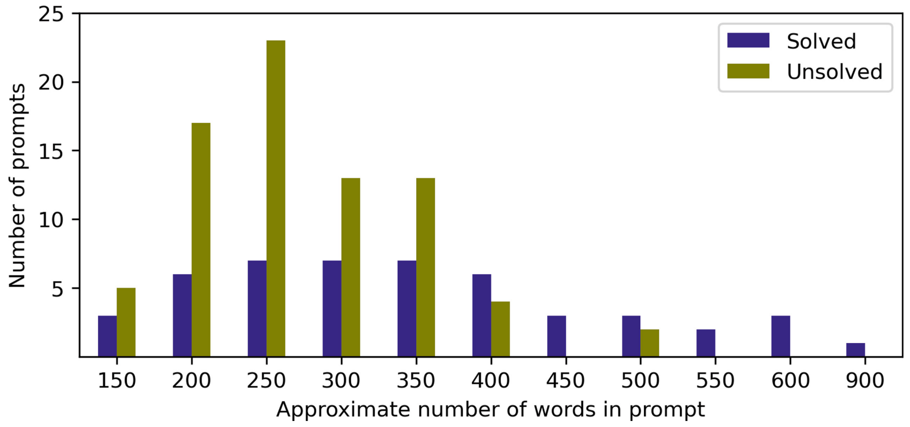

Fig. 4. ChatGPT results compared to the number of words in the prompts.

Next, we performed a qualitative analysis of each problem statement and classified them according to four characteristics.

1. Direct or Indirect Problem. An indirect problem is understood as one that gives a context or describes a scenario and then poses a problem to solve in that specific scenario. An example of a short indirect problem is the following: "The computing department tracks computer usage in one of the computer labs for its students. Basic data is collected on the date and start time of each session, how long the computer was used measured in minutes, and who was the student. Find if there were concurrent sessions by the same student.". An example of a direct problem is the following: "Write a function that receives a string as a parameter and removes all the vowels (lowercase or uppercase) in it".

2. Typical Programming Problem. Typical problems are those problems that are commonly studied in CS1 and are found in books, training materials, videos, etc., without major variations. The following is a typical problem: "Write a function that searches within a number (received by parameter) what is the largest digit appearing in it". On the contrary, the following is a non-typical problem "Create a function that calculates the body mass index BMI of a person with the formula $BMI = weight/height^2$, where the weight is in kilograms and the height is in meters. Note that the weight and height that your function receives will be given in pounds and inches respectively".

3. Problem with Format Requirements. While some problems ask for a single and simple return, such as in the BMI example, there are some problems that ask a specific format for the output data. For example, "The return of the function must be of type 'str'. The function returns a string indicating the person's age in years, months, and days as integers separated by single blank spaces".

4. Problem with Rounding Requirements. The fourth characteristic is related to the previous one and focuses on specific issues with rounding operations, including the number of decimal places in the answer and when rounding should be done. For example, "The function returns the angle (in degrees) between the hands of the clock according to the hour and minute given as parameter, which must have a single decimal digit". The problem presented in Sect. 3 also exhibits this characteristic.

After classifying the 124 problems in the experiment with respect to these four characteristics, we obtained the results shown in Tables 1 and 2. Table 1 shows how many problems in the data set had each characteristic, and how many of those problems remained unsolved for ChatGPT. These results show that only the third characteristic proved to be a consistent challenge for ChatGPT: in 51% of the problems with specific formatting requirements, ChatGPT failed to provide a correct answer. The indirect and rounding characteristics followed with 42% and 40% of unsolved attempts.

Table 1. ChatGPT results compared to single features.

Feature	**Indirect**	**Typical**	**Format**	**Rounding**
#Problems	79	66	41	20
Unsolved	42%	33%	51%	40%
Solved	58%	67%	49%	60%

Table 2 shows the results of the analysis by combining two of the four defined characteristics. These results show that indirect problems requiring formatting represent a challenge for ChatGPT, which fails in 53% of the cases. The problems requiring both formatting and rounding shows promising results because ChatGPT failed in 100% of these problems, but unfortunately only 3 problems in the problem set meet this combination of characteristics. We do not report any combination of more than two characteristics because the problem set did not have any problems with this combination of characteristics.

Table 2. ChatGPT results compared to combined features.

Feature	**Indirect + Format**	**Indirect + Rounding**	**Typical + Format**	**Typical + Format**	**Format + Rounding**
#Problems	30	6	19	9	3
Unsolved	53%	17%	47%	22%	40%
Solved	47%	83%	53%	78%	0%

Another result in Table 2 that is worth studying is the 47% of success found in the 'Typical + Format' column: a possible interpretation is that a strategy for making typical problems harder is to add formatting requirements (and get from 42% to 47%). This is an important consideration given that typical problems must be studied in CS1.

