CZU: 004:378

CONTESTS GRADING SYSTEMS IN PROGRAMMING EDUCATION

Krassimir MANEV, Ph.D., Chief editor
https://orcid.org/0000-0002-3275-374X
Journal Mathematics and Informatics, Sofia, Bulgaria

Abstract. Programming contests are very popular form for attracting young people to the profession of program developers. Because of the nature of these contests for long years the evaluation of the contestants' works is performed with developed for the purpose Contest Grading Systems (GS) that spare time and eliminate evaluation mistakes. As the qualities of the contestants that are checked in a programming contest are the same with the qualities of the students that are prepared to be professional programmers are the same, GS could be successfully used in education in programming, too. In this paper we introduce GS – what they are, how they are used in evaluation of programming contest and how they could be used in education of programming. Different challenges that using of GS rise when they are used in such education are outlined and ways to cope with such system are discussed.

Key words: programming contests; evaluation of programs; contest grading systems; education in programming; programming training sites.

SISTEME DE EVALUARE A CONCURSURILOR ÎN EDUCAȚIA PRIVIND PROGRAMAREA

Abstract. Concursurile de programare sunt o formă foarte populară pentru atragerea tinerilor către profesia de dezvoltatori de programe. Datorită naturii acestor concursuri de ani îndelungați, evaluarea lucrărilor concurenților se realizează cu ajutorul Sistemelor de clasificare a Concurenților (GS) dezvoltate în acest scop, care economisesc timp și elimină greșelile de evaluare. Deoarece calitățile concurenților care sunt verificați într-un concurs de programare și calitățile studenților care sunt pregătiți să fie programatori profesioniști sunt aceleași, GS ar putea fi folosit cu succes și în formarea programatorilor. În această lucrare prezentăm GS – ce sunt acestea, cum sunt utilizate în evaluarea concursului de programare și cum ar putea fi utilizate în educația privind programarea. Sunt subliniate diferitele provocări cu care se confruntă în utilizarea GS atunci când sunt utilizate în astfel de educație și sunt discutate modalități de a face față unui astfel de sistem.

Cuvinte cheie: concursuri de programare; evaluarea programelor; sisteme de clasificare a concurenților; educație în programare; site-uri de instruire în programare.

1. Introduction

Using computers and computer programs in education is an inevitable trend nowadays in any domain but especially in the domains of Science, Technology, Engineering and Mathematics. Computer is the main instrument in education in Informatics and Information Technologies because the essence of these disciplines is the computer programming. But in education in programming computer could be not only an instrument and object of teaching programming. Computer applications could be used to help the process of education as well as in any other disciplines.

Nowadays competitions in programming (popular as programming Olympiads) actively use complex software system to optimize the organization of the events – so called *Contest*

Grading (or *Contest Management*) *Systems* – shortly *Grading Systems* (GS). Even if one GS has many components, its principle component is the *evaluating component*, dedicated to check the correctness and the efficiency of the solutions submitted by the contestants. That is why the GS could be extremely helpful in the education in programming. They could save enormous volume of teachers' work and to make in such a way the educational process more intensive and fruitful.

In this paper we would like to present shortly the history and functionalities of GS and to share our experience of using such systems for education in Programming and Algorithms.

2. What is a Grading System?

Programming contests consist of solving algorithmic tasks with computer programs. Contestants have to write the source code of the solution in one the programming languages (C and Pascal in the past, C/C++ and Java recently) proposed by the contest rules. Tasks are such that the programs have to read data from the standard input and write the result on the standard output (Forišek 2006). Reading input from a named text file and writing output in a named text file was asked before but is not applied today. The modern trend is that contestants have to write not a program but one or more functions with prescribed interface that are compiled and linked with a main function of the author, input data are passed to the contestant's function by reference and asked results are returned to the author's part. In such way contestants do not need to read data or to write results. This is saving time during the testing. Some negative aspects of this and other trends in programming contests are discussed in (Manev 2019).

Submitted to the system contestant's code is compiled, linked with the external modules, if any, and tested with a set of test cases. Three criteria are considered for accepting the submit – solving the test cases within some limit of time, with some limit of memory and, finally, writing the correct answers.

