Valid, scalable, authentic, on-site, digital, individual assessment of programming skills

Thore Husfeldt, *Dept. of Computer Science, Lund University and IT University of Copenhagen*

***Abstract*—I summarise the ideals and challenges of implementing an exam in introductory computer programming, and report on a successful implementation in 2018 at ITU. This includes pedagogical, curricular, technical, social, and administrative devisions and experiences that scale to other setting in which a student-owned device is used for individual assessment. The core issue is the difficulty of preventing communication with outside sources using a computer with internet capabilities.**

I. Introduction

SUMMATIVE assessment of individual programming skills needs to simultaneously honour conflicting ideals: Programming is typically done on a computer, not on paper, and programmers actively use an interactive edit-run-debug cycle when developing functional code, which is impossible to emulate in a pen-and-paper exam setting. Moreover, programming environments are highly individualised, to the level of keyboard layout, choice of editor, etc., so that an exam setting ideally lets students use the same computer they use for programming. Today, this is the student's laptop. These devices are capable of connecting to the internet, which allows exam sets to be distributed and collected using a modern learning platform. These are great opportunities for a scalable and authentic assessment format for individual programming skill.

At the same time, a modern laptop accesses the internet freely, allowing students access to outside sources during the exam. Students are highly incentivised to use existing, efficient, and fairly cheap professional services or exploit social connections to have their exam solved by somebody else, completely invalidating the result of the assessment.

As part of a project at IT University of Copenhagen, to introduce newcomers to introductory programming we constructed an exam format that tries to address these issues.

II. Initial Considerations

A. Ambitions

The ambition was to construct an exam form for introductory programming with the following qualities:

1. (*Validity*.) The assessment should represent the examinee's individual programming skill, not that of their social network.
2. (*Scalability*.) The exam form should be implementable for hundreds of students several times per year.
3. (*Authenticity*.) The exam form should assess individual programming skill within a typical environment. (However, this excludes the access to outside help, notwithstanding possible arguments that social capital and plagiarism are an important part of programming.)
4. (*On-site*.) The assessment is performed in a controlled environment on university premises.
5. (*Digital*.) Code is written on a digital computer.
6. (*Individual*.) The assessment is individual.

Introductory programming is mainly concerned with basic, operationally critical, and easily assessed programming skills—code comprehension, syntax of a specific language, creation of small stand-alone programs (dozens of lines of code), manipulation of medium-sized programs (hundreds of lines), or use of library functions. The above exam format parameters suit such a course well. In contrast, an advanced course in software engineering would focus on much larger systems, group work, report writing, etc., which may be more compatible with project based formats, group exams, or take-home exams.

In the current setting, individual assessment of demonstrable basic programming skills was an important external constraint due to a larger process of curricular change at university level.

The usual problems with outside communication (sometimes called "cheating") are exacerbated for basic programming because it is very hard to detect plagiarism (correct solutions tend to look the same), problems are quickly communicated to the outside (by compressing the exam to a single file and sending it over any of dozens of protocols over the internet), exam performance correlates highly with learning outcome (in the sense that an experienced programmer can solve the exam one or two orders of magnitude faster than a struggling examinee), and the service is available either in exchange for social capital in the examinees social network or by dedicated professional services—online, affordable, and reliable help for programming exams is a business model. Moreover, this particular course is critical for student progression in their desired education, and it can be expected that students face a difficult trade-off between earnest desire for honesty and critical career goals.

B. Why not use pen and paper?

The traditional way of individually assessing programming skill is via a pen and paper-based written exam where the examinee hands in handwritten code. The examinator evaluates the quality of the code by reading it. No code is ever executed by an electronic computer.

This format remains attractive for many reasons, not the least of which is its compatibility with established examination routines for distributing questions and collecting answers. By being inherently offline, it does not

invite outside communication.

