Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona

Màster en Formació del Professorat d'Educació Secundària Obligatòria i Batxillerat, Formació Professional i Ensenyament d'Idiomes

Master's thesis

Analysis of the use of automatic judges in computer programming classes in vocational education

Elitza Nikolaeva Maneva

Supervised by Antoni Hernández-Fenández

June, 2022

Acknowledgements

I would like to thank all the teachers who were kind enough to fill out my survey, in spite of their full schedules, and also the students who completed a questionnaire, in spite of it not counting for a grade. I also owe thanks to the creators of automatic judges - Jordi Petit of *Jutge.org*, Marc Albareda of *JOEL*, and Marco Antonio Gómez Martín of *¡Acepta el Reto!* - for answering all of my questions about their creations. Many thanks to my colleagues Iriana Bonet and Santi Rivas for letting me learn from their experience and supporting my work on the master thesis. Thanks also to Alex Ríos and Fran Arias for letting me present *¡Acepta el Reto!* to their students, which was a great way to learn about the challenges of introducing automatic judges to vocational training students.

Most of all, I want to thank my thesis advisor Toni Hernández for all the ideas he has given me, and, especially, for the immediate and thorough responses to all my questions, even when they reached him right in the middle of a long weekend. I have learned a lot about educational research (and about how much I still don't know) from all the sources and the guidance he has given me.

Abstract

The goal of this master thesis is to analyse the use of automatic judges in the teaching of Computer Programming in Vocational Training programs, in a sample of secondary schools in Catalonia and Spain. We review the academic literature on automatic judges and describe their history and the different services that are currently available, with emphasis on the ones that are relevant to our geographic area. Through surveys to teachers and students from Vocational Training centres, we have collected information about the specific tools in use. We classify twelve of them according to features that we have found to be relevant to the teachers we surveyed.

Using the collected data, we study the reasons *in favour* and *against* using automatic judges, as well as the level of familiarity of active teachers with those tools. We identify some discrepancies between teacher's expectations and actual experiences of teachers who have adopted them. We use students' responses to derive statistics about their satisfaction, and to detect differences in the attitude of those students who use judges in class and those who don't with respect to programming.

In order to detect any effects on student learning, we use three "Bebras" challenges to evaluate the competence of algorithmic and computational thinking, which is central to the discipline of computer programming. No difference in the ability of students to complete these challenges has been observed, but a difference in how willing they are to work on them has been detected. The effect is contrary to the one originally hypothesised, which leads to some interesting questions for further study.

Resum

L'objectiu d'aquesta tesi és analitzar l'ús dels jutges automàtics en l'ensenyament de la Programació en els cicles de Formació Professional (FP), en una mostra d'instituts de secundària de Catalunya i d'Espanya. Revisem la literatura acadèmica sobre jutges automàtics i expliquem la seva història i els diferents serveis que hi ha actualment, fent èmfasi en els que són rellevants per a la nostra àrea geogràfica. Mitjançant enquestes a professors i a alumnes dels centres de FP, hem recollit informació sobre les eines concretes que s'utilitzen. Classifiquem dotze jutges automàtics segons les característiques que hem trobat rellevants per als professors enquestats.

Utilitzant les dades recollides, estudiem les raons *a favor* i *en contra* de l'ús de jutges automàtics, així com el nivell de familiaritat dels professors actius amb aquestes eines. Identifiquem algunes discrepàncies entre les expectatives del professorat i les experiències reals dels professors que les han adoptat. Utilitzem les respostes dels estudiants per obtenir estadístiques sobre la seva satisfacció, i per detectar diferències en l'actitud d'aquells alumnes que fan servir jutges a classe i els que no, respecte de la programació.

Per tal de detectar l'efecte en l'aprenentatge dels estudiants, utilitzem tres reptes de "Bebras" per avaluar la competència del pensament algorítmic i computacional, que és fonamental per a la disciplina de la programació d'ordinadors. No s'ha observat cap diferència en la capacitat dels estudiants per completar aquests reptes, però sí una diferència en la seva voluntat de treballar-hi. L'efecte és contrari al que es va plantejar originalment, la qual cosa porta a algunes preguntes interessants per a un posterior estudi.

Keywords

programming, vocational training, automatic judge, automatic assessment, computational thinking

Contents

List of Figures

List of Tables

1. Introduction

Programming is one of the main pillars of informatics education, and without any doubt the most important one in the specialties of software development. In this master thesis we will consider a tool for teaching this subject that has been adopted by many universities, and is gradually getting adopted by more and more vocational training programs. The automated program evaluators, sometimes also called "automated assessment tools", and also "automatic judges" are online services that evaluate the correctness of programs submitted by users [2, 3]. They usually have a large repository of tasks in the form of problem statements, together with several examples of input for the problem and the correct output that the user's program should compute. In addition, there are private test cases that are not known to the user, on which their program gets evaluated [4].

The large majority of automatic judges give as feedback only very limited information. We will describe what this information is in the next section. Even with this limitation,they are doubtlessly a handy tool for any learner, as they take only seconds to detect if a given program works correctly for all the available test cases (which can be hundreds). In the classroom automatic judges can be very useful, because they give immediate feedback that otherwise would take a long time to receive from the teacher, since each student may run into different issues. It is an interesting question then, why are they not more widely adopted, especially in classrooms with high ratios of students to instructors such as in the vocational training setting in Spain?

In this master thesis we have the following objectives:

- Review the literature on automatic judges and their use in introductory programming courses worldwide.
- Describe and categorise the different automatic judges that can be applied to the teaching of computer programming in vocational education in Catalonia and Spain and estimate their adoption, based on surveys and information from the communities of users.
- Identify the positive and negative aspects of using automatic judges in teaching, using surveys to vocational training teaching professionals. The questionnaire, included in Appendix A, was completed by 40 teachers.
- Analyse the effects of automatic judges on student learning and on the students' attitude towards programming, using surveys to current students. The questionnaire, included in Appendix B, was completed by 57 students from three different schools.

2. What are automatic judges?

An automatic judge is an online service that includes a large repository of programming tasks, also known as exercices, problems or challenges [2, 3]. These tasks are often organised in categories, according to the skills they require. The exercises in most automatic judges are not just straightforward statements of the requirements of the program, but instead contain an elaborate description of a situation in which the required computation might need to be performed. The first challenge for the user is to read the statement and filter out the technical requirements for what the program should do. In Figures 1 and Figure 2 we show two examples of problem statements from the websites *Codeforces* [5] and *Jutge.org* [6] of UPC.

In addition to the statement, the user is given a few examples of input and the corresponding output that is desired. These are usually very helpful for making sure the reader understands the statement correctly. There is a field where the user can submit their program. The program can be submitted in several different programming languages, and the user can choose which one. (This section is visible in Figure 1 in the middle of the right column.) On submission, the server compiles and runs the user's program on a large set of test cases, which are unknown to the user. As a result it outputs a message that can be one of several different kinds.

In order to describe the response of the judge we first need to explain some details about the programming process, and the kind of issues that can arise while programming. It's important to keep in mind that this is the workflow only for very small problems, such as the ones that are used typically to practise programming, and the ones proposed in programming competitions and in the repositories of all automatic judges. Programming for a larger project involves a much more complicated process that involves breaking up the project into small tasks. The design of tasks, scheduling and distributing the tasks among a team of developers is a completely different problem, which is also very important, but is not the subject of this thesis.

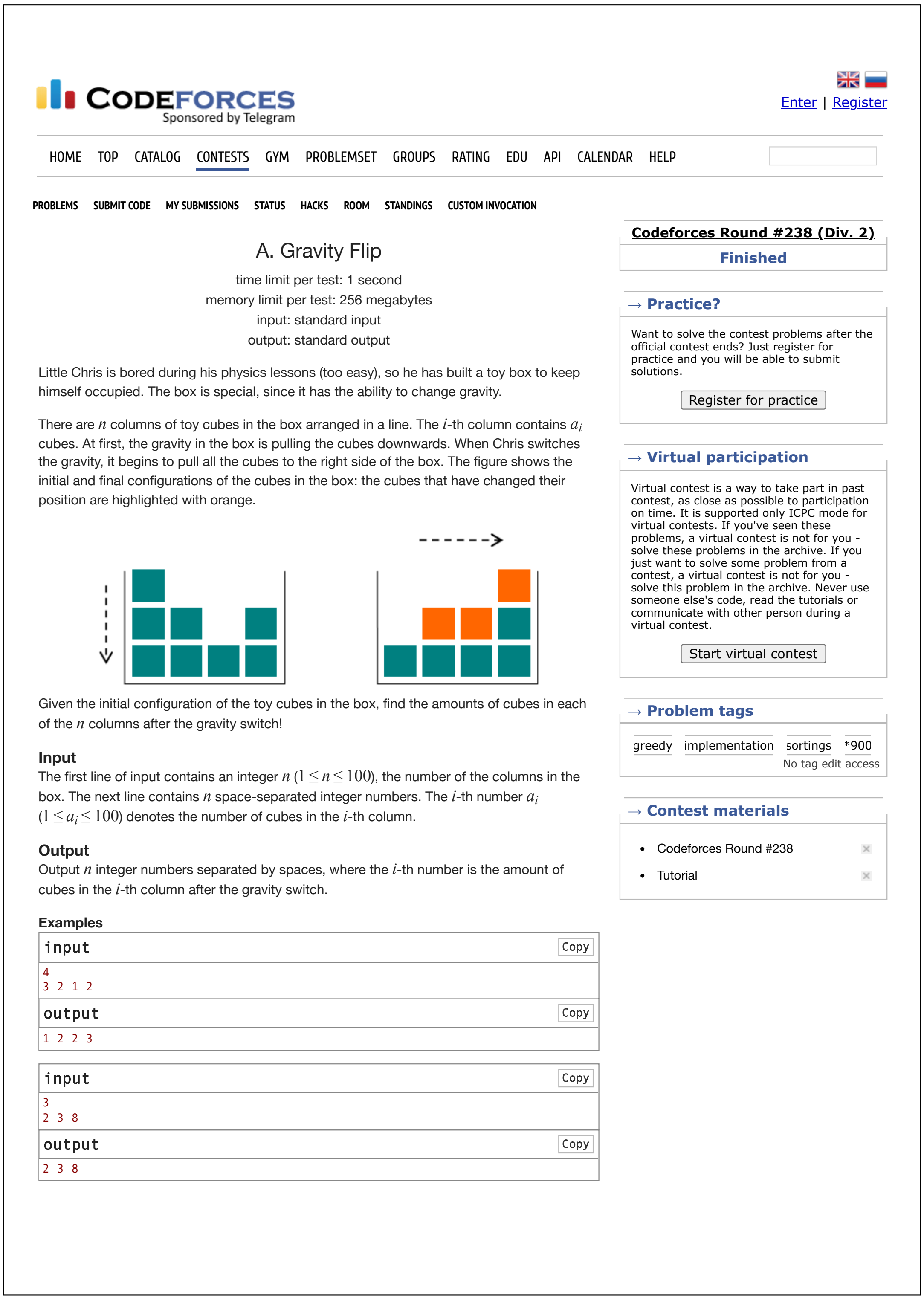
Figure 3 illustrates the workflow of solving a programming practice problem. We detail the sequence of steps below:

1. **Understanding the statement.** The first step in attacking a programming challenge or practice problem is reading and understanding the problem statement. We can verify that we have understood the problem correctly by checking that the input that is proposed as a test case, which is always sufficiently small to be processed by hand, corresponds to the output shown.

2. **Designing an algorithm.** After having understood the statement, a programmer has to design an algorithm that works for all inputs. This can be done with pen and paper in pseudocode (language similar to the programming languages, that doesn't have to be syntactically correct), or just schematically. It can also be done only mentally. It's good practice for first year students to write down their algorithms, but it is usually hard to convince them to do that consistently.

3. **Writing a program in a programming language.** Converting the algorithm into a program in a given programming language is an easy step, once we know what the algorithm is. It requires only knowing the syntax of the language.

4. **Compiling the program.** Some programming languages, such as Python, do not need to be compiled, and in that case we would skip directly to the next step. The compilation step is also often obscured for other languages by the development environment (IDE), because it is possible to do the compilation and the execution steps with just the press of a button. It is better practice for beginner programmers to split the two steps, because there is a big difference between compilation errors, which have to do with errors in the syntax, and execution errors, which are much more sophisticated.



Enter | Register

HOME TOP CATALOG CONTESTS GYM PROBLEMSET GROUPS RATING EDU API CALENDAR HELP

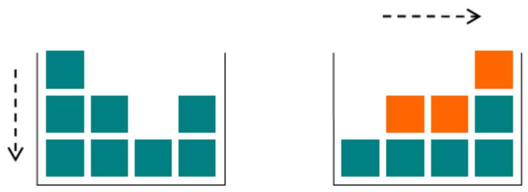
PROBLEMS SUBMIT CODE MY SUBMISSIONS STATUS HACKS ROOM STANDINGS CUSTOM INVOCATION

A. Gravity Flip

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Little Chris is bored during his physics lessons (too easy), so he has built a toy box to keep himself occupied. The box is special, since it has the ability to change gravity.

There are n columns of toy cubes in the box arranged in a line. The i-th column contains a_i cubes. At first, the gravity in the box is pulling the cubes downwards. When Chris switches the gravity, it begins to pull all the cubes to the right side of the box. The figure shows the initial and final configurations of the cubes in the box: the cubes that have changed their position are highlighted with orange.



Given the initial configuration of the toy cubes in the box, find the amounts of cubes in each of the n columns after the gravity switch!

Input
The first line of input contains an integer n $(1 \le n \le 100)$, the number of the columns in the box. The next line contains n space-separated integer numbers. The i-th number a_i $(1 \le a_i \le 100)$ denotes the number of cubes in the i-th column.

Output
Output n integer numbers separated by spaces, where the i-th number is the amount of cubes in the i-th column after the gravity switch.

Examples

input
```
4
3 2 1 2
```
output
```
1 2 2 3
```

input
```
3
2 3 8
```
output
```
2 3 8
```

Codeforces Round #238 (Div. 2)
Finished

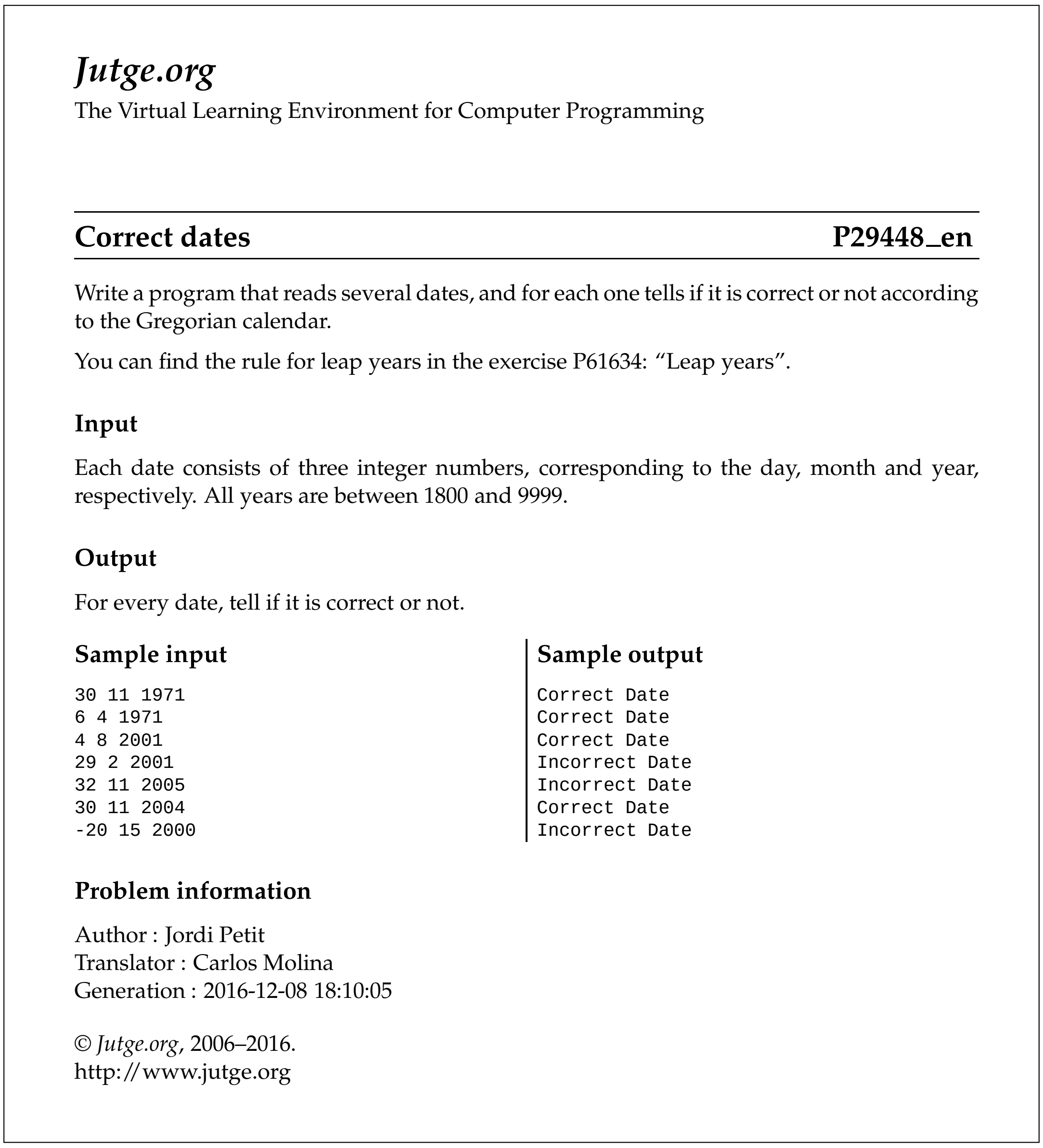
→ **Practice?**
Want to solve the contest problems after the official contest ends? Just register for practice and you will be able to submit solutions.
Register for practice

→ **Virtual participation**
Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you - solve these problems in the archive. If you just want to solve some problem from a contest, a virtual contest is not for you - solve this problem in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.
Start virtual contest

→ **Problem tags**
greedy implementation sortings *900
No tag edit access

→ **Contest materials**
- Codeforces Round #238
- Tutorial

Figure 1: Example of a problem statement in the repository of *Codeforces* [5].