Three principle styles of assigning grading marks are used. In *ICPC style* (practiced in the contests for teams composed of university students) the submitted solution obtains one grading point if the program **passes successfully the three criteria for all test cases** and time passed from the beginning of the contest is registered. Teams are ranked by the amount of the obtained points and teams with equal number of points are ranked by the sun of elapsed times. In *IOI stile* (practiced in the individual contests for secondary school student) for **each successful test case contestant obtains some points** and ranking is by the sum of obtained point from all test cases of all tasks. Intermediate stile (practiced only in IOI style contests) is grouping test cases in a few groups and assigning points dedicated for the group if the program passes successfully all test of the group. Some GS maintain one of the styles, some – two, and some – all three styles.

By the described functionality it could seems that creating GS is not very difficult. But there are two crucial elements in the architecture of GS that need very high level of competence. First, GS has to keep in secret all test cases and not permit usage of some important facilities of the OS of the grading machines from the programs of contestants, because more of them are usually very experienced and could obtain not acceptable advantages. This is implemented by so called *sand box* – a specialized shield component of the system that protects test cases and access to macros of the OS. Second – the control on the used by the *program resources*, especially the elapsed by the program time, is extremely difficult on grading machines with multitasking OS – usually UNIX-like OS and is still an object of intensive research. Discussion on this and other difficulties in implementing GS you could see in (Mareš 2007; Manev et al. 2009; Tochev&Bogdanov 2010).

3. Short history

The first Grading System, probably, is ACM ICPC system PC^2 or PC^2 (PC2 Home page 2022). First version of PC^2 is issued in Sacramento State University in 1988 and used for evaluating local contests in ICPC style. In 1994 the version 4.1 was used for first time during the World Finals of ICPC. Since that PC^2 became the official GS of ICPC – not only for the World Finals but for all Regional Rounds of the contests too for long time. Due to its architecture – a server part and few client parts – it was relatively difficult to maintain and was step by step replaced by modern Web-based GS. The system is still in use and the current version pc2v9 could be found trough the proposed above reference. Bulgarian Web-based Grading System spoj0 for ICPC style grading was created by our student (Sredkov 2006) and was used long years for preparation of the team of Sofia University and in organizing Bulgarian Collegiate Programming Contest.

The first system for grading contests in IOI style was used during IOI'1999, held in Antalia, Turkey. It was implemented as an UNIX Shell script and was very primitive. Nevertheless, grading of the work of 253 contestants per day was finished for 3-4 hours, comparing to the manual grading during IOI'1998 which took more 12-15 hours per day. After only one year later organizers of IOI'2020 in Beijing, China, used normally implemented (not script but executable) GS. The contests of IOI'2001 (Tampere, Finland), IOI'2003 (Kenosha, Wisconsin, USA), IOI'2004 (Athens, Greece) and IOI'2008 (Cairo, Egypt) were organized with the GS of USACO – developed for national contest and training of the national teams of USA for IOI (Kolstad 2007). Refusing to use GS of USACO, the host Technical committee of IOI'2002 (Yong-In, Republic of Korea) succeed to implement own almost full-functional GS inside 3-4 months. Poland also implemented own GS for IOI'2004 (Nowy Sącz, Poland). Preparing for IOI'2009, a team of Bulgarian students extended the Korean GS to full functional GS SMOC, appending a sand box and stable time measuring (Tochev&Bogtanov 2010). Beside IOI'2009 SMOC was used for organizing Bulgarian contest in IOI style and in some Balkan Olympiads in Informatics.

For IOI'2012 (Sirmione, Italy) GS CMS (Contest Management System) was created. It encapsulated the long years experience in implementing GS and became *de facto* standard for creating such systems grading in IOI style (Maggiolo&Mascellani 2012). Since this moment CMS was used for organizing IOI and many others regional and national contests in IOI style.

Anyway, recently some GS with possibilities for *hybrid grading* (in ICPC and IOI style – with or without grouping of the tests) were implemented. For Bulgarian contests, both for secondary school students and university students, in 2020 was developed the system BOS (Bulgarian Olympic System), which could grade in the three styles (Kelevedjiev et al. 2020). Because of the COVID pandemic BOS incorporate also a module for monitoring behavior of the contestants during the on-line contests.