However, the pen and paper-format falls short on a number of other points relating to validity and authenticity:

1. (*Amount of text.*) Fundamental programming tasks involve the engagement with, manipulation, and maybe creation of chunks of text of nontrivial size in a programming environment (say, a code editor). This is not feasibly assessed using pen and paper.
2. (*Debug cycle*.) Computer code is almost never written flawlessly. Instead, most programmers use an edit-run-debug cycle, in which tentative code is written, then tested, and edited to remove errors. This cycle is repeated many times. Most programmers spend their time *editing* code, rather than *writing* it. This is a core competency of introductory programming and impossible to simulate without access to a computer.

C. Why not use departmental machines?

In the early 2000s, we experimented with computer-based individual programming exams at the computer science department of the natural science faculty (CSNatFak). There, students completed the exam seated at a unix machine in one of the computer labs. These were the same machines used for instruction and exercises during the course, so students were familiar with those machines and their programming environments. The machines accessed the internet through a wired (ethernet) connection whose access could be monitored by departmental technicians. During the exam, students were forbidden to access the internet for other purposes than downloading the exam questions and uploading their solutions. This restriction was enforced through a plausible threat of monitoring their internet access.

Similar setups remain viable and in use for programming competitions, where relatively small groups of very resilient and experienced programmers compete for social status.

While the CSNatFak solution worked very well, it is no longer viable for our purposes, because of changes in student demographics, wireless internet, and near-universal access among students to portable computers.

Today, most students use—and are encouraged to use—their own computer for programming, instead of a departmental lab computer. This changes the situation compared to half a generation ago in a number of ways:

1. (*Scale*.) The number of students attending a large, introductory programming exam is in the 100s and typically far exceeds the number of computers that a department can make available at the same time in a controlled environment and in a reliable and cost-efficient manner.
2. (*Familiarity*.) Programming environments are often highly personalised workspaces that reflect individual preferences in choice of code editor, integrated development system, colour schemes, and physical keyboard layout. In particular, beginning students with highly heterogeneous prior experience in programming and computer use can be expected to be significantly constrained from demonstrating their programming skills at the exam if the setup requires use of an alien programming environment.
3. (*Internet access*.) Today, internet access is no longer restricted to wired connections via ethernet, but provided through a variety of channels, including wireless channels.

In summary, it seems difficult to implement a digital exam on university-controlled computers today.

III. Other Rejected Approaches

A number of vendor-based systems that promise valid digital exams while claiming to restrict access to the internet were quickly rejected. Most of these systems seem to provide various standard exam forms (such as multiple choice or free text) in a web-based interface, and use the web browser to report loss of window focus (or termination of the web browser session) to the server. At the time of writing, none of these systems seem relevant for our purposes.

A more attractive solution is to make students insert an external hardware device into their machine, which runs invasive software in the background to monitor internet access, or contains its own small programming environment. This solution somewhat addresses some of the concerns with the university-owned machines, such as scalability, but still requires extensive hardware maintenance. The security and legal implications are not easily understood.

Administrators seriously suggested monitoring internet use during the exam by having invigilators or even teachers ambulate through the exam rooms. This suggestion seems to be based on a misunderstanding of how easily an online programming exam can be communicated to the outside, and how difficult it is to distinguish the required commands from virtuous programming. This suggestion was rejected.

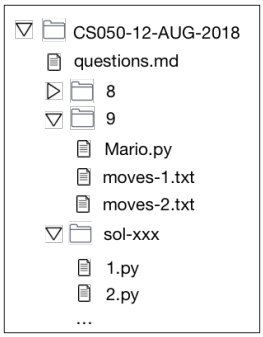
A final, perhaps surprising, suggestion was to monitor student behaviour during the exam using cameras placed above or behind student computers, which record all behaviour (in particular, everything that is typed) and can be consulted after the fact in cases of suspected dishonesty. This is less intrusive than a hardware-based solution (whether by inserting a dongle on the student's device or using a university-controlled machine), but *feels* much more intrusive. This solution scales somewhat worse (due to the fact that dozens or hundreds of cameras need to be acquired, maintained, and installed). However, note that due to the strong *impression* of surveillance, the system may not actually need to work in order to have the desired effect of disincentivising examinees from seeking outside help.