Jutge.org
The Virtual Learning Environment for Computer Programming

Correct dates **P29448_en**

Write a program that reads several dates, and for each one tells if it is correct or not according to the Gregorian calendar.

You can find the rule for leap years in the exercise P61634: "Leap years".

Input

Each date consists of three integer numbers, corresponding to the day, month and year, respectively. All years are between 1800 and 9999.

Output

For every date, tell if it is correct or not.

Sample input

```
30 11 1971
6 4 1971
4 8 2001
29 2 2001
32 11 2005
30 11 2004
-20 15 2000
```

Sample output

```
Correct Date
Correct Date
Correct Date
Incorrect Date
Incorrect Date
Correct Date
Incorrect Date
```

Problem information

Author : Jordi Petit
Translator : Carlos Molina
Generation : 2016-12-08 18:10:05

© *Jutge.org*, 2006–2016.
http://www.jutge.org

Figure 2: Example of a problem statement in the repository of *Jutge.org* [6].

If errors occur in this step we need to go back to step 3 and interpret what the compilation error is telling us about why our program is not able to compile.

5. **Executing the program.** The compiled program can be executed with any test case we might want to try. It is a good idea to start with test cases that we know well from step 1. The result could be that the program generates output that we can go ahead and verify in the next step. Another possibility is that the program enters an infinite loop and never outputs anything. Finally, the program could crash with or without an error message. In the cases in which we don't get any output we know that our problem "has a bug" and we have to start

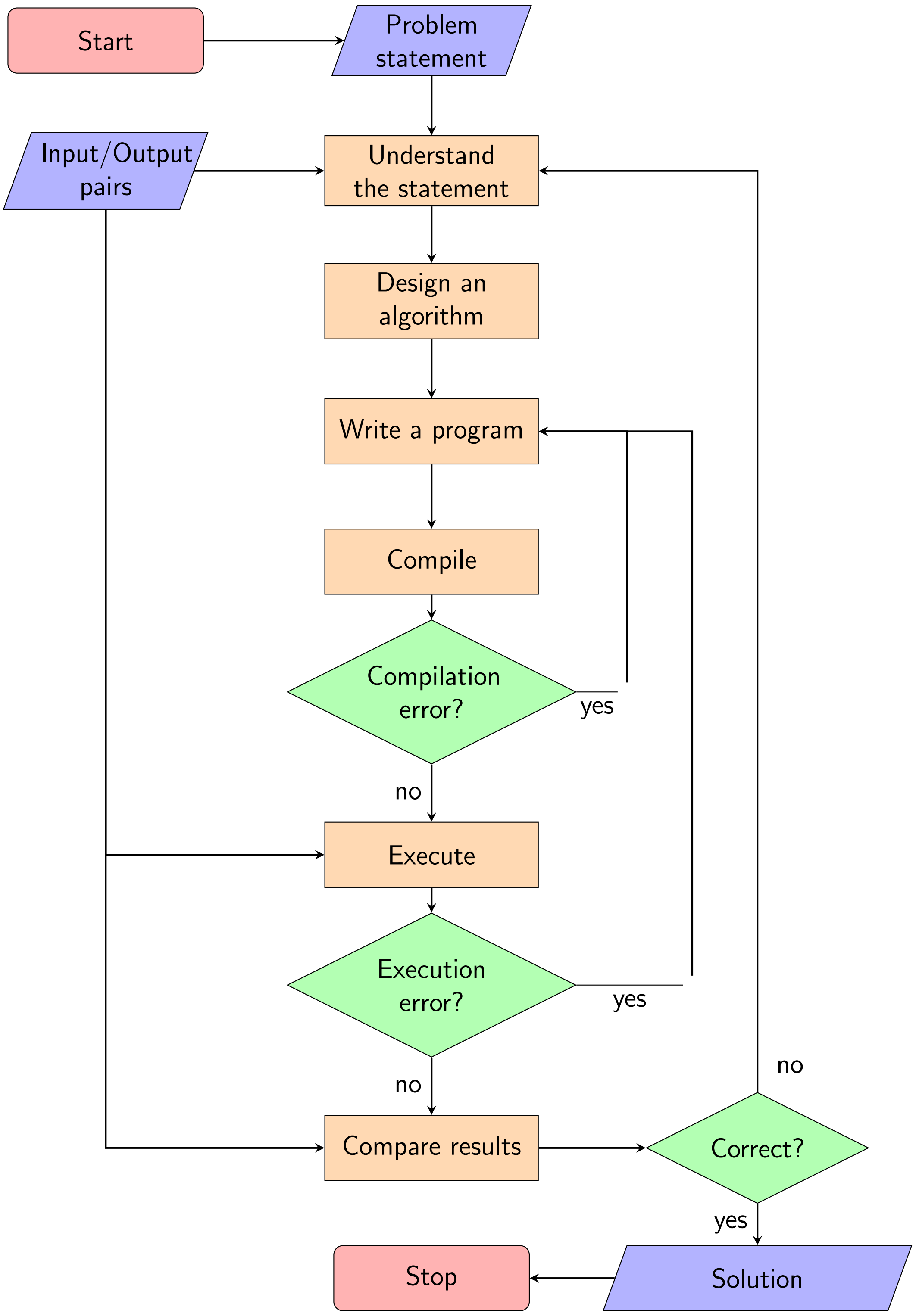


Figure 3: The workflow of solving small programming tasks.

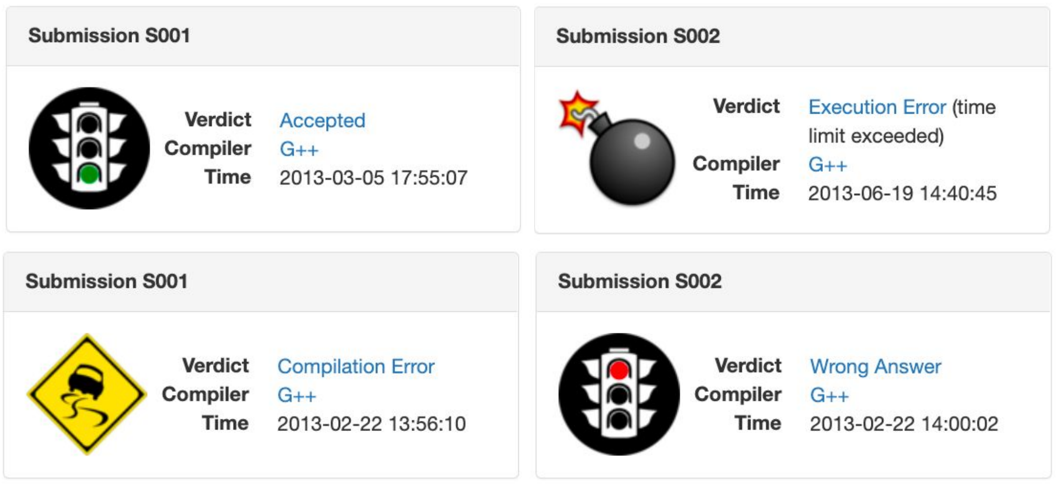
debugging it. This means going back to step 3, and modifying the code.

6. **Verifying the output.** Once we have managed to get some output from the program we can check whether it is consistent with what we expected to get based on the problem statement and the pairs of input/output examples. If it is consistent, our program might be correct. If not, we have to go back to 1 and 2 to check if we had really understood the problem statement corrected, and whether our algorithm when run by hand gives the correct output or not. Our mistake could be hiding anywhere between steps 1 and 3.

When the problem we are solving is from the repository of an automatic judge, at the end of the above process, we can submit the program to the judge for evaluation. The server of the judge will run many more test cases and there are several different possibilities for the outcome. The program can either be accepted by the judge (verdict AC), or we could be told that it failed for one of the following reasons:

- Wrong Answer (WA): the output to one of the test cases is not the expected one;
- Execution Error (EE): the program crashes during execution on at least one of the test cases;
- Time Limit Exceeded (TLE): the program takes too long to finish;
- Memory Limit Exceeded (MLE): the program uses up all of the memory allowed;
- Compilation Error (CE): the program doesn't compile.

Figure 4 shows the different types of *Judge.org* verdicts.



Submission S001

Verdict Accepted
Compiler G++
Time 2013-03-05 17:55:07

Submission S002

Verdict Execution Error (time limit exceeded)
Compiler G++
Time 2013-06-19 14:40:45

Submission S001

Verdict Compilation Error
Compiler G++
Time 2013-02-22 13:56:10

Submission S002

Verdict Wrong Answer
Compiler G++
Time 2013-02-22 14:00:02

Figure 4: Verdicts that the automatic judge can give depending on the outcome of running the program on the full set of test cases. The images are taken from *Jutge.org* [6].

A program accepted by the judge is not guaranteed to be correct. From getting the AC verdict we can only conclude that it has passed the test cases of the judge, but it could still fail on other test cases. However, good test-case design covers all the edge cases and likely errors, so in practice this is a rare situation. In any case, even if the program works correctly, it might be of bad quality. It could be very hard to read and interpret, or it could be doing unnecessary operations. Ideally, a teacher should still look at the accepted submission and give comments about style or efficiency.

What are the savings for the teacher, then? Primarily, the savings come from the detection of wrong algorithms. It is very common for students to be under the impression that they have finished a problem, when in fact it doesn't work correctly even on the test cases that are given with the problem statement. There are even cases when a student is under the impression that they have solved the problem, before they have even compiled it. Having the very precise objective of passing the judge's check is an excellent way to make sure that students reach the end of the workflow in Figure 3 before submitting their work for evaluation by a teacher.

Ultimately, we would like to have a judge that also gives useful information, similar to the one that might be given by a human teacher, in the cases when the program fails to pass the tests. One such piece of information is the actual test case that has failed, and some judges provide this information. An even more useful piece of information would be a hint about where in the algorithm the student might be making a mistake. We expect teachers to have an advantage in such tasks over computers, because they have personal experience with human error.

The task of finding errors in programs is computationally a very difficult one [7]. Theoretically, by Rice's Theorem the decision problem of whether a program is correct or not is an *undecidable* one [8]. This doesn't mean we should give up looking for ways to find errors automatically, because we can still hope to be able to find those in most programs written by human programmers. In fact, the necessary technology might be just around the corner. In Section 3.4 we will discuss some advances in AI research that make this seem ever less implausible.

3. Literature review

3.1 History of automatic judges

Interestingly, the first automatic judge was created in the 1960s in Stanford for grading student's programs in ALGOL (one of the first programming languages) [9]. Other early automatic judges were implemented for carrying out the international programming competitions at both university and highschool levels [10, 11]. The most important competition for university students is the International Collegiate Programming Contest (ICPC) [10] which was established in 1977. For highschool students it is the International Olympiad in Informatics (IOI) [11], established in 1989. The IOI is an individual competition, and students represent their countries, while the ICPC is by teams of 3, and the teams represent their universities [10]. They are both annual competitions.

The first automatic judge system that gained high popularity worldwide is the *UVa Online Judge* [12]. It was created in 1996 in the University of Valladolid by Miguel Ángel Revilla, a mathematician who lectures on algorithms at the University of Valladolid in Spain. Its purpose was training for the International Collegiate Programming Contest. It has a repository of all major international competitions at university and highschool levels, and by now has accumulated more than 250,000 users. The problem archive of the *UVa Online Judge* has over 5,000 problems. The entire website is in English.

The history of *UVa Online Judge* is described in [12]. The article [12] from 2008 also discusses some of the kinds of data the owners of the judge have collected through the years. This data can be useful for designing pedagogical systems, and the authors describe efforts towards creating a pedagogical tool based on the technology and data from their judge, as part of a large collaborative project funded by the European Commission, called *EduJudge*. In 2022, the *UVa Online Judge* is still very popular, but it has not become any friendlier to beginner programmers, such as the first-year students in vocational training programs. The "edujudge.eu" website associated with the project is not functional.

Most early articles on the subject of automatic program evaluation, such as [13], which describes an implementation in the National University of Singapore in 1999-2000, mention courses for preparation for programming competitions as their main application. In [13], the authors describe many of the technical issues around implementing an automatic judge system.

In the early 2000s several open-source systems were created that made it much easier for institutions to develop their own systems by modifying the existing ones, and for creating their own repository and competitions (or exams). The most famous open-source automatic evaluation systems *Mooshak* from the University of Porto was described in [14] in 2003. Another open-source system that is very widely used, especially for hosting contests, but also for creating judges for use in the classroom, is *DOMjudge*.

One authomatic judge that is very relevant to our setting is the system created in Universitat Politècnica de Catalunya (UPC) by Prof. Jordi Petit in 2006, the *Jurge.org* [6, 4]. It was created to reduce the grading load on professors of the introductory programming course in the Faculty of Informatics. An example of a task from *Jutge.org*, and different possible verdicts were shown in Figures 2 and 4. Unlike the *UVa Online Judge*, the *Jutge.org* allows for the creation of users with instructor privileges who can create groups of students, and access their submissions. The added functionality of users with different privileges is very important for the possibility of using these evaluation systems in introductory classes. In Section 5.1 we will discuss the different features that make a judge better suited to classroom teaching.

The organisation of the site of *Jutge.org* is by courses so that teachers can easily find the material of the right level of difficulty. There has been significant research supporting the use of *Jutge.org* in programming classes at UPC [15], as well as a cost-benefit analysis with respect to the effect of different assessment strategies applied through the years [16].

There are many more papers analysing different automatic judge systems, however most of them are about systems that are property of the institution where they are used, and are not open-access. For a rather exhaustive list of the open-access ones we recommend an article on *Programador Clic* [17]. There are also commercial systems, which are not usually studied in the academic literature. A meta-study of published papers on automatic judges was carried out in [1]. The results are summarised in Table 1 taken from that paper. AAT stands for "automated assessment tools".

Another positive experience with an automatic judge in Spain is reported by professors at the University of Murcia in [18]. In a second-year programming course they achieved a decrease in the dropout rate from 75% to 45% by introducing programming competitions with a judge built using the system Mooshak [14].

In the context of vocational training programs in Spain, the judge that has been most influential

Have AATs proven to be helpful in improving student learning?	Yes
Do students think that AATs have improved their performance?	Inconclusive
After having used the tools, do instructors think that the tools have improved their teaching experiences?	Yes
Is the assessment performed by AATs accurate enough to be helpful?	Yes

Table 1: Table of conclusions of the study in [1].

is *¡Acepta el Reto!*. It was built originally for use in classes in the Complutense University in Madrid in 2014 by Professors Marco Antonio Gómez Martín and Pedro Pablo Gómez Martín. It was first described in a research paper in 2018 in [19]. The document serving as a Project Proposal for innovation from 2015 is also available online [20]. The judge *¡Acepta el Reto!* is used for programming competitions for vocational training students called ProgramaMe. We will explain the details of this competition in Section 4.1.

Since we are interested mostly in judges that use Spanish or Catalan, we should mention the largest automatic judge system based in Latin America, specifically in Brazil. It's called *beecrowd*, but it is better known by its previous name - *URI Online judge* [21]. It's available in English, Spanish and Portugueses. It was introduced in 2012 in Universidade Regional Integrada (URI) at Erechim in Brazil. Currently, there are more than 2300 programming challenges in their repository.

Finally, a very extensive classification of automatic judges can be found in [22]. A notable exception is *¡Acepta el Reto!*, because only judges with an English language option are included. The judges are classified in six different categories according to their main intended applications. Out of these six, four categories are relevant to us, or include judges that have already been mentioned:

- For holding competitions: the list includes the *UVa Online Judge* [23], *Codeforces* [5], *Topcoder* [24] and 39 others;
- For educational purposes: the list includes *Jutge.org* [6], *URI Online Judge* [21], *Exercism* [25], *Codewars* [26], *RACSO* [27] [1] and 13 others;
- For recruitment: the list includes *HackerRank* [28], *LeetCode* [29], and 4 others.
- For development of judges: the list includes *Mooshak* [14], *DOMjudge* [30] and 11 others.

3.2 Relation to gamification

Although it is not necessary, some automatic judge systems apply gamification concepts to attract users. Many of them use leaderboards ([5, 31, 32, 26]), and some use points, badges and avatars (e.g. [31]). There are, however, much more sophisticated ways to convert an automatic program

[1] *RACSO* is another evaluation tool created at UPC, specifically for the Theory of Computing class, for evaluating formal languages, formal reductions, etc.

evaluation system into a game. For example, the user can be given the task to create a software agent (small program) for solving tasks within the game. An example of such a system is Asura of [33]. In the same article [33], there is a list of more than a dozen other systems existing in the research literature, although most of them are probably not available for public use.

3.3 Relation to course organisation ecosystems

Whereas we are concentrating on program evaluation, an automatic judge can be considered just a small part of a larger system of course organisation. For an example of a complete architecture for such a system we refer the reader to [34]. For an alternative way of integrating an automatic judge and a learning management system for a course using blended learning, we recommend the work of Georgouli, Skalkidis and Guerreiro in [35] and [36].

3.4 Relation to developments in AI technology

There are several aspects of automatic judges that can be addressed using AI and data-driven technology. The most important one is feedback generation. See for example recent progress with a system used in Rutgers University in [37]. Unfortunately, we don't know of any such system that is open-access. The latest meta-study of the literature on automated feedback generation systems that we are aware of is from 2015 [38].

