Chapter 6
A Web-Based System for Error Correction Questions in Programming Exercise

Yoshinari Hachisu
Nanzan University, Japan

Atsushi Yoshida
Nanzan University, Japan

ABSTRACT
In this chapter, the authors propose a system for generating error correction questions of programs, which are suitable for developing debugging skills in programming education. The system generates HTML files for answering questions and CGI programs for checking answers. They are deployed on a Web server, and learners read and answer questions on Web browsers. It provides an intuitive user interface; a learner can edit codes in place at the text. To make programs including errors for learning debugging, the authors analyze types of errors and define the processes of error injection as code transformation patterns. If learners can edit any codes freely, it is difficult to check all possible answers. Instead, the authors adopt a strategy to restrict editable points and possible answers from the educational view. A demonstration of error correction questions of the C language is available at http://ecq.tebasaki.jp/.

INTRODUCTION
In programming courses, students learn skills for coding, code reading, and debugging through various kinds of exercises. Although debugging is important for actual software development, it is difficult for students, even who have good understandings of programming, to acquire debugging skills effectively (Ahmadzadeh, Elliman, & Higgins, 2005; Chmiel & Loui, 2004; Lahtinen, Ala-Mutka, & Järvinen, 2005). In this chapter, we focus on a Web-based learning support system for debugging, whose controlled exercises are difficult to be provided.

Three types of exercises are often used in programming courses: describing full codes, filling empty boxes embedded in the texts of codes (fill-in-the-blank questions), and error correction questions. The last one is suitable for debugging. Describing full codes is a most typical way of
exercises, and through them learners find errors and inevitably try to debug for detecting faults. Experiences of debugging, however, are different for each learner, and teachers cannot control experiences for them to use all necessary debugging skills. In exercises with fill-in-the-blank questions, learners may not have chances to debug because the codes they read do not include any errors, and they can fill answers without compiling and testing. In the case of error correction questions, teachers can select errors for injecting to a correct program according to debugging skills to be learned. Learners need to detect and correct errors that are injected purposely.

To adopt error correction questions in actual programming exercises, we need to solve two problems: a user interface of a question and answering system and the cost of making questions. Some researches propose systems for generating error correction questions (Hachisu & Yoshida, 2013, 2014; Itoh, Nagataki, Ooshita, Kakugawa, & Masuzawa, 2007; Suganuma, Mine, & Shoudai, 2005). Suganuma et al. have proposed the Web-based system, but its interface is not intuitive. It shows an incorrect program and some texts are marked with id numbers to show the candidates of errors. When learners find errors, they need to write id numbers and correct texts in boxes that are displayed apart from the program; they cannot correct them on the spot. Marked texts give clear hints of answers to learners. On the other hand, when learners can edit any texts freely, some answers may be unexpected ones for teachers; in an extreme case, learners may make a whole new program whose input and output are identical to the original one. Consequently, making error correction questions requires a fair amount of work for teachers. It requires injecting various errors in code fragments and check the correctness of all possible answers, some of which may be unexpected one but correct. Itoh et al. have proposed a method for generating programs containing errors for learning algorithms, but it doesn’t provide a way to check the answers; it is supposed to be used for oral examinations. The key questions about constructing the systems for error correction questions are what kind of user interface is suitable, how to manage error injection into codes, and how to cover all possible answers in correctness checking.

In this chapter, we propose a system for generating error correction questions and checking the correctness. The system generates HTML files including editable text forms for answering questions and CGI programs for checking answers. They are deployed on a Web server, and learners read and answer questions on Web browsers. It provides an intuitive user interface; a learner can edit codes in place at the text. Editable texts are restricted and hidden from learners by using a jQuery plug-in, JEIP (Scott 2014). For management of error injection, we have analyzed types of errors and defined the processes of error injection as code transformation patterns, which we call error patterns. The system synthesizes code fragments including errors by transforming correct code fragments according to selected error patterns. If the system allows learners to edit any texts in codes, it needs to accept all possible correct answers. Full coverage of all possible answers is difficult. Instead, we adopt a strategy to restrict possible answers to the ones that are reasonable for learning objectives, which mean the syntax and semantics of the language, typical program descriptions, and algorithms that learners should understand. We have analyzed types of possible answers and proposed constraints of editable places in codes on answering questions. For evaluating the system, we have implemented a prototype of the system and made an experiment of the programming exercise using the generated questions. Our system can generate questions that widely cover the syntax and semantics of the C language such as variables, types, conditional branches, loops, arrays, functions, strings, pointers, and structures, and some algorithms such as