ABSTRACT
Programming requires both to write code and to read code. In debugging tasks, students can use supporting tools that create additional artefacts like visualizations or traces, but these again need to be read and understood. This contribution presents a concept of code reading exercises that asks students to create the same artefacts for given code as they would use for debugging their own code.

Categories and Subject Descriptors
K.3.1 [Computers and Education]: Computer Uses in Education—Computer-assisted instruction (CAI)

General Terms
Languages

Keywords
Programming; E-Assessment; Reading Code; Feedback Generation

1. INTRODUCTION
In programming, it is not only important to write code, but also to know how to read code: Reading code is necessary to be able to learn from examples; debugging own code requires to read it in order to check where it deviates from the intended behaviour; and collaborative exercises and team projects require to read code written by others.

The first aspect can be tackled by quizzes, where students demonstrate their understanding by answering questions about a given piece of code. The second aspect is often supported with respect to debugging, e.g. by providing additional artefacts such as visualizations [1] or traces [2]. However, this in turn requires the ability to understand these artefacts. Hence it is desirable to train students in reading code by asking them to create these artefacts. It can be assumed that this in turn will help them to also use these artefacts in debugging tasks. As visualizations and traces can be created automatically, automated feedback generation is possible. This contribution elaborates on the use of traces as one possible realization of this concept that has so far been implemented as a prototype in an e-assessment system.

2. EXERCISE DESIGN AND INTERFACE
The user interface for this kind of exercise consists of two parts: In the upper section, a piece of source code is presented to the students. In the lower part, a skeleton of a trace table is provided. It provides a column for line numbers and allows students to insert an arbitrary number of additional columns for each variable used in the source code. In addition, students can add an arbitrary number of lines to the trace. With this interface, students are able to create a complete trace table for the given piece of code and some given input. Notably, exercises can be generated from submissions to other programming exercises, thus also tackling the third aspect mentioned in the introduction. Moreover, one exercise can be attended repeatedly for different inputs that result in different traces.

3. FEEDBACK OPTIONS
Feedback can be based on different observations: The number of columns or the name of columns can be wrong, which indicates errors in understanding the variables used in the program code. The number of lines or the sequence of line numbers can be wrong, which indicates errors in understanding the control flow of the program. Finally, individual values in the trace table cells can be wrong, which hints towards errors in understanding data flow or the semantics of individual operations. All observations can be made automatically by comparing the student’s response to an automatically created trace for the same piece of code.

4. REFERENCES