Among the open-access judges that we introduced in Section 3.1, there are also efforts to create systems of feedback and hints. There is a beta version of *¡Acepta el Reto!* with hints on the site [39]. However only 25 exercises are available as of June 2022. The research effort is described in [40]. With respect to the UPC judge *Jutge.org*, some important progress for generating feedback was reported in [41]. The approach taken in this work is to find test cases that are most likely to give useful information to the user about why their code fails. However, as the authors note "an important question that remains to be addressed is how to use these distilled sets in the workflow of educative online judges". As of 2022 the idea has not been applied to the general site yet.

Another application of AI to automatic judge services has as an objective the personalization of the learning process via recommender systems. A recommender system would suggest to the user the next task to work on, based on their history and on data from the submissions of previous users with similar histories. In this way, the learning process is personalised based on the strengths and the weaknesses of the student. For some advances in this direction we refer the reader to the work of Fantozzi and Laura [42] and the references therein. Specific developments of recommendation systems for the judge *¡Acepta el Reto!* are presented in [43] and [44].

Finally, there is progress in AI research that one could argue makes it less necessary for humans to learn how to program. The reason is that AI is ever more capable of substituting human programers. We prefer to look at these technologies as a warning call that programming abilities are becoming more important than ever, because we believe humankind should stay in control of the software creations of AI agents. The concrete advancement that is very relevant to the topic of this work is AI technology capable of generating programs that solve problems stated in natural language, exactly in the format of the regular automatic judges. A recent advance of the company DeepMind, presented in [45] and made public in March 2022, is an AI system that performs as well as a medium level human user on computer programming tasks from the repository of the *Codeforces* automatic judge.

4. Context of vocational training

The general trend in the world is to look for ways to nourish computational-thinking skills in *all* children from young age (see the European Commission report [46]). However, in Spain, there is no programming requirement in secondary education, beyond the development of some basic competences in the subject of "Technology" in secondary education, some of them extended in the recent legislative modification of the LOMLOE[2]. Programming is not included in the curriculum of tertiary-education ("cicle formatius de grau mitjà") of informatics and communication degrees. Some schools, by their own initiative, include in their curriculum 1 or 2 hours a week of programming during the second year of the "cicle formatius de grau mitjà".

The programming courses in higher-education vocational training programs ("cicles formatius de grau superior") assume no previous knowledge of programming. There are two kinds of degrees - systems administration ("Administració de sistemes informàtics i xarxes"), with the abbreviation ASIX in Catalan and ASIR in Spanish; and software development ("Desenvolupament d'aplicacions multiplataforma/web"), with the abbreviations DAM/DAW. The programming class for systems administration students is significantly smaller - approximately 4 hours/week during the first year. For the development degree the programming class is the most important introductory class and can take 7-9 hours/week of class time during the first year.

4.1 ProgramaMe: annual programming competition

ProgramaMe is an annual programming competition held in the Spring semester for all vocational training students in informatics and communication. It is done in teams of 3 students, and each school can send one or two teams. The competition is executed with the automatic judge *¡Acepta El Reto!* which is the most widely used automatic judge in vocational training programs. In Figure 5 we have plotted the number of schools participating in this competition in Spain, and in Catalonia specifically, since the year 2014. These numbers should be compared with the total number of schools offering the degrees of applications development (DAM and DAW), which in Spain in 2022 is approximately 515, and in Catalonia approximately 75.

In 2018 the competition *ProgramaMe* was part of the *Catskills* program [48], which is the local organiser of *Skills* competitions [49]. Before the year of the pandemic, 2020, the competition was held in two rounds - regional and national. It was always on-site. In 2020 it had to be cancelled due to the pandemic. In 2021 and 2022 it was held online. There is a surprising negative effect of distance learning and the Covid-19 pandemic on participation. Although it seems like it should be easier for schools to participate when the competition is in online format, there was a very important drop in motivation among students and professors in the year 2020. The team work culture also

[2] "Ley Orgánica 3/2020 (LOMLOE)" dated 29 Dec, 2021, modifying the previous law "LOE 2/2006 (LOE)" of 3 May,2006.

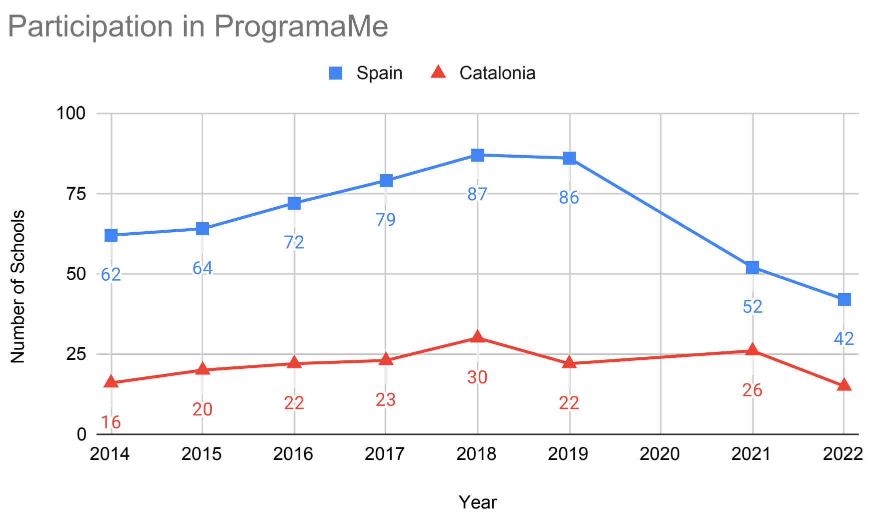


Figure 5: The number of schools participating in the annual ProgramaMe competition since 2014. Data taken from the site of ProgramaMe [47].

suffered and this has made it more difficult for schools to create teams. For schools that use *¡Acepta el Reto!* as a tool in the classroom it is significantly easier to find a team.

4.2 Context of our surveys to teachers

Teachers in vocational training programs of informatics have several informal channels to keep in touch with each other. There is a Telegram group for teachers from all of Spain with 1309 members, and a Discord group with 869 members. In addition, there is a Telegram channel dedicated specifically to the ProgramaMe competition with 112 members. Finally, there is a Telegram channel for techology and informatics teachers in Catalonia with 584 members.

The survey in Appendix A was sent to all these channels, asking for collaboration. The questionnaire was prepared with *Google Forms*, was available online, and could be filled both in Catalan and in Spanish. The responses are anonymous. We received 40 answers from teachers who have given programming in vocational training programs, out of which 24 are teachers who have used judges in class, and the rest haven't. Out of the 16 teachers who haven't used automatic judges in class, 7 said that they had considered it in the past.

Since the sample of teachers is not uniformly random, and instead was based on the choice of the teachers to fill out the survey, it is not possible to deduce what percentage of teachers use automatic judges. Among the 40 respondents of our survey, there is only one who has used *¡Acepta el Reto!* in class, but has not had students participate in ProgramaMe. On the other hand, there are many examples of schools that use *¡Acepta El Reto!* only with a small group of students during the competition and for preparation. Among the teachers who responded to our survey 7 out of the 16 who have not used judges have had students participate in the competition. This data suggests that the numbers of participation in ProgramaMe can be used at least as an approximate upper bound on the number of schools that use *¡Acepta el reto!* in the classroom. That number is approximately 20%.

We can also conclude that our sample of teachers (who responded to the survey) is very heavily biassed towards teachers who have used judges, and ones who have participated in ProgramaMe. In our sample 70% of the teachers have participated in ProgramaMe, which is very high in comparison with the approximately 20% of schools out of the total that have participated.

5. Review of automatic judges

In this section we will discuss automatic judges that are available to teachers in vocational training programs in Spain. They are listed in Table 2, ordered according to how many people have used them in class among the 40 respondents to our questionnaire to teachers. The frequencies are shown in Figure 6. The judge *Gripau* is not included in the table, because we don't have access to it (see Section 5.3 for an explanation).

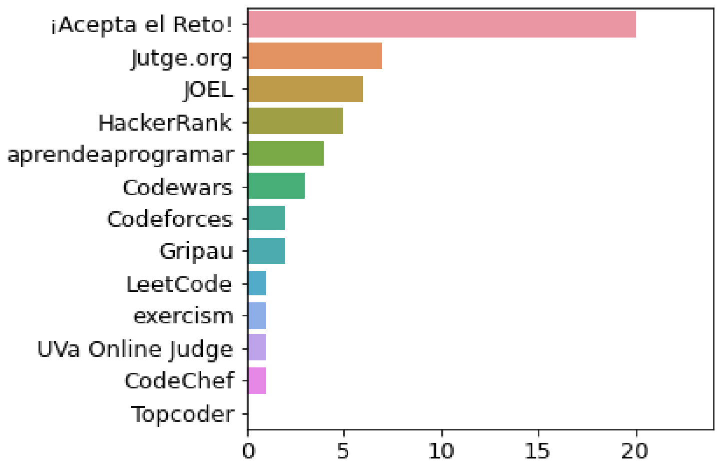


Figure 6: A histogram of how many respondents have used each judge based on the answers collected for Question 4 of the questionnaire in Appendix A.

In the next section we will describe the automatic judge features that most affect a teacher's decision to use a specific one in the classroom. Next, in Sections 5.2 to 5.5 we will introduce all the automatic judges in our list that were not already introduced in Section 3.1 (because they do not appear in the academic literature, as far as we know).

5.1 Features of automatic judges

The first two considerations when choosing an automated evaluation tool are (1) the language in which the problem statements are given, and (2) the programming languages that the judge accepts.

We have presented in Table 2 the information about each judge with respect to the availability of English, Catalan, and Spanish, and of the most common programming languages used in introductory classes.

	¡Acepta el Reto! [32]	Jutge.org [6]	JOEL [31]	HackerRank [28]	aprendeaprogramar.org [50]	Codewars [26]	Codeforces [5]	LeetCode [29]	exercism [25]	UVa Online Judge [23]	CodeChef [51]	beecrowd (URI) [52]
English	✗	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓
Catalan	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
Spanish	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓
C/C++	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
Java	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Python	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Rich repository of introductory problems	✓	✓	✓	✗	✗	✓	✓	✗	✓	✗	✓	✓
Classification of tasks by programming concept	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗
User "Teacher" with acces to students' submissions and progress	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓
User or their "Teacher" has access to failed test cases	✗	✗	✓	✓	✓	✗	✓	✓	✓	✗	✗	✓
Scheduled tasks	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓
New problems can be added to the repository	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗
Access to solutions of other users	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗
Forums/comments	✗	✗	✓	✗	✗	✓	✗	✓	✓	✗	✗	✓
Recruitment	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓

Table 2: Availability of most essential features for the set of judges under consideration.

Next, we need to consider how we want to use the judge in class:

Sporadic use. All of the available automatic judges have something to offer for sporadic use. Naturally, it would be easier to make good use of a platform that has a rich repository of introductory level tasks, and on which the tasks are categorised by the concepts practised (such as conditionals, loops, arrays, string, etc). A good example of such a judge is *¡Acepta el Reto!* [32]. For such sporadic use, the teacher just has to select the problems that best capture the concepts of the lesson, and ask the students to work on those problems. We have to keep in mind that on some platforms - such as *CodeChef*, solutions are readily available to the user [51]. These platforms can be used only with students who are not going to look for a shortcut to the solution.

Regular use. In order to use a judge on a regular basis we would need to be able to keep track of the progress of all students in class over a sequence of weeks. For this, the automatic judge should have the feature of a user with privileges ("Teacher") with access to the submissions of other users. This feature is offered by only a very small number of automatic judge platforms. Among the platforms that have a "Teacher" user we can distinguish between those that give access to the test cases that the submitted code has failed, and those that don't. Most notably, *Jutge.org* does not give access to such test cases. On the other hand there are many automatic judges that give access to this information even to the regular or "Student" user.

All judges that have a "Teacher" user give such a user the ability to assign scheduled tasks. Another useful feature is the possibility to add new problems to the repository, so that the teacher can create automatically graded exams.

The last three features in the table are not of great importance, but they give the user a sense of being part of a community. The first is access to other user's solutions once a challenge is solved. This can be very useful for comparing different styles of solutions. One can learn a lot from another user's code, especially if it happens that the other user is more advanced. The second feature is the availability of forums or comments under the challenges where students can discuss ideas, ask about problems they run into, or help each other out with hints. Finally, several sites are also used for job recruitment, and preparation for interviews. We will discuss these in Section 5.5.

As discussed in Section 3.4 a very valuable feature for an online judge is the ability to give non-binary feedback and hints to the user. In 2022 these capabilities are still very rudimentary for all the judges that we have considered, and are still under development, so we have not included this feature in Table 3, but it is a very important aspect to be considered in future works, and when making recommendations to teachers.

In Question 16 we asked teachers who have used automatic judges in class how much they value each of the features described above, and three more. In Table 3 we show the average value for each feature[3] on a scale from 1 to 4.

[3]The feature "Classification of the tasks by programming concept" was not included in the questionnaire, because it was identified as an important one in a later stage of the research.

Feature	Avg. value
There is a rich repository of introductory problems.	3.3
User "Teacher" has access to students' submissions and progress.	3.3
User or their "Teacher" has access to failed test cases.	3.3
New problems can be added to the repository.	3.3
Student is provided hints when a program fails.	2.9
"Teacher" can create scheduled tasks.	2.5
There is space for giving feedback to students' submissions.	2.5
Student has access to other's solutions after passing a task.	2.3
Grades can be transferred automatically to Moodle.	2.2

Table 3: The average of the value that respondents gave to each feature on a scale from 1 to 4 (based on 24 replies by teachers who have used judges in the classroom).

5.2 Platforms created in vocational training institutions

In addition to the judges created by universities - *Jutge.org*, *¡Acepta el Reto!*, *URI Online Judge* and *UVa Online Judge*, that we already introduced in Section 5.1, there are judges that were created in Vocational Training schools, and do not appear in the academic literature.

JOEL [31] The name is an abbreviation for *Free Online Judge for Education* ("Jutge Online Educatiu Lliure"). It was created by Marc Albareda in IES Sabadell and was open for student use during the pandemic in 2020 to help them work from home. It is now used by 3 different schools. It has some features that make it very friendly for the setting of vocational training programs. In 2022 it was recognized by the Catalan department of education in their program ImpulsFP as a valuable innovation, and is going to be implemented in more schools during the next academic year.

Among its features, it's worth mentioning the fact that the feedback is not binary. The student is given partial credit if their program passes some of the test cases. There are hints, and also the possibility of adding new problems. Several competitions have already been hosted on site too, so it has been shown to be robust, too.

aprendeaprogramar.org [50] This site was built in 2017 by Sergi García of IES Serra Perenxisa. It is organised by courses on different programming languages, so that the students have a fixed itinerary to follow. There is no capability for a "Teacher" user, which makes it harder to monitor students' progress, so it is more appropriate for sporadic use. One advantage of the site is that it is built with Moodle, so its interface is very familiar to both students and teachers.

5.3 Closed-access and paid platforms

We are aware of at least one example of a school that uses its own automatic judge in the vocational training program, which is not made available to the wider community. It's called *Gripau*. There may be others that we are not aware of.

In the last few years several paid platforms have been introduced, but due to their commercial nature it is hard to review them. For the same reason they have not been adopted by many schools yet. Concrete examples, specifically located within Catalonia, are *CodedArena* [53], *Leagues of Code* [54], and *Codelearn* [55]. Their primary market for the moment are extracurricular activities and programming classes at the secondary education level in private schools.

5.4 English-language options

Some of the most popular English language sites that can be used as automatic judges are *Codeforces* [5], *Codewars* [26] and *exercism* [25]. The oldest one is *Codeforces*, and, although it is primarily designed for competitions, it is a very rich source of good problems of all levels. It has the added advantage that it offers very frequent online competitions. In each competition there are at least one or two problems that can be tackled by beginner programmers. *Codewars* also has a competitive spirit, but its gamification system is much more complicated than just leaderboards, which can be very motivating for some students. It is modelled after martial arts and the challenges are called “kata”. Finally, *exercism* is a relatively new site, but has much less competitive and more pedagogical spirit. Anyone can add exercises, and act as a mentor. There are tracks for the different programming languages. In none of these three sites there is a role of “teacher” or “instructor” who can follow the progress of a group of students. For this reason, they are primarily suitable for sporadic use.

5.5 Sites for job recruitment that are also automatic judges

There are a number of popular sites that have automatic assessment capabilities, but their primary purpose is not education. This is the case of *HackerRank* [28], *LeetCode* [29], and *CodeChef* [51]. They are for-profit sites and their business is head-hunting. They are a good place for employers to find good programmers and for good programmers to exhibit their skills. They can be useful as automatic judges for sporadic use, because they also expose the students to the market of programming jobs.

The site *beecrowd* [52] is a very new site of this type, but it is really the sequel of the academic *URI Online Judge* [21]. It has an academic section, which allows for the creation of instructor accounts, granted by request. Currently the service to academics is free. Given that many of the problem statements are available in Spanish, and that it has the nice features of academic judges, it's a good alternative for teachers in Spain, which seems to not be very well known. None of our respondents quoted it as a site they use in Question 4. We were also not aware of it until the later stages of this research.

6. Attitudes towards judges among teachers

We have said a lot about the advantages of using an automatic judge, but there are also some difficulties, and there are teachers who are generally against their adoption, independent of how well these evaluation tools might be made, or what nice features they might have. In this section we will use the results from our questionnaire to try to put some light on the reasons different teachers give for using them and for avoiding them. We have 40 replies, out of which 24 have used them, and 16 have not. In Section 4.2 we explained how the respondents were selected.

In Section 6.1 we will try to clarify how familiar teachers are with the available options. In Section 6.2 we will study in detail teachers' opinions and give a ranking of the reasons *for* and *against* the use of judges. Section 6.3 is about the experience of teachers who have used the judges. In Section 6.4 we will give data about the degrees and teaching units in which judges are useful within the Vocational Training curriculum for Informatics in Spain. Finally, in Section 6.5 we will describe our attempt to correlate the use of judges to other characteristics of the teacher's methodology or experience.

6.1 Familiarity with automatic judge options

We asked teachers who haven't used automatic judges about their opinion and familiarity with the concept. In Figure 7 we show how many people have heard of each judge from Figure 6, out of our 16 respondents.

