Augmenting CS1 courses with a semester-long contest

C. Maria Keet
Department of Computer Science
University of Cape Town
Cape Town, South Africa
mkeet@cs.uct.ac.za

The two main first-year Computer Science courses at the University of Cape Town have increasing enrolment figures, reaching 850 students for the Introduction to Computer Science—problem solving and Python—in 2016 and an expected 600 students for the second semester course—object-oriented programming and Java. The CS1 courses are compulsory for several degrees in the faculties of Science, Engineering, and Commerce, with the effect that assignment topics are of a general nature rather than direct relevance to a student's major. Moreover, students arrive with a wide range of programming competencies because relatively few secondary schools offer IT. In addition, due to the highly unequal wealth distribution in South Africa, learners may not have had regular access to computers outside school. Consequently, at the start of CS1, no programming experience can be assumed from the students, yet multiple students do have this, and if so, principally in Delphi, Java, or Python. Streaming students based on prior programming experiences has shown to deepen divides, yet not accommodating students with prior programming skills leads to boredom and some of the most competent students leave CS for other degree programmes; both are deemed undesirable consequences. Further, there is anecdotal evidence that programming contest students perform better in the 3rd-year theory of algorithms course. Finally, contest winners tend to be a very small pool of students mainly from privileged secondary schools and who have participated in national secondary school programming contests. This pool possibly could be extended with top CS students who have not yet been exposed to programming contests.

These observations have led to the idea to augment the CS1 courses with a contest component running throughout the semester. In designing the set-up of such a contest, the following principal considerations were taken into account, with R=requirement or issue, S=solution.

- R: time cannot be used as tie-breaker, and it would discourage late-comers. S: Allocate points to a problem, and deduct 1 point for each wrong submission.
- R: Problems have to be of varying level of difficulty, some have to match the course topics and others have to go beyond that, and the problem descriptions should be relevant to several degrees. S: Existing contest problems were selected bearing in mind the former two, and rewritten on topic and tailored to South Africa to cater for the latter; e.g., an electricity load-balancing app, a pre-paid mobile phone record start-up in Cape Town, and bacteria populations.
- R: 'Late' entrants—those who had not any programming experience before—should have a chance at winning prizes, to incentivise participation. S: In addition to the top-5 and 'first to solve', a new category 'first new person to solve' the problem was added.
- R: The average and struggling students should not be discouraged by the contest. S: The contest runs on a separate course site in the course management system (UCT's Vula, which is based on Sakai).
- R: Seamless automated judging. S: The automated marker for assignments, developed by a colleague (H Suleman), was repurposed to meet this requirement, as it can be plugged into the course management system.

The first instalment, the 1016Challenge in 2015, had all 10 problems released at once. This resulted in 'binge solving' as if it were a weekly assignment, so an additional set of 3 problems was released later in the semester. For the 1015Challenge, running now in 2016, three problems are released every fortnight, which generates more recurring engagement. Most problems are being solved in the first two days from their release, and then during public holidays, mid-term vacation, and near the deadline. Participation rate for the 1015Challenge is still increasing during the semester, which barely happened after the 1st week of the 1016Challenge. In both instalments, about 10% of the enrolled students participated, most of whom have not had any prior experience with programming contests. This shows that such a course-associated contest substantially lowered the barrier to exposure to programming contests and also constitutes a substantial pool of new candidates. Informal feedback is positive, from second-year tutors feeling to have lost out in their first year without a contest, and first-years hoping to see the same in their second year.

The 1016Challenge of 2015 will be re-run in 2016 to determine whether having had exposure to contest problems in the first semester will result in better or more problem solving in the second semester. The rewritten problems are available upon request.

Acknowledgments. I would like to thank Takunda Chirema and Robin Visser for their assistance in the implementation.

CLI '16, Phuket, Thailand

ACM ISBN 978-1-4503-2138-9.

DOI: 10.1145/1235