4. Evaluation with Grading System

Teaching programming is a complex process composed of two streams of activities – *learning* and *practicing*. Getting ability to create computer programs starts with learning. Different kinds of knowledge are absolutely necessary for being programmer. Future programmers have to know some parts of the Discrete mathematics, principles of the programmable machines, computer architecture, at least one operating system, at least one programming language with its library of standard programs, algorithms, etc.

But learning such amount of knowledge is still not enough for being a programmer. Practicing, and only practicing, would demonstrate the effect of learning. Teaching of programming long years ago perceived this concept, nowadays called STEM. And programming contests for young programmers is the instrument of our community to implement this concept. So, for being successful contestant it is necessary to practice, practice and practice.

The benefits of using GS in teaching programming are many. First, using GS the teacher (university professor) could make the process **more intensive**, because the GS give the possibility to perform intensively control testing of the level of progress of the students, and in result to slow the process when the results of testing are not satisfactory. In my practice as a teacher/professor of Programming and Algorithms I had in some years to teach groups of few tens and even hundred students. Before to start using GS we were able to achieve 2, or maximum 3, control testing. With using of GS we are performing control test after the end of each topic.

Second, using GS **spares a huge amount of time**. Without GS evaluation of the works of the students could take few days and even weeks. With GS the results of evaluation are ready in the moment of finishing the test. In such a way the teacher could dedicate the time saved to other activities, for example to preparing tasks.

Third, with manual evaluation of students' programs teacher could execute 3-5 test cases per work, because checking with more tests will increase even more mentioned above necessary time. In programming contests, for being sure that the contestants' programs are perfect, authors of the tasks prepare, sometime, **hundreds of test cases** and this do not increase significantly the elapsed evaluation time.

Fourth, using GS for control testing totally change the behavior of the students during the control tests. When students make a control test without GS they usually check their program with 2-3, not very appropriate test cases or do not test it at all. Huge tests cases, with MiB of input, for the tasks with intensive input, are not performed too. As a result, the students' programs are practically not tested at all. When teacher performs more adequate testing the obtained by the student grades is bad. The GS return for each submitted solution so called *feedback* – how many tests are passed successfully (Accepted), on how many tests the program stopped abnormally (Run Time Error), on how many tests the program overpassed the permitted resources (Time Limit Exceeded or Memory Limit Exceeded) and for how many test cases the program produced not correct result (Wrong Answer). This feedback of GS force students **to debug programs** in order to obtain higher grading. My experience categorically showed that with usage of GS in control tests the results of the students ameliorated significantly. Here is the moment to stress that ability to debug program code is at least so important as the ability to write cod because, statistically, necessary time for writing code is much more than the necessary time to debug it.

Using GS is not just beneficial but generates some difficulties also. The main difficulty is preparing competitive tasks which include a sequence of activities. As a beginning the teacher have to formulate **adequate programming tasks** for the topic adoption of which will be tested. As many tasks as better. Initially this will take much time. But with a constant and systematic work the amount of tasks will increase permanently. It will be necessary more easy tasks for the primary control tests as well as more hard for the final testing. Traditionally, task's statement has to comprise precise formulation of what the asked program has to do, precise formulation of input data format (contest tasks suppose that input data obey the specified format and the student program has not to check the correctness of the input), precise formulation of the format of the output data, limits for the size of input data, some sample input with corresponding sample output, and if it is necessary – explanation of the sample output.

When the adequate task is ready the teacher has to consider **different algorithms** that solve the task. The ideal is to have one **trivial algorithm**, usually directly following from the statement of the task. Such algorithm, with time complexity $O(N^3)$ or $O(N^2)$ for example, students could invent with good knowing of the programming language only. Then one **intermediate algorithm** that involves, beside the good knowing of the programming language, good ruling of corresponding data structuring, with time complexity $O(N.\log N)$ for example. And, of course, one **not trivial algorithm** that needs deep understanding of the tested material and possible algorithmic approaches, with time complexity $O(N)$ for example. For each of possible algorithms a corresponding program has to be written in each of proposed by the teacher programming languages, in order to check possibility to implement the algorithm with the language and its library of standard subprograms.