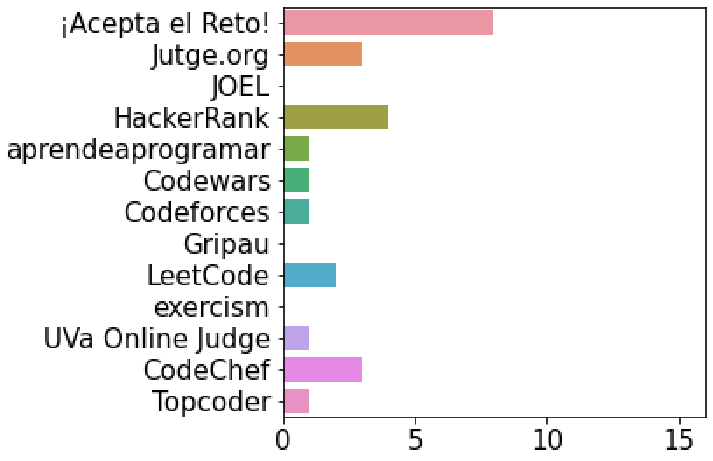


Figure 7: The number of respondents who have heard of each judge, out of the 16 respondents who don't use judges in class.

The judge of IES Sabadell *JOEL* is very new so it is understandable that none of the teachers who haven't used judges have heard of it. On the other hand, it is surprising that only half of the respondents have heard of *¡Acepta el reto!*. For this judge, as well as for *Jutge.org* and *UVa Online*

Judge, we also included the name of the university from which they come, respectively Complutense of Madrid, UPC and University of Valladolid. Even so, only 3 people out of 16 were aware of *Jutge.org* and only one of *UVa Online Judge*. The last one has worldwide fame, so it is a bit sobering that it is unknown in its own country.

Some of the comments of people who use the judges also indicate that teachers are not exposed to enough options of different judges:

> *"The possibility to work in different programming languages on the same challenge is something that few platforms offer."*

In our list of judges, the majority let you choose the language in which you want to submit each problem. Perhaps this respondent is referring to platforms that have courses or tracks tailored to particular languages. This is the case only for *aprendeaprogramar.org*, *exercism* and *HackerRank* out of the 12 judges in our table.

> *"Another point against is the editing of the statements, because many are not understandable (not even I, as a teacher, am able to understand them). In addition, they contain many spelling mistakes."*

Out of the judges we have considered, the only one that has typos or spelling mistakes is *JOEL*. It's understandable, because it is very new, and still edited on the fly, and, also, problems can be added by students, and although they are monitored the process is still very informal. Other judges don't have this problem. The problem statements in many repositories (especially the ones created for competitions) can be long and elaborate, and one has to be able to filter out the technical content, but this is not unusual for real-life projects, so perhaps should be considered a plus.

> *"They can save work for the teacher as long as the solution is the one that is expected, but solutions can come up that are not expected by the professor that might be better and they will count as wrong. It does not help to automate the work of the teacher, work that could disappear due to the substitution of these automatic tools. It would be much better to hire more teachers to do the evaluation and I want to comment that it closes the results to a programmatic thought without common sense."*[sic]

In this response there is an important misunderstanding about how judges work. Automatic judges do not consider how the problem is solved, but only whether the output is correct [3]. For this reason, there can be many different solutions, and they will all be accepted, as long as they work.

6.2 Pros and cons of judges according to respondents

Among teachers who have considered using judges, but have not used them yet, 7 out of 7 said that the reason they decided not to use them was that they didn't have time to research the different options. Among teachers who have not considered using judges, there were some interesting comments about their reasons. Here are some quotes:

> *"I believe that the judges are a complementary way to know if the programs work or not, but the interaction of the teacher with the student to see the functioning of the programs provides a qualitative element that the judges do not have."*
>
> *"It facilitates the work, but depersonalises and reduces the value of the teacher and the motivation of the students, there is risc of layoffs for teachers and lack of medium-term innovation."*
>
> *"It is a good way to improve students' algorithmic reasoning, but I think (without much experience) that for those who are weak in programming it can be discouraging."*
>
> *"The truth is that I have considered it on occasion, but I think that in the end a judge checks that what was requested is fulfilled but does not check if an efficient, clean code, etc. has been used, which is verified by seeing what they do and giving personalised feedback. Then there are the judges like ¡Accept el Reto! that, if I'm not mistaken, are thought for computer competitions that involve several test cases, etc. which I personally find a bit of a drag and unrealistic. I really like Codewars and use it to come up with ideas, but I have never proposed it to my class. Maybe I should give one of those a try ... "*
>
> *"The judge does not act as a guide and I think it is an important task in order for the students to improve."*
>
> *"The judges will create teachers who will not correctly carry out the function of educators."*

To get a more global and less personalised view of the reasons *against* using judges, we offered a set of reasons and asked the teachers to evaluate how valid those reasons are. The responses are summarised in Table 4. It's interesting that the reasons that teachers who have not used judges consider the most valid, are among the least valid for the teachers who have used the judges in class. Among the teachers who have used judges there are also reservations about their effectiveness.

> *"Really, the judges still have to get better, because they evaluate the result of the algorithm itself, but they do not value the quality of the code. It's this point where the judges play pretty bad tricks, because students can take on very bad vices."*
>
> *"I think the students have to learn to validate their own programs."*
>
> *"[The automatic judges] cannot be integrated into Moodle and there are solutions that can be copied over the internet."*

All of these are valid points, and should be taken into account when using a judge in class. We also ranked the reasons *in favour* of using judges, but in this case we only asked teachers who have some experience with them. The results are summarised in Table 5. Some more reasons that teachers wanted to add as positives can be found in Table 11 in Appendix C under Question 12.

Reasons *against* using automatic judges	Avg. validity according to users	Avg. validity according to non-users
The repeated rejection of programs can be frustrating.	2.9	3.9
It's easy for the professor to stop reading students' submissions.	2.9	3
Programs can pass the judge and still be very bad.	2.9	3
The automatic judges generate competitiveness.	2.4	2.9
The problems in the repositories are too difficult for beginners.	2.3	3.8
The output has to be exactly in the right format to pass the judge.	2.1	4

Table 4: Reasons *against* using an automatic judge ordered by the average value of validity given by respondents, on a scale from 1 to 5. The average estimated validity is given separately for the respondents who have used judges in the classroom, and those who haven't used them.

One reason that was mentioned by three different respondents was that the automatic judges are an excellent source of additional problems for students who are more advanced than the rest. Another important reason that was quoted is that it is a very good monitoring tool. Finally, three respondents mentioned technical knowledge that is better acquired while using automatic judges. Specifically, the memory and time limitations are very important in practice, and gain special protagonism in introductory classes that use judges, as they can be one of the reasons for negative verdict.

6.3 Teachers' experiences of using judges in class

In Question 8 we asked teachers who have used automatic judges about their level of satisfaction. The question was posed giving several different options regarding their experience (see Appendix A). We also asked them how much they would recommend the use of a judge to other teachers on a scale from 1 (No) to 5 (Yes) (Question 13). The answers are detailed in Figure 8. We can conclude that almost all teachers are positive about their experience, with the exception of between 1 and 3 out of 24 who are neutral.

The most important factor for deciding *for* or *against* the use of an automatic judge is the effect it may have on students' learning. We asked teachers what they consider this effect is in Question 9 and in Question 23. In Figure 9 we give a histogram of the answers in the two groups. The difference between the expectations of teachers who haven't used automatic judges and the experience of teachers who have used them is very large.

The questions were worded slightly differently, adapting to the two different situations, as you

Reasons *in favour* of using automatic judges	Avg. validity
The repository of challenges is a source of good exercises.	4.2
The format motivates the students.	4.1
The feedback is instantaneous.	4
They teach to follow rigorously technical specifications.	3.7
They improve the students' reading comprehension.	3.1
They are an important time saver for the instructor.	3.1
They create a sense of community among different schools.	2.9

Table 5: Reasons *in favour* of using an automatic judge ordered by the average value of validity given by respondents, on a scale from 1 to 5.

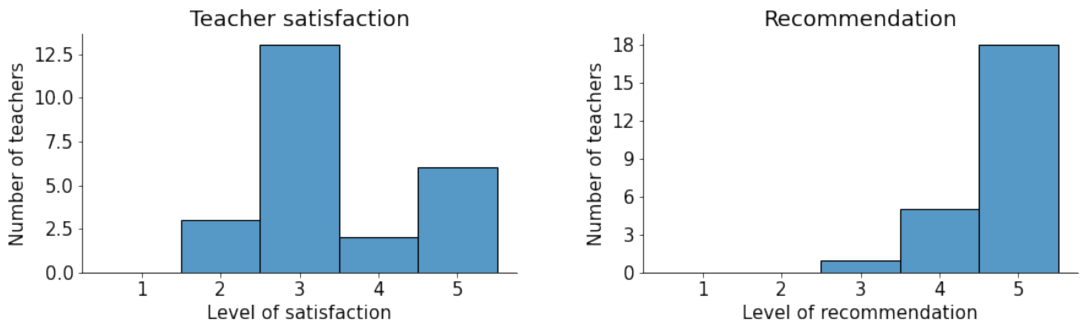


Figure 8: Quality of experience with judges, and recommendation to other teachers. In the question about the teacher's experience the options were (1) "I would rather not use them", (2) "I'm indifferent", (3)"They simplify my work", (4) "It would be hard to teach without them" and (5)"I find them indispensible".

can see in Appendix A. For teachers who have used a judge (Question 9) there were 4 options. In order to place the centre at 3 we number them from 2 to 5. The possible answers are (2) "They would learn more without a judge." (3) "There is no significant effect on their learning."(4)"They learn a little bit better with a judge" and (5) "They learn much better with a judge". For Question 23 it was just a scale from 1 to 5 where 3 means "no effect".

We asked teachers whether the benefits of using a judge might be different for students of different levels (Questions 10 and 24). In both groups only one teacher said that they are more beneficial for students with worse performance. The rest were approximately evenly split among the option that judges are equally beneficial to all students, and that they are more beneficial for the better performing students.

Finally, it is interesting to consider whether newer or more experienced teachers would have more difficulties adopting an automatic judge. On the one hand, one might expect that a more

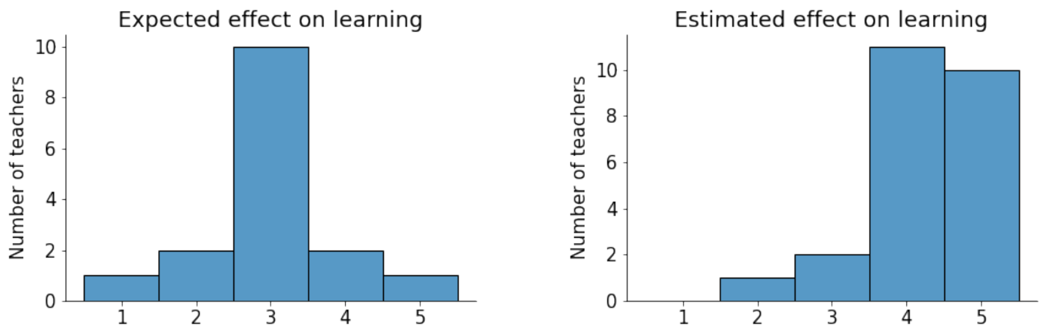


Figure 9: The relative effect on learning that automatic judges have according to teachers who haven't used them on the left, and according to teachers who have used them on the right. The third bin corresponds to no effect on learning, and higher bins indicate a positive effect on learning.

experienced teacher would be more resilient to change, and have more difficulty adopting automatic grading. On the other hand, having a lot of experience grading without automatic judges might make one better prepared to avoid the handicaps that an automatic grading method might have in allowing bad programming style. We asked the opinion of people who used them and the result is summarised in Figure 10. On average, they seem to consider experience to be a positive factor.

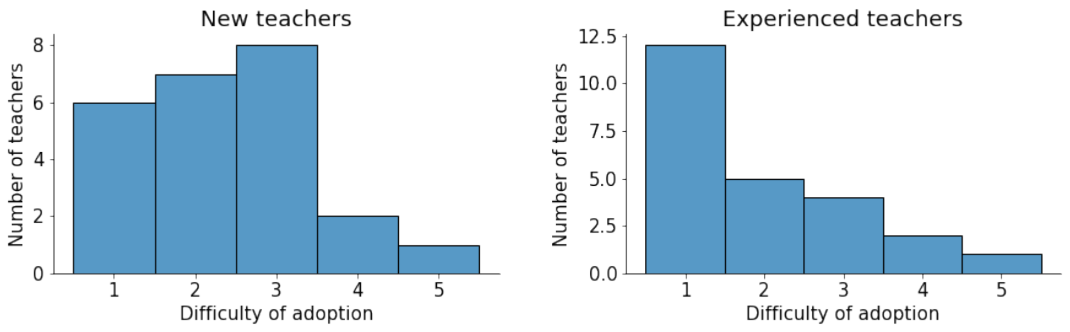


Figure 10: Level of difficulty of adopting a judge for new teachers, and experienced teachers, as estimated by the respondents who have used judges, based on their responses to Questions 14 and 15 in Appendix A.

6.4 Applicability to the vocational training curriculum in Spain

It's important to know what parts of the curriculum of vocational training degrees in Catalonia automatic judges are useful for. For this, we asked teachers in which units of the programming class they have used an automatic judge, and in which degrees. The results are summarised in Table 6. From these answers it appears that the automatic judges are used more often for the degrees on software engineering. In the next section we will discuss the different effects they appear to have on the attitude of students of the two degrees towards programming.

Teaching unit	Num. teachers
UF 1 - Structured programming	22
UF 2 - Modular design	14
UF 3 - File management	5
UF 4 - Fundamentals of object oriented programming	6
UF 5 - Libraries of fundamental classes	2
UF 6 - Introduction to data base persistence	1
Other classes (e.g. Data bases).	2

Degree	Num. teachers
DAM/DAW - Multi-platform/web application development	18
ASIX/ASIR - Systems and network administration	5
SMX/SMR - Micro-informatics systems and networks	2

Table 6: The number of respondents who have used automatic judges in each section of the programming course ("UF" stands for "unitat/unidad formativa" or "teaching unit" in Catalan and Spanish) and in each degree.

6.5 Correlations between the use of judges and other teacher or institution descriptors

We looked for correlations between the use of judges in class, and other variables provided by the teachers, however we did not find any significant correlations. Specifically we considered the following factors:

- number of different gamification tools used (Questions 33 and 34);
- years of teaching experience (Question 26);
- weather the home institutions offers the degree of video games development (Question 32);
- the percentage of submitted programs that are evaluated by hand, as reported by the teacher (Question 27);

- the average number of words in program evaluations, as reported by the teachers (Question 28).

There was no significant correlation between any of these factors and the use of a judge in the classroom. On the other hand, as we expected, there is a very strong positive correlation between the use of judges in the classroom, and participation in the programming competition ProgramaMe (Question 31). We illustrate this by a histogram of the number of years of participation in the two groups in Figure 11.

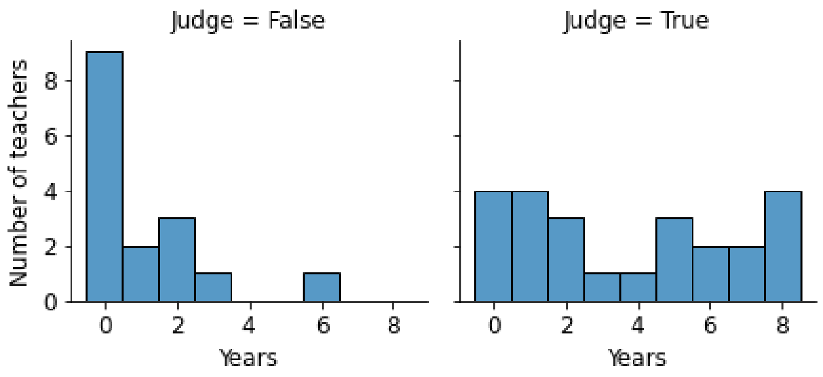


Figure 11: Number of years teachers have participated in ProgramaMe. In the left graph is the group who haven't used automatic judges in class, and in the right are those that have used them.

7. Study of effects on student learning and attitude

For our comparative study of the effect of using automatic judges on student learning and student attitude towards programming, we prepared a questionnaire with *Google Forms*. The questions are presented with their original text in Catalan in Appendix B. The questionnaire was sent to 4 different professors from 3 different institutes. One of these institutes does not use automatic judges (but participates in ProgramaMe). The other two use primarily the automatic judge *JOEL*, but also other tools.

The survey was passed to students of programming classes in their first year of a vocational training program. It was presented in the beginning of the class of programming during the month of May (approximately a month before the end of the school year). All the students were first-year students in the degrees of system administration (ASIX) or software development (DAM or DAW). We received 57 responses distributed by degree and usage of judges as shown in Table 7.

In order to make informed decisions about the use of automatic judges we have to define clear goals with respect to student learning, as well as define measures of whether those goals are better achieved with or without the automatic judges. Ideally, we would want to use a common test. An objective test of programming would necessarily involve an automatic judge for grading programs, so it is clear that the students who have already used such a judge in class would have an advantage. Therefore we propose using a different kind of test.

	ASIX degree	DAM/DAW degree
Use an automatic judge	8	14
Do not use an automatic judge	13	22

Table 7: Distribution of student respondents by degree and use of judges in class. Only students who have used a judge in more than one session are counted in the first category.

Specifically, we propose using a test of computational thinking ability. This kind of test doesn't involve programming, but it evaluates reasoning skills very closely related to programming. The concept of "computational thinking" has relatively short history, but it has been studied very extensively in the last few decades. It's a skill (or "competence") that many countries have already included into their school curriculums at all levels (unfortunately, Spain is not one of them), and in many of the others there are public initiatives for including it.