Finally, we also analyzed the impact on the success rate of ChatGPT of the restrictions in problems which disallow certain language constructs. Initially, close to 38% of ChatGPT's solutions were rejected because they did not respect or meet restrictions such as not using loops or lists in some problems. We lifted the restrictions and identified a minimal increase in the percentage of unsolved problems (from 38% to 35%). This showed that those restrictions are not the defining factor that makes a problem easy or difficult for ChatGPT to solve. This is interesting especially because empirical observations tell us that they make the problems considerably harder for those students that already know the language structures that are forbidden.

5 Conclusions

ChatGPT is already changing things in many contexts including software development and education, and it is here to stay. Educators and curricula need to adapt, learn how to use it to their benefit, and introduce changes in courses and evaluation methods to take advantage of this technology instead of seeing it just as a cutting-corner mechanism. Since it is impossible to control what students do and, it is probably impossible to discover when a solution was created by ChatGPT, we ought to learn how to design exercises that are not solvable - to a certain degree - with this kind of technologies.

With the experiment reported in this paper, we showed that in the context of a CS1 course, ChatGPT is not infallible. Even in the limited scope of the 124 problems in the Senecode platform, we found that especially problems with special formatting restrictions tend to be harder for ChatGPT. This information should be useful for us to write new problems in the future, but we believe that it is also applicable to any CS1 course.

There are several limitations in the experiments that we report in this paper that we expect to address in subsequent experiments. In particular, we would like to see the behavior of more advanced models, like GPT-4, which was not available via an API when this report was written. Another possible evaluation is to analyze the difference between several of OpenAI's models (ChatGPT, GPT-3, GPT-4) since there are slight differences in their training methods and tuning, and also with other companies' models such as Bard from Google.

Anther future experiment is to assess the impact of language. All of our prompts were prepared in Spanish, and the generated Python functions had function names and parameters in Spanish as well. Even though restricted experiments seem to imply that ChatGPT has a comparable behavior in Spanish and English, it would be interesting to have a better confirmation. Another avenue for research is the fact that ChatGPT does not always produce the same answer for the same prompt. In our experiment we only asked for one solution for each problem, but maybe asking for several could lead to different and possibly better answers. In fact, this is part of the strategy used by DeepMind's AlphaCode to succeed in programming competitions [14].

Finally, one of the goals of Senecode is to give students feedback for wrong submissions and guide them to the right answer without giving away the solution. Using ChatGPT we could assess the value of the feedback and improve it to better help the students.

References

1. Zhang, K., Begum, A.: AI technologies for education: recent research & future directions. Comput. Educ. Artif. Intell. **2**, 1–11 (2021)
2. Joyner, D.: ChatGPT in education: partner or pariah? XRDS **29**(3), 48–51 (2023)
3. Zhai, X.: ChatGPT for next generation science learning. XRDS **29**(3), 42–46 (2023)
4. Pratim Ray, P.: ChatGPT: a comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. Internet Things Cyber-Phys. Syst. **3**, 121–154 (2023)
5. Assaraf, N.: Online ChatGPT - Optimizing Language Models for Dialogue. Accessed 2 June 2023. https://online-chatgpt.com/
6. ChatGPT for good? On opportunities and challenges of large language models for education. Learn. Individ. Differ. **103**, 1–9 (2023)
7. Hang Choi, E., Lee, J.J., Ho, M., Kwok, J., Lok, K.: Chatting or cheating? The impacts of ChatGPT and other artificial intelligence language models on nurse education. Nurse Educ. Today **125**, 1–3 (2023)
8. Kashefi, A., Mukerji, T.: ChatGPT for Programming Numerical Methods, arXiv pre-print arXiv:2303.12093. Accessed 2 June 2023
9. Biswas S.: Role of ChatGPT in computer programming.: ChatGPT in computer programming. Mesopotamian J. Comput. Sci. **2023**, 8–16 (2023)
10. Surameery, N.M.S., Shakor, M.Y.: Use chat GPT to solve programming bugs. Int. J. Inf. Technol. Comput. Eng. (IJITC) **1**, 17–22 (2023)
11. Dale, N.: Content and emphasis in CS1: SIGCSE Bull. **37**(4), 69–73 (2005)
12. Buitrago, F., Sanchez, M., Pérez, V., Hernandez, C., Hernandez, M.: A systematic approach for curriculum redesign of introductory courses in engineering: a programming course case study. Kybernetes **1**(1), 1–10 (2022)
13. Sanchez, M., Salazar, P.: A feedback-oriented platform for deliberate programming practice. In: ITiCSE 2020: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, pp. 531–532 (2020)
14. Yujia, L., et al.: Competition-level code generation with alphacode. Science **378**(6624), 1092–1097 (2022)