Next stage of the preparation is creating the **test cases**. According the theory of "*black box testing*" (i.e. testing without knowing the code of the program) the set of test cases has to comprise tests with different sizes – short, middle and large enough. We are practicing a "logarithmic" increasing the size – *N = 10, 20, 50, 100, 200,500, 1000, 2000, 5000, 10000,* and so on. It is clear that short test cases could be prepared by hand. But for middle and large test cases using a **random generation** is inevitable. Set has to include any imaginable specific or extremal combination on test data which could be done mainly in hand made short test.

Having algorithms implementations and the test cases, the author of the task has to determine the **time limit** and **memory limit** that will be applied on each run. Choosing the time limit is the only possible way to control efficiency of the used by the student algorithm when testing is "black box". That is why it has to be chosen in such a way that with the "slow" algorithms no more than 20-30% of the grading points (or 3-3,50 in 6-degree grading system) to be earn, no more than 50% of points (or grades 4-5) to be earned by the middle speed algorithms, and 90-100% (or grades 5,50-6) – by the fastest algorithms. For most of the tasks memory limits are by default fixed to some reasonable quantity – 256 MiB for example. But there are tasks for which the quantity of used memory is a measure for the quality of the algorithm. Such for example are some tasks solvable by Dynamic programming, when the size of the used by the approach table to store solution of the sub problems is crucial.

For most of the competitive tasks the result that program has to find is unique. But some tasks have multiple possible result. When the task is with unique possible result checking it correctness is done with simple comparing of the program input and the output of the author. When the task is with multiple possible answers then the author has to prepare a special program – ***checker*** – that checks correctness of the program result. Writing a checker could be much more difficult than writing a solution of the task because it has to be able to catch many possible deviations of the program output from the specified output format. This difficulty could be escaped by including in the statement of the task a rule that makes the correct output unique – min or max of the possible results (for scalar results), sorted in some way output (when the result is a sequence of values), output in lexicographical order (when the result is a set of sequences), etc.

Sometime the teacher would not like to ask students to write a complete program but just one or few modules (as in mentioned above current trend in programming contests). In such case she/he has to prepare the necessary **main function and the other necessary modules** and supply them to the student via the corresponding function of the GS (in programming contests author's modules are secret because they could contain information that contestants have not to know).

Closing this section, we would like to stress that some GS maintain many programming languages, some of which are not usable (and never will be used) in programming contest. In (Ribeiro et al. 2009) is presented the experience of the authors in teaching Logic Programing with a GS.

5. **Grading Systems in education**

In order to use one GS in programming education some additional functionality have to be appended to it in order to become a *Training System* (Manev et al. 2011). On the first place the GS has to be extended with a *Tasks Repository*. We have mentioned above that creating collection of appropriate tasks is long and time-consuming process. That is why each created task have to be carefully archived in the Repository for using in the future with all its attributes – statement, test cases, expected results, checker(s) (when the possible results are more than one), description and complexity analysis of possible algorithms.

For programming contest test cases, corresponding results and author's description and complexity analysis are secret. But, after the end of the contest, a good practice is contestant to finish the solving of their imperfect solutions and/or to try to solve the tasks that was not solved at all during the contest – this is the essence of the training of contestants. That is why Training System have to include a functionality giving possibilities (without or with some form of control) that provides some access to the task's resources. Exactly the same is appropriate for teaching programming in secondary schools or in the universities.

A modern trend is Training System to be included in Web-based *Training site* (TS), also known as *On-line Judge* site. So, after the necessary registration, each person which is interested has a possibility to train within the system. There are different ways to organize tasks in the Repository of the system. Some TS keep simply *List of the tasks*, others keep them in *Classified categories* depending of necessary algorithmic approach, and third – in a form of *Contest sets of tasks* (from real or "possible" contest). Some TS maintain mixed organization of the tasks.

Some of the Web-based TS judge in ICPC style, other in IOI style. Some TS provide both forms of judging. Statements of the tasks in TS are mainly in English, but in some TS some of the tasks are translated in other languages.

The first Web-based TS we know, that judges in ICPC style, is the University of Valladolid, Spain, Online Judge site or UVaOJ (Revilla et al. 2008; UVAOJ 2022). List of more than 13000 task (including all tasks from past ICPC) are proposed for training as well as sets of tasks, from programming contest or combined by the managers of the site for training, too. Missed classification of tasks based on applicable algorithmic approach is compensated to some level by the fundamental book (Hallim et al. 2020), where each section is illustrated by tasks from UVaOJ's Repository. Hardness of each task could be estimated by the statistical data for the ratio of submitted and accepted solutions.