With the objective of popularising the teaching of computational thinking skills, in 2006 educators in Lithuania created an international contest named "Bebras" [56]. In the next section we describe some aspects of this contest and its importance in educational research. We describe how we used challenges from this contest to gain some first impressions of the effect of automatic judges on computational thinking skills of students.

In addition to computational thinking skills, we tried to evaluate how the attitude of students towards programming might be affected by the use of automatic judges. We describe the questions and the results in Section 7.2.

7.1 Testing computational-thinking using Bebras challenges

The Bebras competition [56] is an international competition of computational thinking, very similar in format to the Mathematical Kangaroo [57, 58]. It was created in 2004 in Lithuania, and it's named after the beaver, the engineer of the animal kingdom, which is called "bebras" in Lithuanian. In 2006, four more countries joined, and it has grown every year since then. Currently there are more than 50 members in the list of participating countries.

Bebras is a 45-minutes long competition, and it takes place in schools all over the world in the Fall approximately on the same date [56]. Students of all grades in primary and secondary school work independently on a set of problems chosen for their grade level. The problems have multiple-choice answers for ease of grading. There is an international committee that designs the challenges every year. The local committee for each country (or region, in the case of different languages within the country) is responsible for translating the problems and selecting ones that they consider appropriate for each specific grade level, based on the country's curriculum. Students' scores are ranked only within each country.

A presentation of the competition, its objectives and design, can be found in the paper by the creator of the competition Prof. Valentina Dagiene and Gerald Futschek [59]. The types of tasks and the skills that they require have been studied extensively by several groups in the last few years (ex. [60, 61]). Bebras Challenges have been used to analyse computational thinking skills in different

	Total answers	Avg. attepmts	Avg. correct	Avg. wrong
Using automatic judges	22	1.27	0.82	0.45
Not using automatic judges	35	1.91	1.26	0.66

Table 8: Shown are the average number of attempts out of the 3 possible attempts (for the 3 challenges), the average number of correct answers, and the average number of wrong answers in each group.

populations, such as students in Uruguay [62], and primary school teachers in Argentina [63] [4].

For this work we did not make a specific effort to select tasks that span a range of computational thinking skills, because most of the materials are not openly available. We used the most convenient ones - tasks that we could find translated in Catalan, and that would not require too long to read, so that the students don't give up[5]. All three questions are in the lower end of the complexity spectrum in the pool of challenges.

The objective was to check if we can detect significant differences in performance and participation between students who have studied with automatic judge and students who have not. The original hypotheses that students who have used an automatic judge are more accustomed to facing computational thinking challenges, and would be more accepting of them and better able to resolve them correctly.

The questionnaire was completed by 57 students. The results with respect to their performance on the three Bebras challenges is summarised in Table 8. We split the students according to whether they have used an automatic judge during the year (more than once) or not.

The original hypothesis was not confirmed, and actually the opposite trend was observed in terms of willingness to solve challenges. In both groups approximately 65% of the answers were correct, but there were more attempts among the group of students who haven't used automatic judges in class.

The complete distribution of answers is plotted in Figure 12. Each student is a point in one of the two graphs, depending on the use of automatic judges in class. The coordinates of the point show the number of correct and the number of wrong answers out of the 3 answers to the challenges. The x-coordinate is distorted a little in order to be able to visualise multiple data points (otherwise they would all appear in the same spot).

When we compare the performance of students who have used judges and those who have not used judges, there is a big difference in the number of participants who have made at least one mistake, and also in the number of participants who have given at least one correct answer. In the group of students who have not used judges 43% have made at least one mistake, versus 23% in the other group. A similar trend can be observed for the number of students who have given at least one correct answer: there are 66% in the group who have not used judges, versus 32% in the group that have used judges.

In Table 9 we summarise the percentage of tries and the percentage of correct answers. We split

[4]We have selected specifically Spanish-speaking examples, because these can be good sources of challenges for future work

[5]Students were told the questionnaire wouldn't take more than 5 minutes to complete, in order to assure some minimum level of participation.

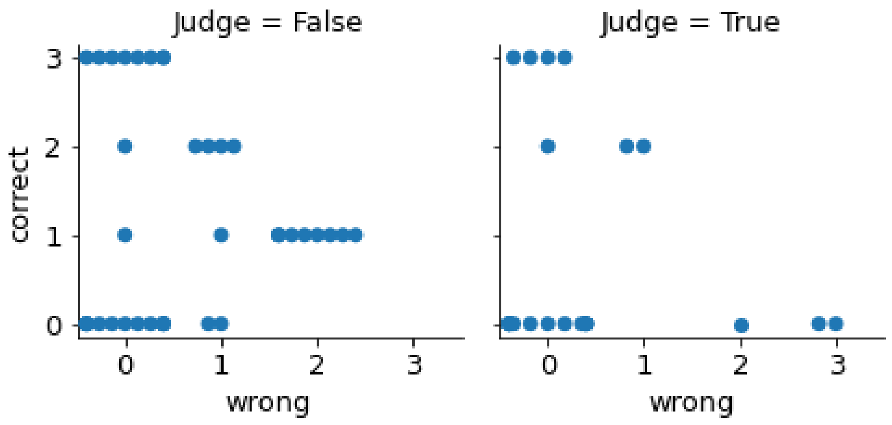


Figure 12: A plot of the number of correct and wrong answers on the 3 Bebra challenges for the 57 students, split according to whether they have used automatic judges or not in class.

	Attempted			Correct answers		
	DAM/DAW	ASIX	Total	DAM/DAW	ASIX	Total
Judge = False	56%	77%	64%	73%	57%	66%
Judge = True	48%	33%	42%	80%	25%	64%
Total	53%	60%	56%	75%	50%	65%

Table 9: The fraction of challenges attempted, and the fraction of correct answers (out of the attempts), split by degree and use of a judge.

the students by degree and by use of a judge. As you can see the results are very different for the system administration degree ('ASIX') and for the software development degree ("DAM/DAW").

Before we draw any conclusions, we need to point out that the conditions under which the questionnaires were presented to the students were not sufficiently identical. In all cases, they were presented by the teacher of a programming module in the beginning of a class during the last four weeks of the school year. Due to practical constraints, the wording while presenting the questionnaire was not predefined, and the questionnaire wasn't presented by a stranger. In some cases (among the classes that haven't used automatic judges) the students knew the researcher personally. Furthermore, we could not ensure that the timing is identical, such as right before starting a new topic. Considering all these different factors, we cannot rule out the possibility that their decisions were influenced by external factors. The results here should be thought of as a prototype for a more sophisticated experiment, which should take into account the different school's circumstances, and make the setting as uniform as possible.

With all this in mind, the numerical results are suggesting some hypotheses that might require further study. They seem to indicate that the performance is very similar between the two groups, but the willingness to accept a challenge seems noticeably higher among students who have not been using an automatic judge during the year. Perhaps towards the end of the year, the group of students who have seen a lot of challenges from automatic judges are more worn out.

Overall, the rate of 56% of voluntary participation in the challenges (95 answers out of 3×57 possible), as well as the rate of 65% of correct answers are both quite low, considering that the three Bebras challenges were designed for middle and highschool students who are not preselected for having special interest in STEM fields. The questions are included in Appendix B.

7.2 Effect of the use of automatic judges on students' attitude

From the analysis in the previous section, it seems like the differences that we are detecting between students who have used automatic judges, and those who have not used them are not in their computational thinking ability, but rather in their attitude towards computational challenges. In order to understand these differences better, we have to look at the students' responses to questions about their preferences and their self-evaluation of their abilities.

The purpose of this part of the questionnaire is to identify differences between the two groups in the following aspects:

- students' attitude towards programming, including their level of affinity to programming, and to what extent they think they are good at it;
- students' attitude towards competitions and team-work;
- the time students spend programming at home;
- students' satisfaction with the amount of feedback received;
- the number of programs students write in the course of a year,
- students' satisfaction with automatic judges.

In Figure 13 we report the responses of the students in the two groups to the question about the level to which they like programming on a scale from 1 ("not at all") to 5 ("a lot"). We have split the two groups also according to the degrees of the students, because it turns out that while on average students who used an automatic judge report to like programming a little more, this is not the case if we consider only the students in the degree of systems administration (ASIX). Whereas in the first group there is only a small difference between the students in the two different degrees, in the second group the distribution of answers is visibly bimodal. The same trend is visible in the way students evaluate their programming abilities, shown in Figure 14.

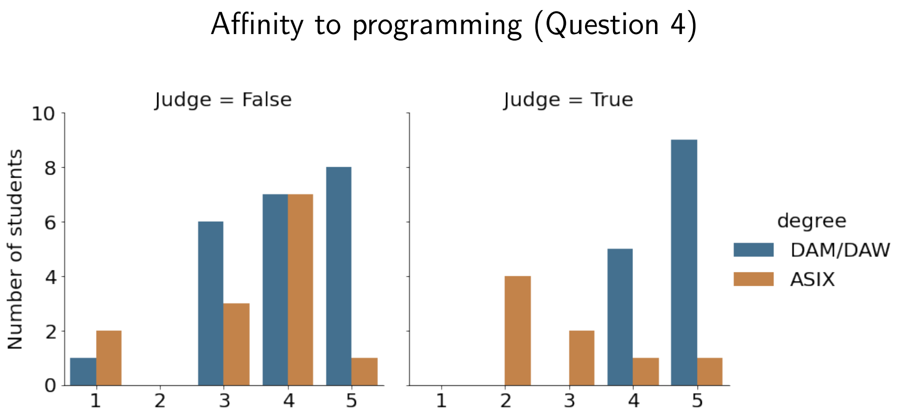


Figure 13: Histogram of the level to which students report to like programming, split by degree, for the group that didn't use a judge and the one that did.

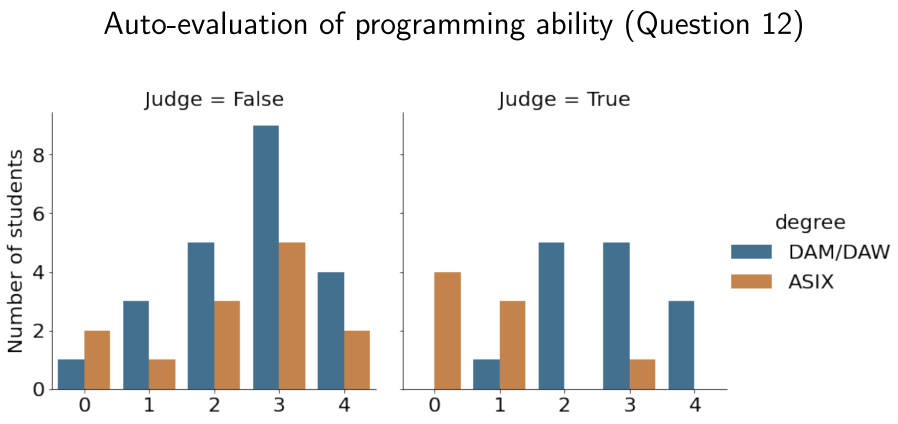
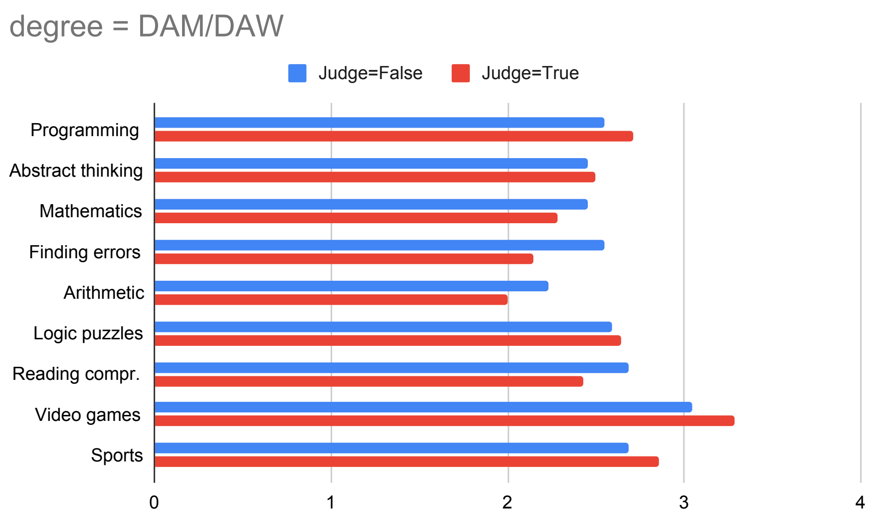


Figure 14: Histogram of the level to which students report to be good at programming, split by degree, for the group that didn't use a judge and the one that did.

We asked students to evaluate their abilities in 8 more areas. Other areas that are related to programming included mathematics, finding errors, abstract thinking, logic puzzles, and arithmetic. Skills that are only vaguely related to programming included videogames and reading comprehension. Finally, we also asked about sports skills, in order to account for general bias. In Figure 15 we show the averages for each of the nine skills. We split the groups by degree.

Among the students of the group of the degree of software engineering (DAM/DAW) there is very little difference in the self-evaluation of all skills. Perhaps the most important difference is in how they evaluate their ability to find errors. This could be attributed to their experience with failing the tests of automatic judges. On the other hand this experience has not resulted in lower evaluation of their programming skills.

Among the students of the group of the degree of systems administration (ASIX) it looks like



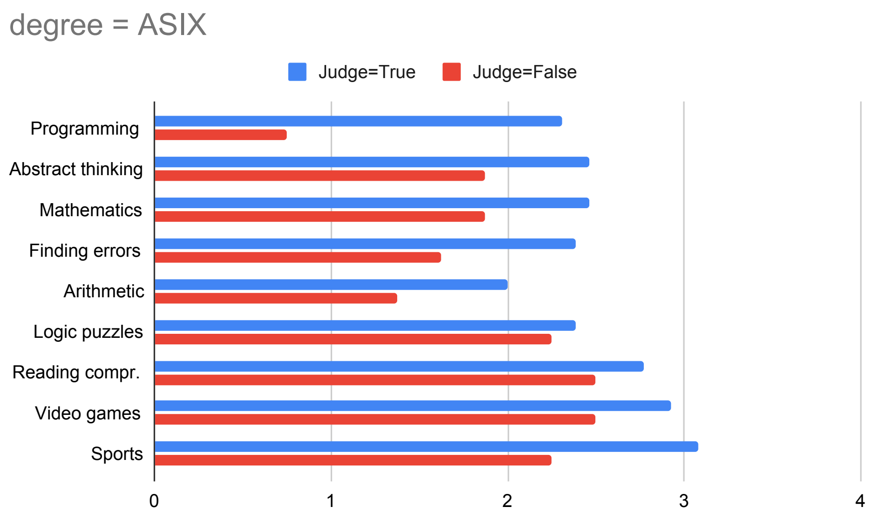


Figure 15: The average score that students give to their own skills.

those that use an automatic judge evaluate their abilities much lower in general, including in sports and videogames. We do not have an explanation for this phenomenon. The programming skills are especially evaluated much lower than all other skills, whereas this is not true for the group of students who don't use an automatic judge.

The same trend as the one in Figures 13 is noticeable in the answers to the questions about competition and teamwork (Questions 13 and 14). In both of these aspects the group that has used a judge reports higher values for the degrees of software development, and lower values for the degree of systems administration. The graphs are shown in Figures 16 and 17.

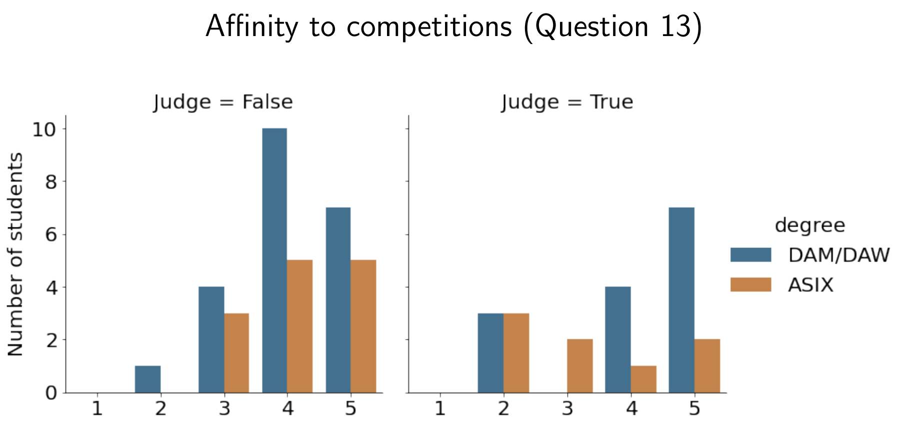


Figure 16: A histogram of the level to which students report to like competing, displayed by degree, for the group that didn't use a judge and the one that did.

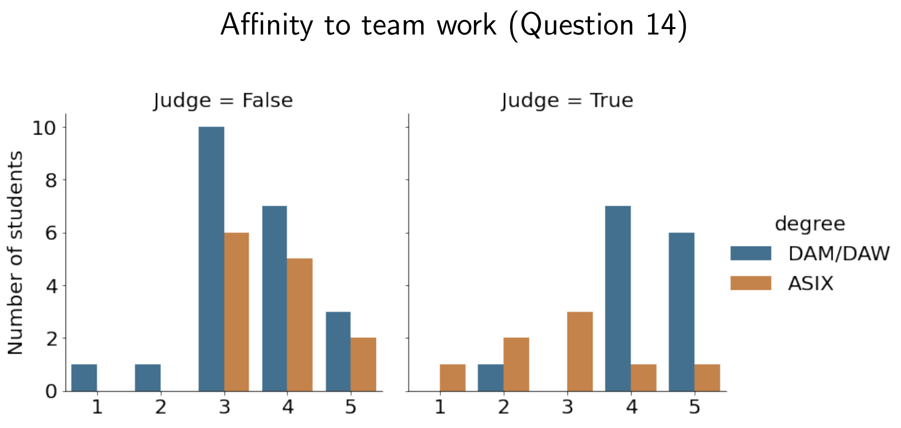


Figure 17: A histogram of the level to which students report to like to work in a team, displayed by degree, for the group that didn't use a judge and the one that did.

