New Orleans JAZZED about Engineering Education

Paper ID #14906

A System for Analysis of Code on Cloud as an Educational Service to Students

Mr. Jinyi Zhang, Purdue University

Jinyi Zhang is going to receive his B.S degree in computer engineering from Purdue University, West Lafayette, IN, in 2016. He led a team of undergraduates building the system for Analysis of Code on Cloud as an Educational Service to Students. He aims to explore machine learning and data mining by pursuing an M.S degree in next two years, and probably a Ph.D. degree in the future.

Mr. Pan Fengjian, Purdue University, West Lafayette

Fengjian Pan is a senior student major in ECE at Purdue university, West Lafayette, IN. He is the current leader of A.C.C.E.S.S. team.

Mr. Mrigank S. Jha, Purdue University, West Lafayette

Mr. Jha works as a Web Developer for World Wide Technology in St Louis, Missouri. He graduated from Purdue University in December 2015, with a Bachelor's Degree in Electrical Engineering. He worked as a front-end developer for A.C.C.E.S.S - Analysis of Code on Cloud as an Educational Service to Students.

Mr. Pranav Marla, Purdue University

Pranav Marla is an undergraduate student at the College of Science in Purdue University. He is pursuing a major in Computer Science, with a specialization in Machine Intelligence. He designed the entire backend of A.C.C.E.S.S.

Mr. Kee Wook Lee, Purdue University

Kee Wook Lee is a senior student at Purdue University, West Lafayette, IN, majored in electrical engineering.

Dr. David B Nelson, Purdue University, West Lafayette

David B. Nelson is Associate Director of the Center for Instructional Excellence at Purdue University. He received his Ph.D in World History from the University of California, Irvine in 2008.

David has been involved in many educational research projects at Purdue, including published worked in the programming education, student engagement and academic performance in dynamics engineering courses, and educational modalities in engineering, technology and economics.

Dr. Yung-Hsiang Lu, Purdue University

Yung-Hsiang Lu is an associate professor in the School of Electrical and Computer Engineering and (by courtesy) the Department of Computer Science of Purdue University. He is an ACM distinguished scientist and ACM distinguished speaker. He is a member in the organizing committee of the IEEE Rebooting Computing Initiative. He is the lead organizer of the first Low-Power Image Recognition Challenge in 2015, the chair (2014-2016) of the Multimedia Communication Systems Interest Group in IEEE Multimedia Communications Technical Committee. He obtained the Ph.D. from the Department of Electrical Engineering at Stanford University.

A System for Analysis of Code on Cloud as An Educational Service to Students

Abstract

Guiding students to move smoothly from beginner to professional programmers presents many challenges. Modern software development and management requires knowledge and skills of multiple tools. Learning programming without these tools is like learning flying without instruments. Common approaches can be generally categorized into two branches. First, the instructors explain programming tools at the beginning of the courses, before the actual needs arise and hence lacking the context of the tools. Second, the students focus on coding without learning these tools; this approach can be deficient when the students enter a workplace where knowledge and skills in using these tools are expected.

This paper presents a new approach: a system designed to guide students learning programming using a broad range of tools. This system provides a web interface which hides all the tools behind the interface and runs students' programs on containers. Students gain a practical understanding of the valuable information presented by these tools without learning how to install, configure, execute, and update them. The system can analyze and test students' programs based on a set of specifications from the instructors, and provide personalized feedback about the mistakes each student makes. Furthermore, the system can help the instructors identify common mistakes before the assignments are submitted for grading.

The core functions of this system have already been implemented. It is operational and is performing alpha testing in a sophomore-level C programming course. We will conduct an online survey and will have an in-person focus group at the end of this semester. The survey will measure students' perceived value of and satisfaction with the system, particularly in comparison with existing methods of coding instruction. The utility of our system is gauged with a detailed focus group, conducted by an external interviewer. These focus groups follow Krueger's framework to identify benefits and potential limitations of the system, as well as suggestions for improvement.