The first Web-based TS we know, that judges in IOI style, is the training site of the mentioned above USACO (USACO 2022). Tasks in this site are organized in contest sets, mainly from national Olympiads of USA, training camps of national teams of USA for IOI and regularly provided open training contests in 3 levels (Bronze, Silver and Gold division) with different level of difficulty.

First Bulgarian TS Mycamp Arena was created in 2010 (Mihov 2011). Recently mentioned above Bulgarian GS BOS was incorporated in a modern training site (ARENA 2022), keeping part of the name of the first Bulgarian Training site. Tasks from all Bulgarian national contest since 2009 and some international contest where Bulgarian national team took part are classified by topic and in contest sets. Judging is in IOI stile. Statements of some of the tasks have English translation.

Other popular TS are:

- Sphere Online Judge (SPOJ 2022) of Sphere Research Labs proposing more than 13000 tasks, including original tasks of SPOJ team and judging in ICPC style;
- Kattis Problem Archive (KATTIS 2022), which propose list of tasks and IOI style judging;
- Timus Online Judge (TIMUS 2022), maintained by a team from Ural Federal University, Russian Federation. Tasks are mainly from local contests and are organized in different lists. Difficulty of the tasks is estimated by the number of accepted solutions;
- The chineese Peking University Online Judge (PKOJ 2022) and Tsing Hua University Online judge (THOJ 2022), where the judging is in ICPC style and tasks are organized in lists, etc.

6. Creating own tasks and contests

Mentioned above TS are "closed" – the teachers can not include their own tasks in the Repository, neither to create own "contest" and to obtain automatic list of grades of the participant, which is important for using GS in education. Fortunately, some TS provide not very difficult user interface for teachers to include their own tasks in the Repository, to create their own contest/exams and to obtain final grading of the students. We will shortly present here two such TS.

TS Hacker Rank (HACKRANK 2022) is created from consortium of software companies originally as "a technology hiring platform that is the standard for assessing developer skills for over 2,800+ companies around the world." The system has full functionality for organizing programming contest including facilities for creating own tasks, composing task sets and automatically grading of obtained by the participant results. For using the system, a registration is necessary.

Figure 1. Creating a task

After identification of the user, choosing the Link Administration opens two main tabs – Manage Challenges and Manage Contests. For creating a task, we press the button Create Challenge (Fig. 1). In the tab Details of the corresponding windows we fill the attributes of the task in the traditional format of contest tasks. Main attributes – Problem statement, Input format, input Constraints and Output format – are shown on Fig. 2. When the statement is saved we choose the programming language(s) that student could use in the tab Languages. System could judge programs written in 61 languages (or language dialects).

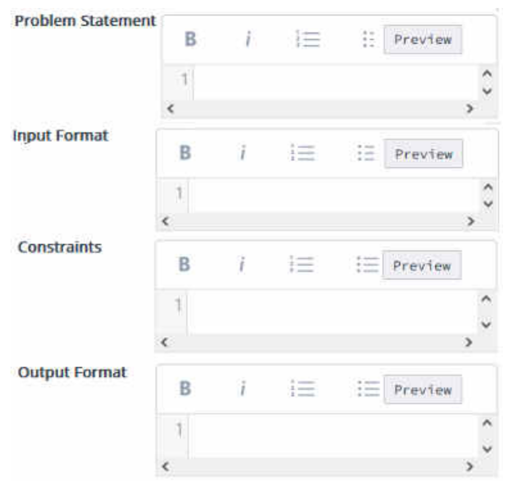


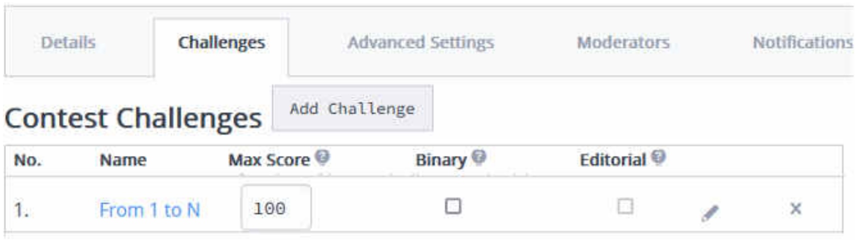
Figure 2. Task statement

In the same tab teacher chooses Time Limit and Memory Limit for the different programming languages. Unfortunately, minimal time limit that HackerRanc can assign for a single test case is 1 sec. which is anachronism having in mind the speed of nowadays computers. This could force the teacher to create very large test cases in order to estimate time complexity of the used by the submitted program algorithm.