There is a similar trend, but not exactly the same, for the amount of time spent programming at home (Question 6). The effect is again different for the students of different degrees - it is much more positive for the students of the degrees of software development, but in this case, it is not negative for the students of the degree of systems administration. The histograms of the responses are shown in Figures 18.

Answers to Question 6 about time spent programming at home.

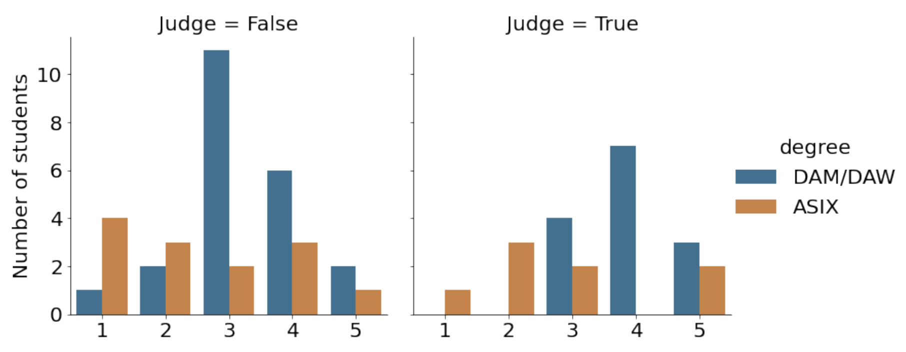


Figure 18: A histogram of student's answers to Question 6: "How many hours per week do you spend programming at home per week?" The options were (1) "0", (2) "More than 0, but less than 1", (3) "Between 1 and 2", (4) "Between 2 and 4", (5) "More than 4 hours".

We did not find any significant differences in the distribution of the answers to Questions 7 and 8 about the number of programs written and the amount of feedback received during the year. Histograms of the answers are shown in Figures 19 and 20.

Number of programs written (Question 7)

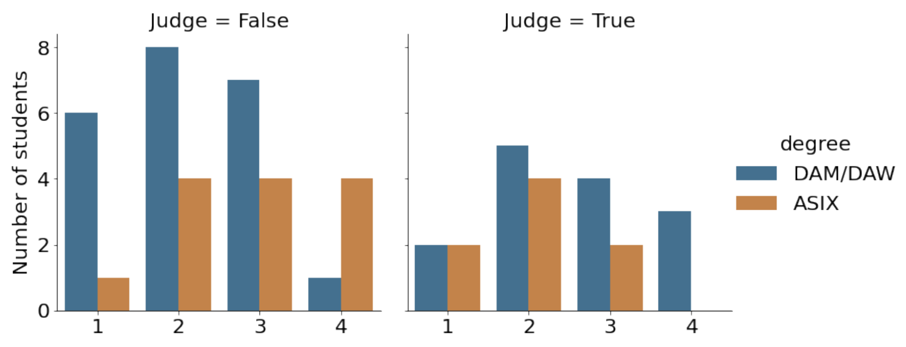


Figure 19: A histogram of the answers to Question 7: "How many programs have you written this academic year?" The options were (1) "Less than 20", (2) "Between 20 and 40", (3) "Between 40 and 60", (4) "More than 60".

Finally, one of the most important questions is how do students feel about the use of automatic judges. Their answers were quite positive as can be seen on Figure 21. Even the students from the degree of systems administration, who give a very low grade to their own programming skills, consider that the judge is useful for learning to program.

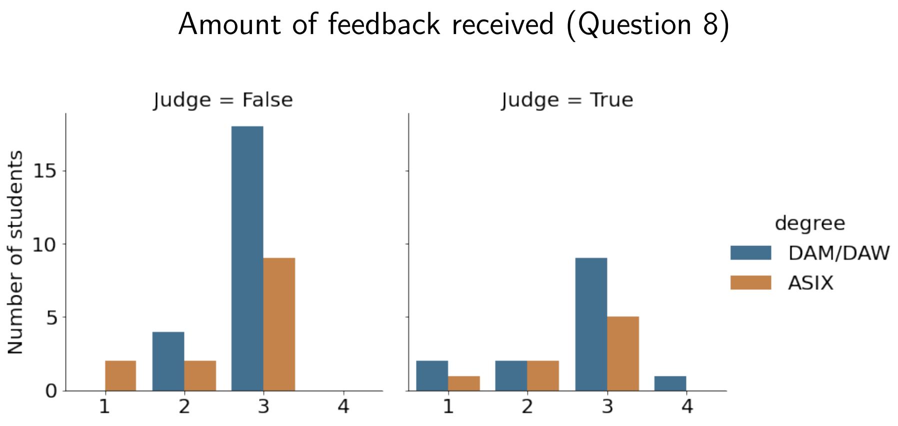


Figure 20: A histogram of the answers to Question 8: “Do you think you have received enough feedback?” The options were (1)”Very insufficient”, (2) “Some, but not enough” (3) “The correct amount” (4) “Too much feedback”.

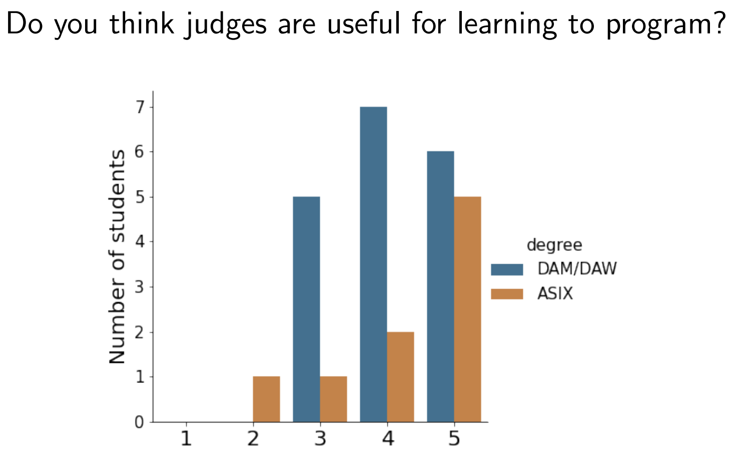


Figure 21: Answers to Question 11 about the usefulness of automatic judges for learning how to program. The scale is from (1) “Not at all” to (5) “Very useful”.

8. Conclusions

Our results deviate from those of [1] that were described in Table 1. We summarise our answers to the same four questions in Table 10. Based on the results of the three Bebras challenges evaluating computational thinking skills, we do not observe any improvement in students' performance. With respect to the students' impression on whether automatic judges have improved their performance, the data we collected would correspond to a definite “Yes”, based on the answers summarised in Figure 21. With respect to the instructor's impressions of whether these tools have improved their teaching experience, we can conclude that the answer is “Yes”, based on the answers summarised in Figure 8.

Question	Conclusion of [1]	Conclusion of this work
Have AATs proven to be helpful in improving student learning?	Yes	No
Do students think that AATs have improved their performance?	Inconclusive	Yes
After having used the tools, do instructors think that the tools have improved their teaching experiences?	Yes	Yes
Is the assessment performed by AATs accurate enough to be helpful?	Yes	Inconclusive

Table 10: Table of conclusions of this study compared to those of [1].

Finally, the question about whether the assessment performed by AATs is accurate enough to be helpful, is arguably too vague to have a clear answer. We consider our results "Inconclusive" in this respect, because one of the issues identified as a reason *against* using automatic judges was that programs that pass the judge can be deficient anyway. The majority of teachers think that the lack of feedback with respect to the style of the programs is a valid reason (rated 3 out of 5 on average) *against* using judges as the only method of evaluation. We base this on the answers to Question 35, summarised in Table 4.

The reason we consider the question too vague is that the level to which the automatic assessment is helpful depends on how it is used. While it's true that teachers complain about the lack of qualitative feedback, at the same time they indicate that the assessment is useful enough to recommend its use to other teachers (as indicated in Figure 8). Perhaps the use of judges frees teachers from mechanical assessment so that they can give more detailed qualitative help themselves.

On a related note, we did not observe significant differences between the groups of students who used a judge and those who didn't in their impressions about the quality of feedback they received (Figure 20). This suggests that the most sceptical comments we got about teachers becoming worse instructors when they use judges, seem to be misguided.

On the other hand, we did observe that students who hadn't used judges were more likely to work on the Bebras challenges we proposed (Figure 12). This raises an interesting question: does the use of judges make students less confident? Our data in Figure 15 seems to indicate that this is the case for the group of students enrolled in the degree of systems administration (ASIX), but it could also be a result of some other characteristic of their environment or the school that they attend.

The fact that we observe very different effects on the attitude of students from different degrees means that perhaps automatic judges are better suited to the setting of the degrees of software development (DAM/DAW). It is still an interesting question for future study to identify the specific reasons for this difference. It may have to do with the number of hours of class time, the previous exposure to programming, or some other factor that we have not thought of.

In any case, a more complete and extensive study is required in order to answer these questions and the questions in Table 10 with better confidence. Ideally, for such a study one should have a larger sample of teachers, and that sample should be less biassed. For the student survey, too, a larger sample should be used, the conditions of taking the test should be controlled, as we commented in Section 7.1, and a questionnaire should be filled both before and after the first year of programming to compare the relative improvement. Our experience shows that Bebras challenges are simple enough to complete in a short time, and at the same time complicated enough to not be considered trivial by the majority of vocational training students. We recommend their use for future studies.

Another conclusion that we can draw from our study, which is not related to the use of automatic assessment, is that students in vocational training programs generally demonstrate weak performance on computational thinking questions. This suggests that we should also put some effort in teaching computational thinking skills at earlier stages of education, in order to better prepare students for learning programming. If we do this in secondary school we would have much better prepared students in our vocational training programs. A global competition of computational-thinking skills such as Bebras for schools (similar to the Mathematical Kangaroo, in which this year more than $105,000$ students participated only in Catalonia) would be an excellent source of information for educational-policy decision makers.

Finally, with respect to the question of why automatic judges are not getting adopted more rapidly in Vocational Training programs, the data from our surveys indicates that teachers are not familiar enough with the automatic judges that are available. An effort should be made to have this information available easily to all teachers.

References

[1] Raymond Scott Pettit, John D Homer, Kayla Michelle McMurry, Nevan Simone, and Susan A Mengel. Are automated assessment tools helpful in programming courses? In *2015 ASEE Annual Conference & Exposition*, pages 26–230, 2015.

[2] Kirsti M Ala-Mutka. A survey of automated assessment approaches for programming assignments. *Computer science education*, 15(2):83–102, 2005.

[3] Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, and Otto Seppälä. Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th International Conference on Computing Education Research*, pages 86–93, 2010.

[4] Jordi Petit, Omer Giménez, and Salvador Roura. Jutge. org: an educational programming judge. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 445–450, 2012.

[5] Codeforces. `https://codeforces.com/contest/405/problem/A`. Accessed: 2022-05-30.

[6] Jutge.org. `https://jutge.org`. Accessed: 2022-05-30.

[7] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. Introduction to automata theory, languages, and computation. *ACM SIGACT News*, 32(1):60–65, 2001.

[8] Henry Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical society*, 74(2):358–366, 1953.

[9] George E Forsythe and Niklaus Wirth. Automatic grading programs. *Communications of the ACM*, 8(5):275–278, 1965.

[10] International Collegiate Programming Contest. `https://icpc.global`. Accessed: 2022-06-07.

[11] International Olympiad in Informatics. `https://informatics.org`. Accessed: 2022-06-07.

[12] Miguel A Revilla, Shahriar Manzoor, and Rujia Liu. Competitive learning in informatics: The UVa online judge experience. *Olympiads in Informatics*, 2(10):131–148, 2008.

[13] Brenda Cheang, Andy Kurnia, Andrew Lim, and Wee-Chong Oon. On automated grading of programming assignments in an academic institution. *Computers & Education*, 41(2):121–131, 2003.

[14] José Paulo Leal and Fernando Silva. Mooshak: A web-based multi-site programming contest system. *Software: Practice and Experience*, 33(6):567–581, 2003.

[15] Jordi Petit, Salvador Roura, Josep Carmona, Jordi Cortadella, Jordi Duch, Omer Gimnez, Anaga Mani, Jan Mas, Enric Rodrguez-Carbonell, Enric Rubio, et al. Jutge. org: Characteristics and experiences. *IEEE Transactions on Learning Technologies*, 11(3):321–333, 2017.

[16] Maria J Blesa, Amalia Duch, Joaquim Gabarró, Jordi Petit, and Maria Serna. Continuous assessment in the evolution of a CS1 course: The pass rate/workload ratio. In *International Conference on Computer Supported Education*, pages 313–332. Springer, 2015.

[17] Programador Clic. Programador Clic: Sistema de jueces en línea (OJ para abreviar). `https://programmerclick.com/article/37451890668`. Accessed: 2022-06-05.

[18] Gines Garcia-Mateos and Jose Luis Fernandez-Aleman. Make learning fun with programming contests. In *Transactions on edutainment II*, pages 246–257. Springer, 2009.

[19] Pedro Pablo Gómez Martín and Marco Antonio Gómez Martín. ¡ Acepta el reto!: juez online para docencia en español. In *TICAI 2017: TICs para el Aprendizaje de la Ingeniería*, pages 45–52. Universidade de Vigo, 2018.

[20] Marco Antonio Gómez Martín. Desarrollo de material educativo reutilizable para el portal educativo de la UCM "Acepta el reto", 2016.

[21] Jean Luca Bez, Neilor A Tonin, and Paulo R Rodegheri. URI Online Judge Academic: A tool for algorithms and programming classes. In *2014 9th International Conference on Computer Science & Education*, pages 149–152. IEEE, 2014.

[22] Szymon Wasik, Maciej Antczak, Jan Badura, Artur Laskowski, and Tomasz Sternal. A survey on online judge systems and their applications. *ACM Computing Surveys (CSUR)*, 51(1):1–34, 2018.

[23] UVa Online Judge. `https://onlinejudge.org`. Accessed: 2022-06-07.

[24] Topcoder. `https://www.topcoder.com`. Accessed: 2022-06-07.

[25] Exercism. `https://exercism.org`. Accessed: 2022-06-07.

[26] Codewars. `https://www.codewars.com`. Accessed: 2022-06-07.

[27] RACSO. `https://racso.cs.upc.edu`. Accessed: 2022-06-07.

[28] HackerRank. `https://www.hackerrank.com`. Accessed: 2022-06-07.

[29] LeetCode. `https://leetcode.com`. Accessed: 2022-06-07.

[30] DOMjudge. `https://domjudge.org`. Accessed: 2022-06-07.

[31] JOEL. `https://jo-el.es`. Accessed: 2022-06-07.

[32] ¡Acepta el Reto! `https://www.aceptaelreto.com`. Accessed: 2022-06-07.

[33] José Carlos Paiva, José Paulo Leal, and Ricardo Queirós. Fostering programming practice through games. *Information*, 11(11):498, 2020.

[34] Yutaka Watanobe, Chowdhury Intisar, Ruth Cortez, and Alexander Vazhenin. Next-generation programming learning platform: Architecture and challenges. In *SHS Web of Conferences*, volume 77, page 01004. EDP Sciences, 2020.

[35] Katerina Georgouli, Ilias Skalkidis, and Pedro Guerreiro. A framework for adopting LMS to introduce e-learning in a traditional course. *Journal of Educational Technology & Society*, 11(2):227–240, 2008.

[36] Katerina Georgouli and Pedro Guerreiro. Incorporating an automatic judge into blended learning programming activities. In *International Conference on Web-Based Learning*, pages 81–90. Springer, 2010.

[37] Georgiana Haldeman, Monica Babeş-Vroman, Andrew Tjang, and Thu D Nguyen. CSF: Formative feedback in autograding. *ACM Transactions on Computing Education (TOCE)*, 21(3):1–30, 2021.

[38] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education (TOCE)*, 19(1):1–43, 2018.

[39] !Acepta el Reto! (hints). `https://www.aceptaelreto.com/doc/hints.php`. Accessed: 2022-06-07.

[40] Marco Antonio Gómez Martín, Pedro Pablo Gómez Martín, José Alberto Verdejo López, María Isabel Pita Andreu, Clara María Segura Díaz, Gómez Albarrán, Mª De Las Mercedes, Jénnifer Hernández Bécares, Jesús Javier Domenech Arellano, Luís María Costero Valero, et al. Añadiendo mecanismos de ayuda en un juez on-line automático para soporte a mentorías académicas. *Ene*, 17:08, 2019.

[41] Anaga Mani, Divya Venkataramani, Jordi Petit Silvestre, and Salvador Roura Ferret. Better feedback for educational online judges. In *Proceedings of the 6th International Conference on Computer Supported Education, Volume 2: Barcelona, Spain, 1-3 April, 2014*, pages 176–183. SciTePress, 2014.

[42] Paolo Fantozzi and Luigi Laura. Recommending tasks in online judges using autoencoder neural networks. *Olympiads in Informatics*, 14:61–76, 2020.

[43] Guillermo Jimenez-Diaz, Pedro Pablo Gómez-Martín, Marco Antonio Gómez-Martín, and Antonio A Sánchez-Ruiz. Similarity metrics from social network analysis for content recommender systems. *AI Communications*, 30(3-4):223–234, 2017.

[44] Antonio A Sánchez-Ruiz, Guillermo Jimenez-Diaz, Pedro P Gómez-Martín, and Marco A Gómez-Martín. Case-based recommendation for online judges using learning itineraries. In *International Conference on Case-Based Reasoning*, pages 315–329. Springer, 2017.

[45] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy,

Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode. https://arxiv.org/abs/2203.07814, 2022.

[46] Stefania Bocconi, Augusto Chioccariello, Giuliana Dettori, Anusca Ferrari, Katja Engelhardt, Panagiotis Kampylis, and Yves Punie. Developing computational thinking in compulsory education. *European Commission, JRC Science for Policy Report*, 68, 2016.