Introduction

Technologies are foundations of modern societies and software plays an essential role in technologies. Recognizing the importance of software, computer science has become an integral part of general education in USA [1]. Developing high quality software is a complex process and many tools are involved, for example, version control, debugging, performance profile, test coverage, memory access violation, and resource utilization. As an analogy, these tools are instruments in an airplane; these instruments provide crucial information about the condition and status of the plane. The speedometer informs a pilot how fast the plane is flying; this is similar to the performance profile of a computer program. Modern planes have many sensors; software should also be diagnosed by advanced tools. Flying without the instruments can be dangerous.

Pilots need to understand the information presented by the instruments but they do not have to install, configure, calibrate, or repair these instruments. In contrast, when students learn tools for software development, students need to install, configure, and update these tools. Imagine that each pilot needs to install and calibrate the speedometer of a plane! Tools for software development may have many options and learning these tools can be overwhelming for beginners. There are two possible solutions (1) teaching tools so that students are familiar with the tools or (2) teaching students programming without using many tools. For the first approach, students may not see the need of these tools yet. The second approach is equivalent to teaching flying without instruments. Students do not understand many essential properties of their programs. Thus, neither approach is ideal. A better approach is to adopt what the aviation industry has been doing: separate the responsibilities. Pilots focus on controlling airplanes, leaving design, installation, maintenance, and repair of instruments to other experts. This is the third approach we suggest in this paper.

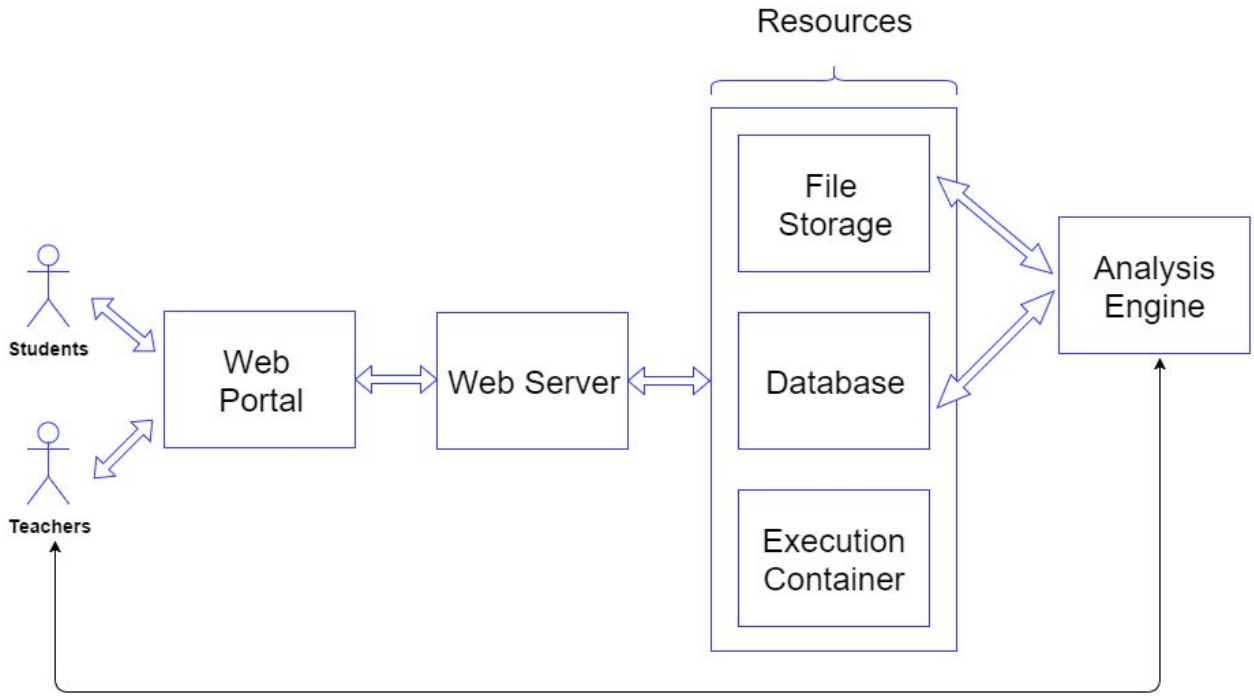

Figure 1: The Architecture of the Proposed System. The main components include a web portal, a web server, a file system, a database, an execution container, and analysis engines. The necessary tools are integrated into the execution container.