Final mandatory step in creating the tasks is uploading the test cases and expected output in the tab Test Cases. For judging in IOI style test cases have to be in separate files, and for ICPC style – in a single file. The test case marked as Sample and the corresponding output will be shown to students in statement of the tasks and will be tested but not graded.

Judging in IOI style HackerRank will assign for each successfully passed test a proportional part of the assigned by the teacher points. Judging in ICPC style, for successive run HackerRang will assign 1 grading point for the task and will register the elapsed time and the punishments (usually 20 minutes for each rejected submit).

When the necessary tasks are ready the teacher could create a contest in the tab Manage Contests. In the tab Details teacher has to fill the start and the end the the contest or to leave the end undefined if it is a long term training contest. Tasks are appended in the tab Challenges (Fig. 3). Each task is appended by name with the button Add Challenge and grading points are filled in the text box Max Score. Saving the defined contest is obligatory, of course.



No.	Name	Max Score	Binary	Editorial
1.	From 1 to N	100	☐	☐

Figure 3. Appending tasks in the contest

During the contest the teacher could observe the progress of the students from the link Current Leaderboard in the page of the contest (Fig. 4) and make corrections of parameters of the contest (end time, assigned grading marks, not correct tests or/and outputs, etc.) from the link Manage Contest. If some parameters of the task are changed during the contest, then the teacher could reevaluate all concerned submits from the link Review Submissions. And more, the teacher could view source codes (extracted by the login of a student or by name of the task) from the link View All Submissions and to provide some remarks or/and suggestions when necessary.

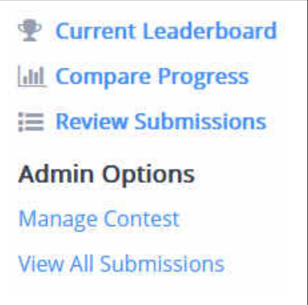


Figure 4.

Another popular TS with the full functionality for creating own tasks and contests is Yandex.Contest (YANDEX 2022). We will not consider here its functionalities in depth because they are very similar to the functionalities of HackerRank. But it is important to mention its functionality for preventing cheating – very important and useful when GS are used for education and especially in the case of online education and examining.

7. Conclusions

Modern Grading Systems, developed to make the evaluation of programs of participants in different programming contest – for secondary school and university students as well as professionals to be hired are result of long years work of very qualified specialists. It is natural that they evaluated to the modern Training sites for contestants and many such sites are nowadays in use. As the preparation of contestants is not principally different from the programming education of future software developers, it is expected that GS could be helpful in education of programming, too.

GS could make the work of teachers in programming in schools, universities and different other forms of programmers teaching more easy and effective which will be very helpful for the society. But the usage of GS for programming education need some additional functionalities to be appended to existing GS. For example, Repositories of tasks to be appended, which is a fact. Other functionalities, as controlled access to the resources are still not completely implemented and this have to be the next step to adapt GS for education in programming.

Bibliography

1. ARENA. *Арена*. https://arena.olimpiici.com/#/, last visited on 23.10.2022.
2. FORIŠEK, M. On the suitability of programming tasks for automated evaluation. *Informatics in Education*, v. 5(1), 2006, 63-75.
3. HACKRANK. *HackerRank for Developers*. https://www.hackerrank.com/, last visited on 23.10.2022.