IV. Procedure at IT University of Copenhagen

A. Exam contents

The exam itself lasts 4 hours and consists of a number of programming exercises of increasing difficulty and range. The examinee downloads the exam from the learning platform at the beginning of the exam as a single folder containing both the questions as a single file in Markdown format ("exam-questions"), some additional files needed for the larger tasks, and a solutions directory into which the student places their answers. (For consistency, this directory contains empty files for each answer which the examinee is supposed to edit.)

A typical medium question asks students to consider a small piece of given code, and



1. edit it so as to perform according to specified behaviour (this may include running the code to see what goes wrong, identify the error, and fix it),
2. add comments or other documentation to make it human-readable and easier to maintain,
3. rewrite it so as to maintain functionality but adhere to another paradigm (for instance, object orientation)
4. add functionality by extending the code.

The hardest of these exercise typically involve manipulating a small textual database or writing a very simple video game engine (with text-based or no graphics.) The easiest exercises are mere code comprehension and mastery of syntax.

B. Procedure

The exam works like this:

1. (*Weeks in advance*.) Examiner constructs all the files necessary for the exam. Various standard quality control mechanisms are executed (a teaching assistent solves it and provides feed-back, other teachers may have to be consulted for constructive alignment, etc.). The exam is uploaded to the learning platform, where it remains invisible until the time of the exam.
2. (*Beginning of exam*.) The examinee arrives with their own laptop, logs in to the learning platform, and downloads the CS050-12-AUG-2018 folder to their machine.
3. (*Exam*.) The examinee renames the "sol-xxx" folder to their own user ID (say, "sol-jdoe"), solves some of the exercises, and puts the result into the renamed solutions directory.
4. (*End of exam*.) The examinee uploads the renamed solutions folder to the learning platform.
5. (*Grading*.) The examinator downloads the solutions from the learning platform and starts grading.

C. Internet Access Restriction

Student access to the internet is restricted using the existing firewall management system at ITU. A simplified description of the process is this:

1. (*Weeks in advance, well-established procedures.)* Students register for exams well in advance, using existing procedures, based on enrolment, successful completion of mandatory activities, and external regulations. These procedures are handled by the Examinations Office at ITU, which is part of the Student Affairs and Learning department.
2. (*One week in advance, manually*.) From the list of registered students (say, John Doe 200205043823), a technician at the IT Department creates the corresponding list of ITU network users (say, jdoe@itu.dk). This group is visible for the network access control ("firewall") settings of the ITU domain control system.
3. (*Morning of the exam, automatically*.) At the day of the exam, the firewall settings of the group are restricted to access only the relevant web page of the learning platform.
4. (*During exam*.) Students arrive with their own laptop. Invigilators perform standard identification routines to verify student identity. Students voluntarily relinquish electronic communication devices (mobile phones, smart watches) and disable access to the cellular network by removing identification modules ("SIM card") from their laptop if needed.
5. (*Day of the exam, automatically*.) After the exam, the group's firewall settings revert to normal.

In effect, all internet access for registered students through the ITU network, including wireless access, or access through *eduroam*, is restricted to accessing the exam on the learning platform and uploading the answer.

V. Conclusion

A. Validity in the future

Various options for accessing the internet remain for students. This includes short-range communication with wearable access points to the cellular network (say, Bluetooth to an extra mobile phone hidden on the body) or other networks in geographic reach. These opportunities will only increase with increasing technological miniaturisation and ubiquitous connectivity of wearable devices.

To the extent to which universities aim to maintain any form of valid assessment, examination forms need to react to these technological changes in order to ensure valid exam forms that do not punish honesty, ultimately to retain public trust. This seems to require physical measures on the level of building design to create examination environments in which students are guaranteed to be off-line, as well as moderately invasive checks for worn electronic devices when entering such environments, comparable to airport or museum security checks. This seems to be a major responsibility that requires consideration at a much higher level.

B. Further Automatisation

The fact that exam answers are electronically available to the examiner in a standardised format would make it feasible to automate grading, at least in part, by subjecting student code to a battery of automated test. We have not pursued this possibility.