We recognize the need to provide detailed analysis of students' programs, without the burden of managing these tools. Our approach hides all the analysis tools behind a web interface, adopting the concept of "software as a service". The website provides a fully-featured code editor and same functionalities as a local file system. A student may create folders, move, copy, or delete files. Students interact with our system by typing, pointing, and clicking as they do for most other applications. Our graphical user interface aims to provide students direct control and predictability. Most text editing keyboard shortcuts, such as copy, cut, paste, find, and replace are enabled in our system. Version control is built-in: a student clicks a button to commit a

version. The student can see the history of changes without the knowledge of creating a repository. The repositories created by this tool is private in order to prevent academic dishonesty. Our system currently supports C programs and we plan to extend it to other languages in the near future. The execution engine is a Docker container. The system automatically generates `Makefile` to compile and link the source code. When a program is tested, the system invokes a set of analysis programs, for example, `Valgrind` to detect invalid memory access. If a program terminates abnormally, the system invokes `gdb` and reports the call stack information. The system also allows modular testing. An instructor provides the correct answers for the functions needed for a programming assignment. After a student finishes one function, the student can test the program by using the instructor's answers of the unfinished functions. The code from the student and the instructor is compiled and linked together to create the executable. This allows the student to learn programming one step at a time. The instructor's answers are hidden behind the website and the student cannot see the answers.

This system offers many benefits that are unavailable in today's grading systems. First, the web interface greatly simplifies the students' need to manage their coding environments. This is one main advantage of "Software as a Service": users do not have to maintain (install and upgrade) the tools. Second, the instructor can monitor students' progress. By allowing students to use the hidden answers from the instructor, the instructor can understand which part of the assignments are the most difficult (because most students use the instructor's answers for testing the rest of the programs). Third, the system automatically analyzes the students' programs and the students do not need to configure these tools. Presenting the insightful information about a computer program without the student's effort managing the tools is equivalent to presenting the vital information about an airplane to pilots without asking pilots to manage the instruments. The core function of this system has been operational since September 2015 and is performing alpha testing now.

This system does not intend to completely replace the tools and students should still need to learn some tools. Instead, *the system smoothes the learning curve*. A student can benefit from all the tools from the first day. As the student learns more about programming, some tools can be disabled (under the instructor's control) and the student has to manage the tools.

Integrated Tools

This section briefly describes the tools that have already been integrated into this system:

1. `git` for version control
2. `gdb` for showing call stack
3. `Valgrind` for detecting memory errors

Version control is commonly used in software development. A complex program is written in stages, i.e., different versions. Version control is particularly useful for a team project where new features and bug fixes are often created by different team members. Merging these changes by hand is error-prone. This system uses `git` for version control. Students can save different versions, compare different versions, and roll back to an earlier version. Version control benefits students because they can experiment, without the fear of losing their partially working solutions. Even though many books and instructors encourage students to "design carefully, code

correctly, and never write buggy code", debugging is difficult to avoid completely. The tool `gdb` shows the call stack of a program if it terminates abnormally ("crashed"). This system automatically invokes `gdb` when a program crashes, parses the outputs of `gdb`, and shows the exact line where the program crashes. Invalid memory accesses and memory leak are common problems when students learn C programming. This system uses `Valgrind` to detect such errors. We plan to integrate additional tools (test coverage and performance profiling) in the near future.