4. HALLIM, ST.; HALLIM, F.; SUHENDRY, E. *Competitive programming 4*. authors' edition, 2020.
5. KATTIS. *Kattis Problem Archive*. https://open.kattis.com/, last visited on 23.10.2022.
6. KELEVEDJIEV, E.; BRANZOV, T.; PETROV, P.; SHALAMANOV, M. Bulgarian Platform for Competitive Informatics. *Mathematics and Education in Mathematics*, Proc. of 49-th Spring Conference of UBM, Borovetz, 2020. 123–130 [in Bulgarian].
7. http://www.math.bas.bg/smb/2020_PK/tom_2020/pdf/123-130.pdf
8. KOLSTAD, R.; PIELE, D. USA Computing Olympiad (USACO). *Olympiads in Informatics*, v. 1, 2007, 105–111. https://ioinformatics.org/journal/INFOL016.pdf
9. MAGGIOLO, ST.; MASCELLANI, G. Introducing CMS: A Contest Management System. *Olympiads in Informatics*, v. 6, 2012, 86-99.
10. https://ioinformatics.org/page/ioi-journal-index/44
11. MANEV, KR.; SREDKOV, M.; BOGDANOV, TS. Grading Systems for Competitions in Programming. *Mathematics and Education in Mathematics*, Proc. of 38-th Spring Conference of UBM, Borovetz, 2009, 103-116.
12. http://www.math.bas.bg/smb/2009_PK/tom_2009/pdf/103-116.pdf
13. MANEV, KR.; SREDKOV, M.; ARMYANOV P. Software Platform for Teaching Programming with Grading Systems. *Mathematics and Education in Mathematics*, Proc. of 40-th Spring Conference of UBM, Borovetz, 2011, 300–305.
14. MANEV, KR. Teaching computer programming in schools, *Acta et Commentationes*, 4(18), 2019, Chisinau, ISSN 1857-0623, E-ISSN 2587-3636.
15. MAREŠ, M. Perspectives on grading systems. *Olympiads in Informatics*, v. 1, 2007, 124-130. http://www.mii.lt/olympiads_in_informatics/pdf/INFOL003.pdf
16. MIHOV, V. Maycamp Arena Project – using contest grading systems in programming education, *Mathematics and Education in Mathematics*, Proc. of 40-th Spring Conference of UBM, Borovetz, 2011. 438-443. http://www.math.bas.bg/smb/2011_PK/tom/pdf/438-443.pdf
17. PC^2 HOME PAGE. http://www.ecs.csus.edu/pc2/, last visited 13.10.2022.
18. PKUOJ. *Peking University Online Judge*, http://poj.org/, last visited on 23.10.2022.
19. REVILLA, M. A.; MANZOOR, S.; LIU, R. Competitive learning in informatics: the UVa on-line judge experience. *Olympiads in Informatics, v.* 2, 2008, 131-148.
20. http://www.mii.lt/olympiads_in_informatics/pdf/INFOL035.pdf
21. RIBEIRO, P.; SIMÕES, H.; FERREIRA, M. Teaching Artificial Intelligence and Logic Programming in a Competitive Environment. *Informatics in Education,* v. 8(1), 2009, 85-100. http://www.mii.lt/informatics_in_education/pdf/INFE143.pdf
22. SPOJ. *Sphere Online Judge,* https://www.spoj.com/problems/classical/, last visited on 23.10.2022.
23. SREDKOV, M. *Contest grading system spoj0*. Diploma thesis for obtaining MSc degree, Faculty of Math. and Comp. Science, Sofia University, 2006 (in Bulgarian).

24. THUOJ. *Tsing Hua University Online Judge*, https://dsa.cs.tsinghua.edu.cn/oj/, last visited on 23.10.2022.
25. TIMUS. *Timus Online Judge*. https://acm.timus.ru/problemset.aspx, last visited on 23.10.2022.
26. TOCHEV, T.; BOGDANOV, TZ. Validating the Security and Stability of the Grader for a Programming Contest System, *Olympiads in Informatics,* vol. 4, 2010, 113-119.
27. http://www.mii.lt/olympiads_in_informatics/pdf/INFOL058.pdf
28. USACO. *USACO Training Program Gateway*. https://train.usaco.org/, last visited on 23.10.2022.
29. UVAOJ. *University of Valladolid On-line Judge*. https://onlinejudge.org/, last visited on 23.10.2022.
30. YANDEX. *Yandex.Contest Online Judge,* https://contest.yandex.com/, last visited on 23.10.2022.