

# Improved Automated Marking of Student Programming Assessments

Nir Oren\*, Greg Myers

University of Aberdeen

\*contact author, email: n.oren@abdn.ac.uk

## The Problem

Programming is an applied skill. While the effects of basic programming constructs can easily be learned and understood, the ability to apply this learning to solving a computational task is more difficult to instil in students. Like many other applied skills, practice and repetition are critical to becoming a competent programmer.

In a University setting, limited marker resources are a major obstacle to providing students with chances to practice their programming. The standard practice in introductory computing courses is to set students small formative assessments to achieve simple tasks. Here, the only feedback available is whether the task succeeds or fails. These formative assessments are followed by one or two larger summative assessments, where the student's ability is evaluated and detailed feedback given.

We have found that the students who need the most help typically "suffer in silence" once they start falling behind in the formative assessments, and cannot hope to catch up by the time the summative assessments come around. We have also found that the amount of programming done by students means that they do not have enough experience to solve more difficult programming challenges in later years. Finally, students often ignored the formative assessments.

To address these issues, ***we needed a way to provide students with more summative assessments, without increasing marking effort.***

## Automated Program Marking

CodeMarker is a web based tool for automated program marking. CodeMarker allows a lecturer to set up assessments, and enables students to repeatedly submit their source code to be run against these assessments.

Codemarker tests student submission output for desirable properties, and informs the student whether their submission met, or did not meet, the submission requirements.

Outputs can be more than just a pass/fail - assessment submission, execution and evaluation takes the form of a program, enabling complex marking behaviours to be created.

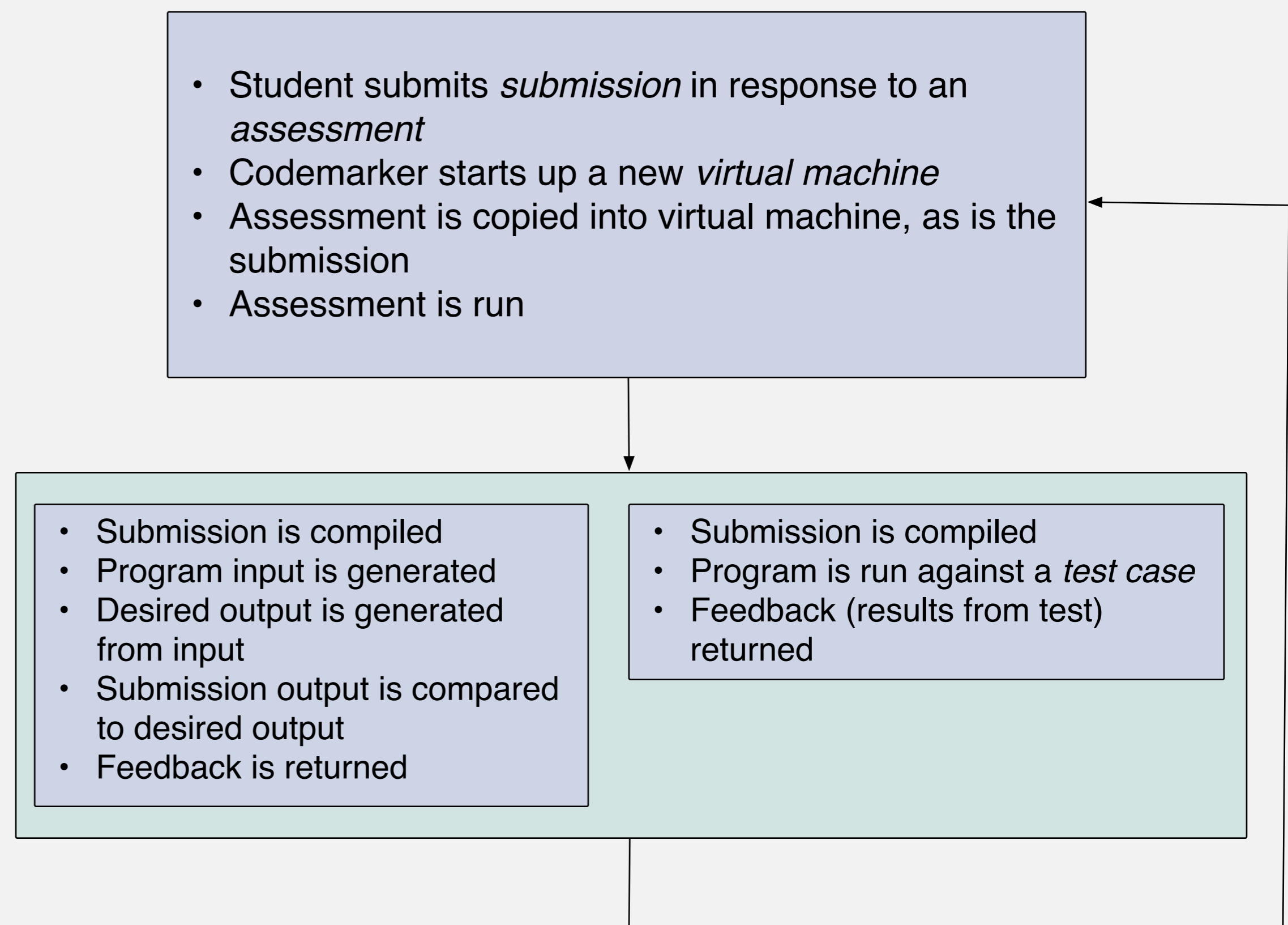
## CodeMarker Usage

The Department of Computing Science has used CodeMarker in several first and second year courses. Its use has allowed us to change our teaching style, providing students with continuous programming assessments and immediate feedback.

Codemarker has been used in both Introductory programming courses and a more advanced Algorithmic Problem Solving course, where imaginative thinking is required in order to solve assessments.

Student Feedback has generally been positive, but no formal assessment of CodeMarker as an intervention has yet been performed.

## Workflow Outline



## CodeMarker Features

### For Assessors

- Set up an assessment, with start and finish times.
- Mark submissions written in many languages (Java, Ruby, Python and C/C++ have all been tested).
- Keep track of source code submissions for audit purposes.
- Complex marking behaviours based on student program output.
- Allows both static and dynamically generated inputs and outputs to submissions.

### For Students

- Allows repeated submissions of source code.
- Instant feedback.

### Miscellaneous

- Security provided via a virtual machine architecture that is wiped after use.

## Future Work

A wide list of future enhancements is planned, which will help assessors (e.g. different assessment types), students (e.g. a library of feedback which they can receive; improved user interface), and administrators (improved scaling capability, integration with LDAP and the myAberdeen infrastructure).

We are in discussions to roll out CodeMarker across the university, and potentially to other higher academic institutions.

From a research perspective, Investigating the effects of CodeMarker on student outcomes is one area of potential future work that could be undertaken.

## Acknowledgments

CodeMarker was originally conceived and Developed by Nir Oren, and then reimplemented by Greg Myers and Chris Shanks