Related Work

An integrated development environment (IDE), such as Eclipse, is the most popular approach to facilitate the learning process for beginners. An IDE provides many advanced tool-kits, but just like any other software, it needs to be installed, configured, and maintained by programmers. Learning how to use IDE can be overwhelming for beginners, not to mention that many features of an IDE are designed for advanced users. This is exactly the problem we want to solve. Refer to the analogy of pilots learning flying and the instruments, we want students to learning programming without worrying about the development tools. Several websites, such as Cloud9 (https://c9.io/) and Codeanywhere (https://codeanywhere.com/), provide full-featured cloud-based IDE service to save the time of installation and maintenance. However, they are designed for professionals, not the beginners since mastering an online IDE takes as much time as mastering a desktop IDE. Our system retains key advantages as a cloud-based programming environment while reducing the burden to students by providing a much cleaner interface. Another problem is that IDE is controlled by users (students) and instructors have no control of the plugins used by students. Therefore, students may use IDEs with different versions of compilers / libraries / operating systems from the environment used by instructors in grading. The programs written by students may not execute correctly in the different environment and this creates confusion among students. The proposed system eliminates this problem because every student uses the same back end that is virtualized by Docker container.

Codecademy (https://www.codecademy.com/) presents step-by-step coding instructions to beginners in an interactive online programming environment. A similar approach is CS Cycles (http://cscircles.cemc.uwaterloo.ca/), which integrates university-style curriculums, embeds thousands of online auto-gradable practice assignments, and an open source program visualizer [2] [3]. Both of these two websites are intuitive but they provide no support for debugging or managing coding projects. Our system provides the necessary tools and let students efficiently manage their coding projects by using these tools.

There are many existing systems focusing on grading. Douce et al. [4] provide an overview of automatic assessment tools. Edwards [5] proposes test-driven development using an automatic grading tool called Web-CAT (http://web-cat.org) and encourages students to submit their own test cases. Helmick [6] presents Computer Science CourseWare as an integrated environment for courses management. Mimir (https://mimirhq.com/) provides online automated code grading, code quality analysis, and statistics service. These applications consider final submissions, telling students whether their programs are correct. However, students need instant feedback of where, why, and how their programs are wrong while writing programs to achieve efficient learning.

Instead of analyzing complete programs, our system considers the software development process and provides useful information for both students and instructors. The system presents many different views of computer programs and students can understand their programs more deeply. It records students' mistakes during the development process so that instructors can provide additional explanation before the students submit the programs for grading. Figure 1 compares the system with other existing work.

Approaches	**Configuration**	**Easy to Learn**	**Debugging Tools**	**Project Management**	**Embedded Assignments**	**Assignments Grading**
IDE	Desktop	✗	✓	✓	✗	✗
Cloud9	Web	✗	✓	✓	✗	✗
Codecademy	Web	✓	✗	✗	✓	✓
CS Cycles	Web	✓	✓	✗	✓	✓
Web-CAT	Web	✓	✗	✗	✗	✓
Our System	Web	✓	✓	✓	✓	✓

Table 1: Comparison of Different Approaches that Facilitate the Learning Process. The system provides debugging tools, programming project management, and embedded gradable assignments.

System Overview

User Interface and Integrated Tools

Our system starts with a simple user interface. As shown in Figure 2, it has a file tree, file operation buttons, a selection bar for choosing source files, an input widget for entering arguments, two execution buttons: Run and Check Memory Error, and an execution result displaying area.