[47] ProgramaMe. `https://programame.com`. Accessed: 2022-06-07.

[48] Catskills. `http://catskills.gencat.cat`. Accessed: 2022-06-07.

[49] Worldskills. `https://worldskills.org`. Accessed: 2022-06-07.

[50] Aprende a programar. `https://aprendeaprogramar.org`. Accessed: 2022-06-07.

[51] CodeChef. `https://codechef.com`. Accessed: 2022-06-07.

[52] beecrowd. `https://www.beecrowd.com.br`. Accessed: 2022-06-07.

[53] CodedArena. `https://codedarena.com`. Accessed: 2022-06-03.

[54] Leagues of Code. `https://leaguesofcode.com`. Accessed: 2022-06-03.

[55] Codelearn. `https://codelearn.es`. Accessed: 2022-06-03.

[56] Bebras. `https://www.bebras.org`. Accessed: 2022-06-07.

[57] Mathematical Kangaroo. `https://aksforg`. Accessed: 2022-06-07.

[58] Meike Akveld, Luis F Caceres-Duque, and Robert Geretschläger. Math Kangaroo. *Mathematics Competitions*, 33(2), 2020.

[59] Valentina Dagienė and Gerald Futschek. Bebras international contest on informatics and computer literacy: Criteria for good tasks. In *International Conference on Informatics in Secondary Schools: Evolution and Perspectives*, pages 19–30. Springer, 2008.

[60] Ana Liz Souto O Araujo, Wilkerson L Andrade, Dalton D Serey Guerrero, and Monilly Ramos Araujo Melo. How many abilities can we measure in computational thinking? A study on Bebras challenge. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 545–551, 2019.

[61] Annalisa Calcagni, Violetta Lonati, Dario Malchiodi, Mattia Monga, and Anna Morpurgo. Promoting computational thinking skills: would you use this Bebras task? In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pages 102–113. Springer, 2017.

[62] Victor Koleszar, Daiana Clavijo, Emiliano Pereiro, and Alar Urruticoechea. Análisis preliminares de los resultados del desafío Bebras 2020 en Uruguay. *Revista INFAD de Psicología. International Journal of Developmental and Educational Psychology.*, 1(2):17–24, 2021.

[63] Francisco Bavera, Marcela Daniele, Teresa Quintero, and Flavia Buffarini. Habilidades de pensamiento computacional en docentes de primaria: evaluación usando Bebras. In *XXV Congreso Argentino de Ciencias de la Computación (CACIC)(Universidad Nacional de Río Cuarto, Córdoba, 14 al 18 de octubre de 2019)*, 2019.

A. Questionnaire for teachers

In the following pages we include the questionnaire prepared using *Google Forms* that was sent to teachers. It is filled out anonymously online and does not require log in. The first question determines whether the questionnaire will be done in Catalan or Spanish. Questions 2-39 are in Catalan and Questions 40-77 are the same ones posed in Spanish.

There is another branching in the questionnaire depending on whether the teacher has used automated judges or not. In addition, some questions have a single follow up clarification question. In Figure 22 we show the number of questions along each path. In parentheses is the number of follow-up clarification questions. The last three questions ask for personal information which is optional, and will be used only for communicating the final work to the respondent.

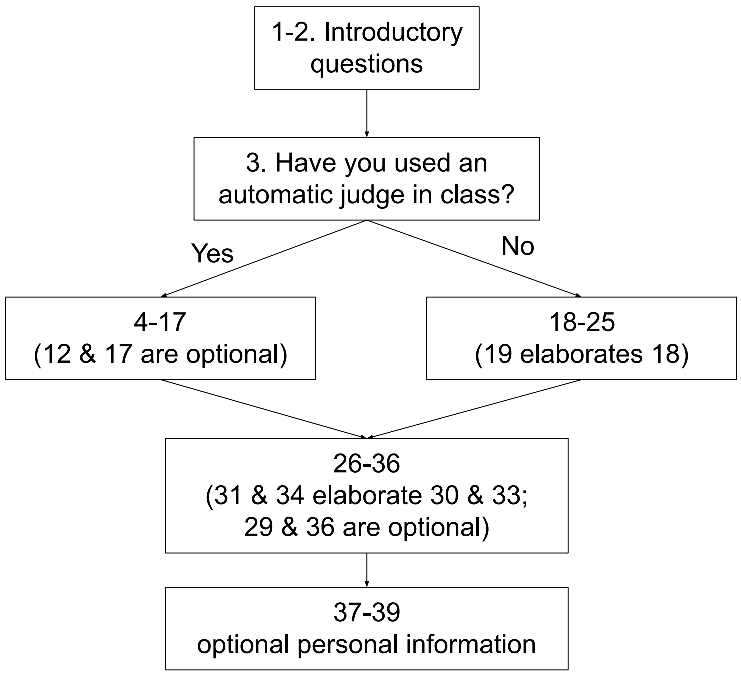

1-2. Introductory questions
3. Have you used an automatic judge in class?
Yes
No
4-17 (12 & 17 are optional)
18-25 (19 elaborates 18)
26-36 (31 & 34 elaborate 30 & 33; 29 & 36 are optional)
37-39 optional personal information

Figure 22: Structure of the questionnaire.

Ús de jutges automàtics a la FP en mòduls de programació / Uso de jueces automáticos en la FP en módulos de programación

* Required

1. Idioma *

Mark only one oval.

- ⬭ Català *Skip to question 2*
- ⬭ Castellano *Skip to question 40*

Ús de jutges automàtics a la FP en mòduls de programació

Aquest qüestionari està adreçat a professors de Formació Professional en la família d'Informàtica i Comunicació que han impartit algun cop el Mòdul 3 (Programació). L'objectiu és avaluar el grau d'ús dels jutges automàtics en l'ensenyament de programació i els seus pros i contres segons els docents.

En aquest treball fem servir el nom "jutge automàtic" per a qualsevol servei en línia amb repositori de problemes de programació que proporciona avaluació automàtica d'entregues dels usuaris mitjançant jocs de proves predeterminats.

2. Titularitat de la institució educativa on imparteixes classes *

Mark only one oval.

- ⬭ Pública
- ⬭ Privada
- ⬭ Privada Concertada

3. Has fet servir algun cop algun jutge automàtic a les classes de programació a la FP? *

Mark only one oval.

- ⬭ Sí *Skip to question 4*
- ⬭ No *Skip to question 18*

Preguntes sobre experiències amb jutges automàtics

4. Quins jutges has fet servir a classes de FP? *

Check all that apply.

- ☐ ¡Acepta el reto!
- ☐ Jutge.org
- ☐ JOEL (jo-el.es)
- ☐ aprendeaprogramar.org
- ☐ UVa Online Judge
- ☐ HackerRank
- ☐ LeetCode
- ☐ CodeChef
- ☐ Codeforces
- ☐ Topcoder
- ☐ Other: ____________________

5. En quins cicles els has fet servir? *

Check all that apply.

- ☐ SMX - Sistemes microinformàtics i xarxes
- ☐ ASIX - Administració de sistemes informàtics en xarxa
- ☐ DAM/DAW - Desenvolupament d'aplicacions multiplataforma/web
- ☐ Other: ____________________

6. Durant quants cursos has fet servir jutges automàtics? *

Mark only one oval.

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

7. En quins tipus d'UFs del mòdul de programació has fet servir jutges automàtics? *

Check all that apply.

- ☐ UF 1 - Programació estructurada
- ☐ UF 2 - Disseny modular
- ☐ UF 3 - Fonaments de gestió de fitxers
- ☐ UF 4 - Programació orientada a objectes (POO). Fonaments.
- ☐ UF 5 - POO. Llibreries de classes fonamentals.
- ☐ UF 6 - POO. Introducció a la persistència en BD.
- ☐ Other: ____________________

8. Fins a quin punt trobes que els jutges són útils per a la teva tasca docent? *

Mark only one oval.

- ○ Prefereixo no haver de fer-los servir.
- ○ Sóc indiferent.
- ○ Em simplifiquen la tasca docent.
- ○ Seria difícil ensenyar programació sense fer-los servir.
- ○ Els trobo imprescindibles.

9. Fins a quin punt trobes que els jutges són importants per a l'aprenentatge dels alumnes en general? *

Mark only one oval.

- Aprendrien millor sense els jutges automàtics.
- No tenen efecte significatiu sobre l'aprenentatge.
- Aprenen una mica més utilitzant jutges automàtics.
- Aprenen molt més utilitzant jutges automàtics.

10. Trobes que els jutges són més beneficiosos per als alumnes amb més o menys rendiment acadèmic? *

Mark only one oval.

- El benefici és més grans per als alumnes amb menys rendiment.
- És igual de beneficiós per a tots els alumnes.
- El benefici és més gran per als alumnes amb més rendiment.

11. Valora les raons per fer servir un jutge automàtic: de 1 (gens vàlida) a 5 (molt vàlida). *

Mark only one oval per row.

	1 (Gens vàlida)	2	3	4	5 (Molt vàlida)
Suposen un estalvi important de temps pel docent	◯	◯	◯	◯	◯
Els repositoris de problemes són una font de bons exercicis	◯	◯	◯	◯	◯
Creen sentit de comunitat entre els diferents instituts	◯	◯	◯	◯	◯
Ensenyen seguir rigorosament les especificacions tècniques	◯	◯	◯	◯	◯
La retroalimentació és instantània	◯	◯	◯	◯	◯
Milloren la comprensió lectora	◯	◯	◯	◯	◯
El format de reptes motiva els alumnes	◯	◯	◯	◯	◯

12. Si se t'ha acudit una altra raó, quina és?

____

13. Recomanaries l'ús de jutges a altres professors o instituts? *

Mark only one oval.

	1	2	3	4	5	
No	⬭	⬭	⬭	⬭	⬭	Sí

14. Com de difícil creus que és adoptar un jutge automàtic per a professors novells? *

Mark only one oval.

	1	2	3	4	5	
Molt fàcil	⬭	⬭	⬭	⬭	⬭	Molt difícil

15. Com de difícil creus que és adoptar un jutge automàtic per a professors experimentats? *

Mark only one oval.

	1	2	3	4	5	
Molt fàcil	⬭	⬭	⬭	⬭	⬭	Molt difícil

16. Quina importància dones a aquestes funcionalitats d'un jutge automàtic? *

Mark only one oval per row.

	Gens important	Poc important	Molt important	Imprescindible
Que tingui un gran repositori de problemes	⬭	⬭	⬭	⬭
Que es puguin afegir nous problemes	⬭	⬭	⬭	⬭
Que el professor pugui veure quin joc de prova li ha fallat a l'alumne	⬭	⬭	⬭	⬭
Que el professor pugui visualitzar les entregues dels alumnes	⬭	⬭	⬭	⬭
Que les notes vagin automàticament al Moodle	⬭	⬭	⬭	⬭
Que el professor pugui veure el progrés de tota la classe facilment	⬭	⬭	⬭	⬭
Que l'alumne pugui rebre pistes automàticament quan el seu programa està denegat	⬭	⬭	⬭	⬭
Que el professor pugui afegir retroalimentació a les entregues	⬭	⬭	⬭	⬭
Que el professor pugui crear	⬭	⬭	⬭	⬭

tasques amb data d'inici i d'acabament				
Que l'alumne pugui visualitzar entregues dels altres alumnes després d'haver superat les proves	⬭	⬭	⬭	⬭

17. Si se't acudeix alguna funcionalitat més que trobis important o imprescindible, quina és?

____

____

____

____

____

Skip to question 26

Actituds envers els jutges automàtics

18. Has considerat fer servir un jutge automàtic en les teves classes? *

Mark only one oval.

⬭ Sí *Skip to question 19*

⬭ No *Skip to question 20*

Raons per haver descartat l'ús d'un jutge automàtic

19. Quina és la raó principal per haver descartat l'ús de jutges? *

Mark only one oval.

- Els meus companys d'institut hi estaven en contra.
- Els estudiants hi estaven en contra.
- No vaig tenir temps per investigar les opcions.
- Després d'investigar les opcions vaig decidir que no n'hi havia cap de suficientment bona.
- Vaig escollir-ne un però no vaig tenir temps per adoptar-lo o no va funcionar prou bé.
- Other: ____________________

Experiència

20. En quins cicles has donat classes de programació? *

Check all that apply.

- SMX - Sistemes microinformàtics i xarxes
- ASIX - Administració de sistemes informàtics en xarxa
- DAM/DAW - Desenvolupament d'aplicacions multiplataforma/web
- Other: ____________________

21. Quins d'aquests jutges automàtics coneixes?

Check all that apply.

- ☐ Jutge.org (de Universitat Politècnica de Catalunya)
- ☐ Acepta el Reto (de Universidad Complutense de Madrid)
- ☐ JOEL (de IES Sabadell)
- ☐ UVa Online Judge (Universidad de Valladolid)
- ☐ aprendeaprogramar.com (de IES Serra Perrenxisa)
- ☐ HackerRank
- ☐ LeetCode
- ☐ CodeChef
- ☐ Codeforces
- ☐ Topcoder
- ☐ Other: ____________________

22. Creus que una eina d'avaluació automàtica t'estalviaria o afegiria temps? *

Mark only one oval.

	1	2	3	4	5	
M'afegiria bastant temps	◯	◯	◯	◯	◯	M'estalviaria bastant temps

23. Creus que una eina d'avaluació automàtica milloraria o empitjoraria l'aprenentatge dels teus alumnes en general? *

Mark only one oval.

	1	2	3	4	5	
Empitjoraria molt	◯	◯	◯	◯	◯	Milloraria molt

24. Creus que una eina d'avaluació automàtica tindria més beneficis per als alumnes a menys o més rendiment acadèmic? *

Mark only one oval.

	1	2	3	
Per als de menys rendiment	◯	◯	◯	Per als de més rendiment

25. Creus que una eina d'avaluació automàtica seria desmotivadora o motivadora per als alumnes? *

Mark only one oval.

	1	2	3	4	5	
Molt desmotivadora	◯	◯	◯	◯	◯	Molt motivadora

Preguntes sobre retroalimentació

26. Durant quants cursos has impartit classes de programació a cicles de FP? *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	10 o més

27. Quin percentatge dels programes entregats pels alumnes (exclosos els exàmens, però incloses totes les pràctiques) corregeixes amb retroalimentació? *

Mark only one oval.

	0	1	2	3	4	5	6	7	8	9	10	
0%	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	100%

28. Quan dones retroalimentació, de quantes paraules a la mitjana és? *

Mark only one oval.

◯ De 5 o menys.
◯ Entre 5 i 10.
◯ Entre 10 i 20.
◯ Més de 20.

29. Si vols explicar alguna experiència que va en contra o a favor dels jutges, o vols compartir comentaris sobre l'adopció dels jutges automàtics, fes-ho aquí:

30. Han participat alumnes teus a ProgramaMe? *

Mark only one oval.

◯ Sí
◯ No *Skip to question 32*

Participació en ProgramaMe

31. Durant quants cursos han participat alumnes teus a ProgramaMe? *

Mark only one oval.

1	2	3	4	5	6	7	8	9	10
⬭	⬭	⬭	⬭	⬭	⬭	⬭	⬭	⬭	⬭

32. Al teu institut actualment s'ofereix el grau de DAM-Videojocs? *

Mark only one oval.

- ⬭ Sí
- ⬭ No

33. Fas servir ludificació (gamification) en les teves classes? (Leaderboards, treasure hunts, escape rooms, kahoot...) *

Mark only one oval.

- ⬭ Sí
- ⬭ No *Skip to question 35*

Ludificació (Gamification)

34. Quines d'aquestes tècniques de ludificació has fet servir? *

Check all that apply.

- ☐ Sistemes student response (Kahoot, Socrative, Mentimeter...)
- ☐ Leaderboards (classificació)
- ☐ Badges (insígnies)
- ☐ Puntuacions
- ☐ Treasure hunt / Break-out / Escape room físic
- ☐ Treasure hunt / Break-out / Escape room digital
- ☐ Other: ______________________

Les raons en contra dels jutges automàtics

35. Valora les raons per NO fer servir un jutge: de 1 (gens vàlida) a 5 (molt vàlida). *

Mark only one oval per row.

	1 (Gens vàlida)	2	3	4	5 (Molt vàlida)
Els problemes en els repositoris són massa difícils per a principiants	◯	◯	◯	◯	◯
Programes que passen el jutge poden ser molt deficients igualment	◯	◯	◯	◯	◯
La sortida ha de ser exactament en el format esperat	◯	◯	◯	◯	◯
El rebuig repetit de programes que no funcionen pot generar frustració	◯	◯	◯	◯	◯
Es fa fàcil que el professor no llegeixi mai els programes dels alumnes	◯	◯	◯	◯	◯
Els jutges automàtics generen competitivitat i van en contra de la cooperació entre alumnes	◯	◯	◯	◯	◯

36. Si se t'ha acudit una altra raó, quina és?

____

Gràcies!

Moltes gràcies per les teves respostes.

Les següents preguntes són totalment OPCIONALS. Si vols rebre el Treball Final de Màster i qualsevol altra publicació derivada d'aquest treball, pots deixar el teu contacte aquí o enviar correu a elitza.maneva@estudiantat.upc.edu. Aquestes dades no es faran servir en l'estudi.

37. Nom

___

38. Nom de l'Institut

___

39. Correu electrònic

___

Uso de jueces automáticos en la FP en módulos de programación

Este cuestionario está dirigido a profesores de formación profesional en la familia de Informática y Comunicación que han impartido alguna vez el Módulo 3 (Programación). El objetivo es evaluar el grado de uso de los jueces automáticos en la enseñanza de programación y sus pros y contras según los docentes.

En este trabajo utilizamos el nombre "juez automático" para cualquier servicio online con repositorio de problemas de programación que proporciona evaluación automática de entregas de los usuarios mediante juegos de pruebas predeterminados.

40. Titularidad de la institución educativa donde impartes clases *

Mark only one oval.

- ( ) Pública
- ( ) Privada
- ( ) Privada Concertada

B. Questionnaire for students

Actituds dels estudiants envers la programació

* Required

1. Quin grau estàs cursant? *

   Mark only one oval.

   ◯ SMX
   ◯ ASIX
   ◯ DAM/DAW

2. Has fet el mòdul de Programació durant aquest curs? *

   Mark only one oval.

   ◯ Sí
   ◯ No

Experiència

3. Havies estudiat programació abans d'aquest curs? *

   Mark only one oval.

   ◯ Sí
   ◯ No

4. T'agrada la programació? *

Mark only one oval.

	1	2	3	4	5	
Gens	⬭	⬭	⬭	⬭	⬭	M'encanta

5. Quantes hores de classe de Programació feu a la setmana? *

Mark only one oval.

1	2	3	4	5	6	7	8	9	10
⬭	⬭	⬭	⬭	⬭	⬭	⬭	⬭	⬭	⬭

6. Quantes hores dediques a la programació fora de classe a la setmana? *

Mark only one oval.

⬭ Cap

⬭ Més de 0 però menys d'1 hora

⬭ Entre 1 i 2 hores

⬭ Entre 2 i 4 hores

⬭ Més de 4 hores

7. Quants programes diferents has escrit durant aquest curs? *

Mark only one oval.

- Menys de 20
- Entre 20 i 40
- Entre 40 i 60
- Més de 60

8. Creus que has rebut suficient retroalimentació per als teus programes? *

Mark only one oval.

- Molt insuficient
- He rebut alguns comentaris però no suficients
- La quantitat correcte
- Massa retroalimentació

Jutges automàtics

Un jutge automàtic és una plataforma en línia, amb un repositori de problemes de programació, on pots enviar el teu codi i rebre resposta inmediata si el teu codi supera els jocs de proves o no. Alguns exemples de jutges automàtics són ¡Acepta el Reto!, JOEL, Gripau, Jutge.org, aprendeaprogramar.org, Hacker Rank, LeetCode etc.

9. Heu fet servir un jutge automàtic a la classe de programació? *

Mark only one oval.

- Sí, a cada classe
- Sí, de vegades
- Només un cop
- No *Skip to question 12*

10. Quins jutges has provat? *

Check all that apply.

- ☐ ¡Acepta el Reto!
- ☐ JOEL
- ☐ Jutge.org
- ☐ aprendeaprogramar.org
- ☐ HackerRank
- ☐ LeetCode
- ☐ Codeforces
- ☐ TopCoder
- ☐ Gripau
- ☐ Codewars
- ☐ Other: ____________________

11. Creus que els jutges automàtics són útils per aprendre a programar? *

Mark only one oval.

	1	2	3	4	5	
Gens	◯	◯	◯	◯	◯	Molt

Auto-avaluació

12. Fins a quin punt se't donen bé aquestes coses? *

Mark only one oval per row.

	Gens	Una mica	Prou	Bé	Molt
Comprensió lectora					
Mates					
Aritmètica					
Videojocs					
Puzles de lògica					
Trobar errors					
Programació					
Pensament abstracte					
Esports					

13. T'agrada competir? *

Mark only one oval.

	1	2	3	4	5	
Gens						Molt

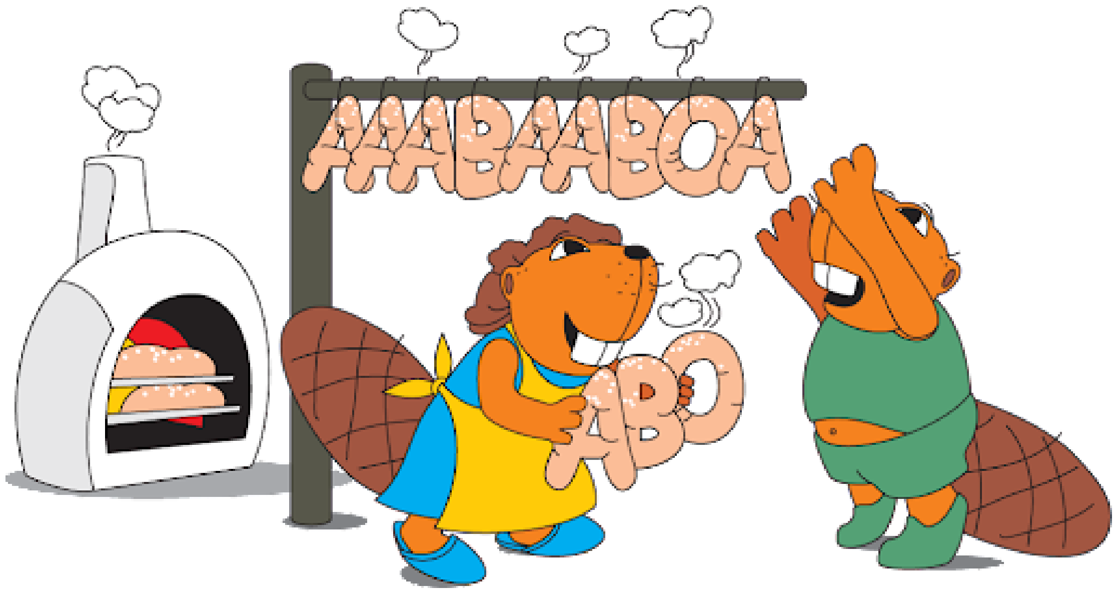
14. T'agrada treballar en equip? *

Mark only one oval.

	1	2	3	4	5	
Gens	⬭	⬭	⬭	⬭	⬭	Molt

Les tres preguntes a continuació són del concurs anual de Pensament Computacional "Bebras" (que vol dir "castor" en lituà). Si et semblen massa complicades pots parar i entregar el qüestionari en qualsevol moment.

15. Dos castors amics han obert un forn de pa. La Susanna forneja els panets de tres en tres, cadascun amb una forma diferent: A, B i O. Els penja d'un pal, junts, posant sempre primer l'A, després la B i finalment l'O. Fet això, torna a començar. En Pere es dedica a vendre els panets. Quan li en demanen un, sempre agafa el que està penjat més a la dreta. Aquest matí, la Susanna està fornejant els panets més ràpid que en Pere els ven. Si el forn té l'aspecte que veieu a la imatge, quín és el mínim nombre de panets que poden haver venut avui els castors?

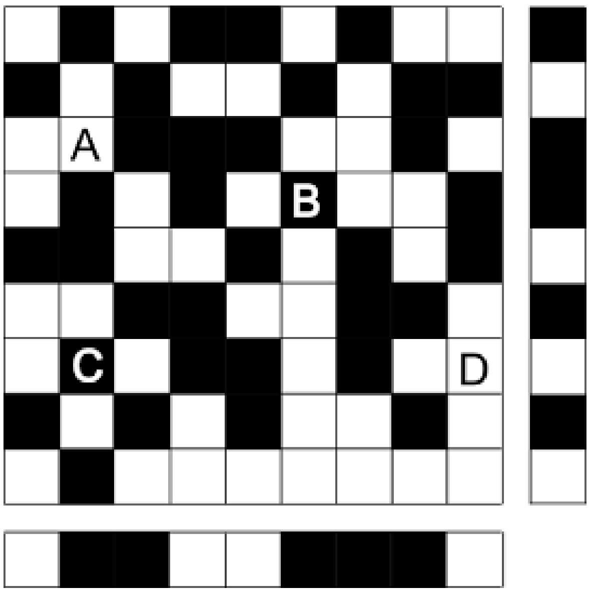


Mark only only one oval.

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12

- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20

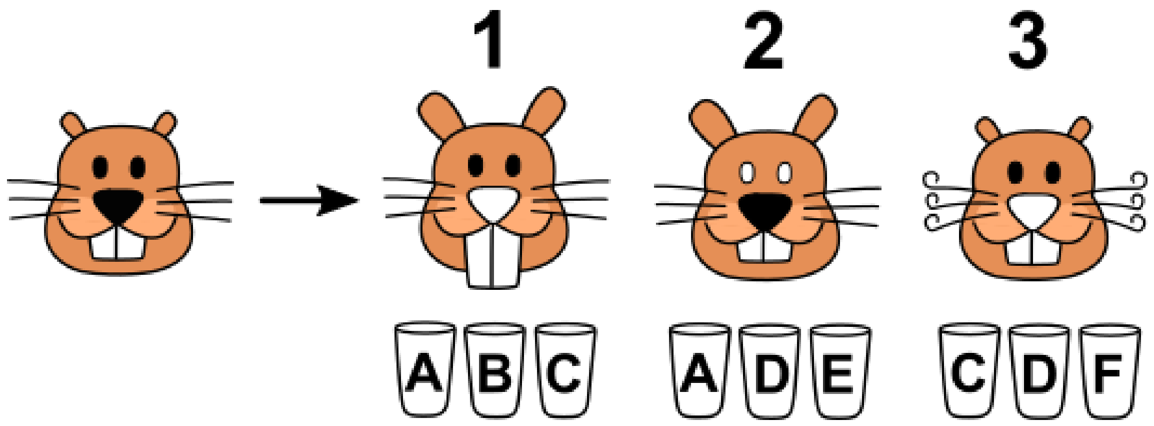
16. Un castor ha creat un dibuix amb quadres blancs i negres. Ara vol explicar per telèfon a un amic seu com fer el mateix dibuix. Per assegurar-se que no comet cap error, afegeix al seu dibuix una fila i una columna extres (a la dreta i a sota). Si el nombre de quadres negres en una fila és parell, afegeix un quadre negre al final de la fila. En cas contrari, el quadre extra serà blanc. Fa el mateix amb els quadrats negres de les columnes. Aquest és el dibuix que ha rebut l'altre castor per telèfon. Tots dos saben que hi ha hagut un error en copiar-lo. Pots descobrir on és l'error? Digues quin quadrat està malament:



Mark only only one oval.

- ◯ A
- ◯ B
- ◯ C
- ◯ D
- ◯ Cap

17. Una empresa ha inventat cinc nous tipus de medicaments que tenen els efectes següents sobre els castors: allargar les orelles, allargar les dents, arrissar els bigotis, fer el nas blanc, posar els ulls en blanc. No obstant, van oblidar posar etiquetes als gots que contenen els medicaments. A més, en un got s'ha posat només aigua, per la qual cosa hi ha sis gots, que han etiquetat de l'A a la F. Per esbrinar quin tipus de medicament són, es fan tres experiments: Experiment I: Si es pren el contingut dels gots A, B i C, el castor passa a tenir l'aspecte de la figura 1. Experiment II: Si es pren el contingut dels gots A, D i E, el castor passa a tenir l'aspecte de la figura 2. Experiment III: Si es pren el contingut dels gots C, D i F, el castor passa a tenir l'aspecte de la figura 3. Quin és el got amb aigua?



Mark only one oval.

- A
- B
- C
- D
- E
- F
- No es pot saber.

C. Qualitative data from the survey to teachers

Q.12. If you can think of another reason to use an automatic judge, what is it?		
ID	Original	Translation
36	Pot servir com a eina complementària per alumnes concrets	It can be used as a complementary tool for concrete students
37	Et permet que el la alumnes puguen avançar practicant tant com vullgues.	It allows the students to advance with practicing as much as they want.
35	Realització d'exercicis extra per aquells alumnes que van més avançats	The judge allows for additional exercises for those students who are more advanced
23	El alumnado puede practicar en casa para reforzar la clase	Students can practice at home in order to do better in class
33	com a eina de correcció tant per validar com per saber com avancen	as a correction tool not only for validating, but also for knowing how students are advancing
21	Permiten afinar mejor cuestiones como eficiencia de código o uso de memoria	They allow you to better fine-tune issues such as code efficiency or memory usage.
26	Us de bones pràctiques (us de git, tags, etc per a l'entrega) i control per part del professor de l'estat de cada alumne	Use of good practices (such as using git, tags, etc. for the submission) and control by the teacher of the state of each student
40	Aprender a usar DDT y Testing	Learning to use DDT and Testing

Table 11: Original answers to the free text Question 12 from the survey to teachers.

Q.36. If you can think of another reason *against* using automatic judges, what is it?		
ID	Original	Translation
8	Crearan professorat que no realitzarà correctament la seva funció de formador	The judges will create teachers who will not correctly carry out the function of instructors
33	Crec que l'alumne ha d'aprendre a validar els seus propis programes	I think the students have to learn to validate their own programs.
11	El juez no hace de guía y creo que es una labor importante para que el alumnado mejore.	The judge does not act as a guide and I think it is an important task in order for the students to improve.

Table 12: Original answers to the free text Question 36 from the survey to teachers.

Q.17. If you can think of another feature that you find important, what is it?		
ID	Original	Translation
21	La posibilidad de trabajar en diversos lenguajes para un mismo reto es algo que pocas plataformas ofrecen	The possibility to work in different programming languages on the same challenge is something that few platforms offer.
33	veure el codi	see the code
35	Realment, els jutges encara han de millorar molt, perquè avaluen el resultat de l'algorisme en si mateix, però no valoren la qualitat del codi. És aquest punt on els jutges juguen força males passades, perquè els alumnes poden agafar vicis molt dolents.	Really, the judges still have to get better, because they evaluate the result of the algorithm itself, but they do not value the quality of the code. It's this point where the judges play pretty bad tricks, because students can take on very bad vices."
40	Poder implementarlo en local y añadir otros lenguajes	Being able to implement it locally and add more languages

Table 13: Original answers to the free text Question 17 from the survey to teachers.

Q.29. If you would like to share any experience with automatic judges that goes in favor or against their use or comment about their adoption, do it here. (Part 1)		
ID	Original	Translation
3	Crec que els jutges són una eina complementària per saber si els programes funcionen o no, però la interacció del professor/a amb l'alumne/a per veure el funcionament dels programes aporta un element qualitatiu que no tenen els jutges.	I believe that the judges are a complementary tool to know if the programs work or not, but the interaction of the teacher with the student to see the functioning of the programs provides a qualitative element that the judges do not have.
7	Facilita la feina, pero despersonalitza i treu valor com a docent i com a motivació vers alumnat, risc d'acomodament dels docents i manca d'Innovació a mig termini	It facilitates the work, but depersonalises and reduces the value of the teacher and the motivation of the students, there is risc of layoffs for teachers and lack of medium-term innovation.
8	Poden treure feina al professorat sempre que la resolució sigui la que està prevista, però poden sortir solucions no previstes pel professorat potser millors i comptar com incorrectes. No ajuda l'automatització a la feina del professorat, feina que pot anar desapareixent per la substitució d'aquestes eines automàtiques. Molt millor posar més professorat per corregir i comentar que tancar els resultats a un pensament programat sense sentit comú.	They can save work for the teacher as long as the solution is the one that is expected, but solutions can come up that are not expected by the professor that may be better and count as wrong. It does not help to automate the work of the teacher, work that could disappear due to the substitution of these automatic tools. Much better get more teachers to grade and comment that it closes the results to a programmed thought without common sense.
17	La verdad es que me lo he planteado alguna vez pero creo que al final un juez comprueba que se cumple lo solicitado pero no comprueba si se ha utilizado un código eficiente, limpio, etc. que se comprueba viendo lo que hacen y dando feedback personalizado. Luego están los jueces como Acepta el reto que, si no me equivoco, están muy pensados a competiciones de informática que implican varios casos de prueba, etc. que personalmente me parecen un poco "rollo" y poco realistas. Me gusta mucho codewars y lo utilizo para sacar ideas pero nunca se lo he planteado a mi alumnado en clase. Quizá debería darle una oportunidad a alguno...	The truth is that I have considered it on occasion, but I think that in the end a judge checks that what was requested is fulfilled but does not check if an efficient, clean code, etc. has been used, which is verified by seeing what they do and giving personalised feedback. Then there are the judges like ¡Accept el Reto! that, if I'm not mistaken, are thought for computer competitions that involve several test cases, etc. which I personally find a bit of a drag and unrealistic. I really like Codewars and use it to come up with ideas, but I have never proposed it to my class. Maybe I should give one of those a try ...

Table 14: Original answers to the free text Question 29 from the survey to teachers.

Q.29. If you would like to share any experience with automatic judges that goes in favor or against their use or comment about their adoption, do it here. (Part 2)

ID	Original	Translation
15	Es una buena forma de mejorar el razonamiento algorítmico del alumnado, pero creo (sin tener mucha experiencia) que para los que llevan floja la programación les puede desmotivar	It is a good way to improve students' algorithmic reasoning, but I think (without much experience) that for those who are weak in programming it can be discouraging.
35	En contra, el que he dit: no valoren la qualitat de codi. Un altre punt en contra també és la redacció dels enunciats, ja que molts no s'entenen (ni jo, com a docent, els arribo a entendre). A part d'això, contenen moltes faltes d'ortografia. A favor: són una font molt bona d'enunciats i poden motivar a aquells alumnes més avançats a la classe.	Cons, what I said: they don't evaluate the quality of the code. Another point against is the editing of the statements, because many are not understandable (not even I, as a teacher, am able to understand them). Also, they contain many spelling mistakes. Pros: They are a very good source of statements and can motivate those more advanced students in the class.
31	L'experiència que conec d'aprop es el concurs Programame, dels creadors de aceptaelreto. També he format part de l'organització del concurs Programame als regionals catalans del 2015 i 2018 d'Olot.	The experience I know up close is the ProgramaMe contest, from the creators of *¡Acepta el Reto!*. I was also part of the organization of the ProgramaMe competition in the Catalan regionals of 2015 and 2018 in Olot.
33	Hem fabricat el nostre propi corrector, i tot i que falten hores de desenvolupament és útil	We have built our own corrector and although we lack hours of development it is useful
36	El jutge ha de permetre/facilitar una adopció flexible	The judge has to allow/ease a flexible adoption
40	No poder integrarlos en Moodle o existir soluciones que se pueden copiar en internet	Not being able to integrate them into Moodle and there are solutions that can be copied over the internet.

Table 15: Original answers to the free text Question 29 from the survey to teachers.