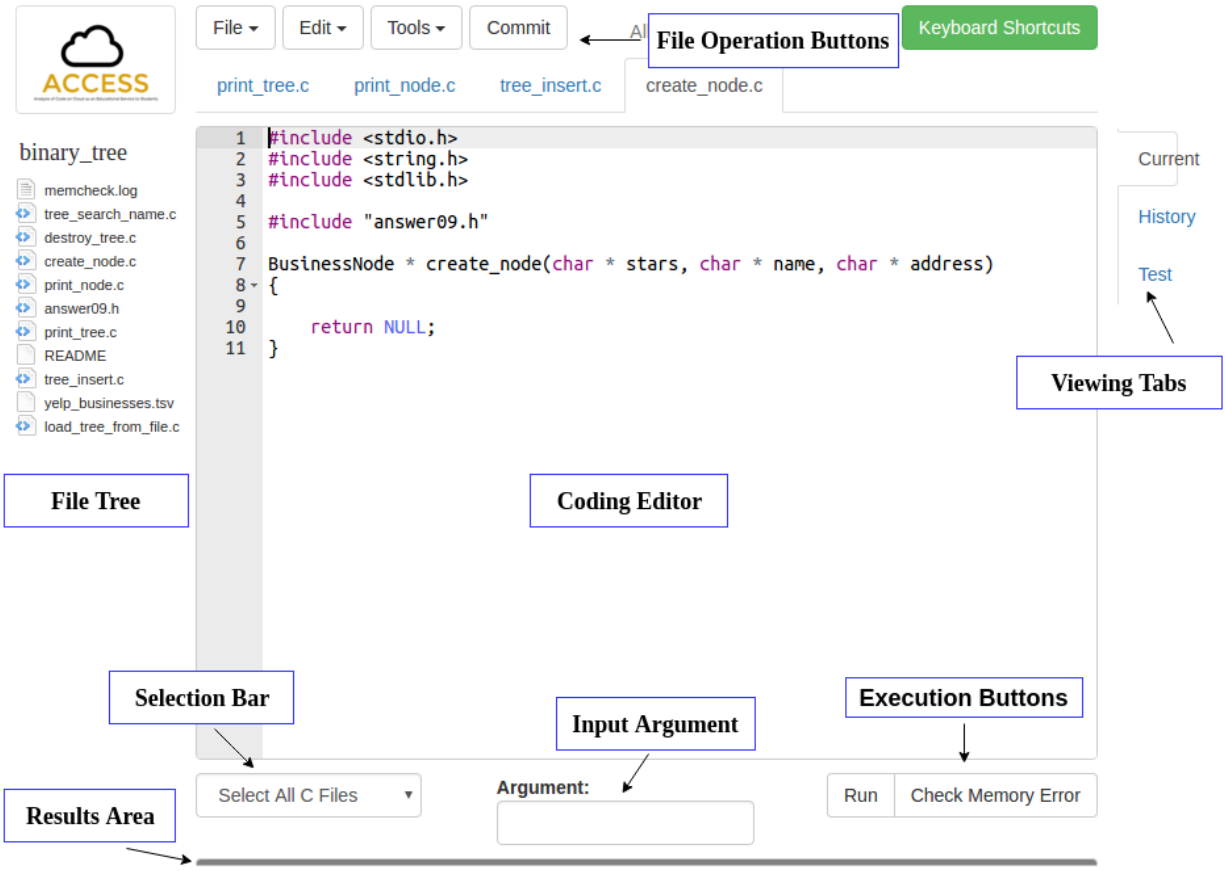

Figure 2: The User Interface of the System

The coding area has three different views: Current, History, and Test, shown as Viewing Tabs in Figure 2. The "Current" tab means editing mode, and a fully-featured open-source editor [7] with multiple file tabs are there for students to write programs. The system supports students to edit multiple files at the same web page, and the editor is customized to autofit different sizes of screens. The editor also supports common keyboard shortcuts and syntax highlight to improve students coding experience. The "History" tab shows all the previous versions of a file. The "Test" tab pops a testing panel for students to modularly test their programs. The latter two viewing modes would be further explained in the following sections.

If a student clicks the "Run" button, the system automatically generates correct configuration, compiles the student's coding project, runs the program, and shows the programming outputs on the results displaying area. If the student's code cannot compile, the system shows the compile errors. If the student's program crashes during execution, `gdb` is automatically invoked. In this case, the system displays the exact line number where the program crashes and highlights the line in the editor, as shown in Figure 3.

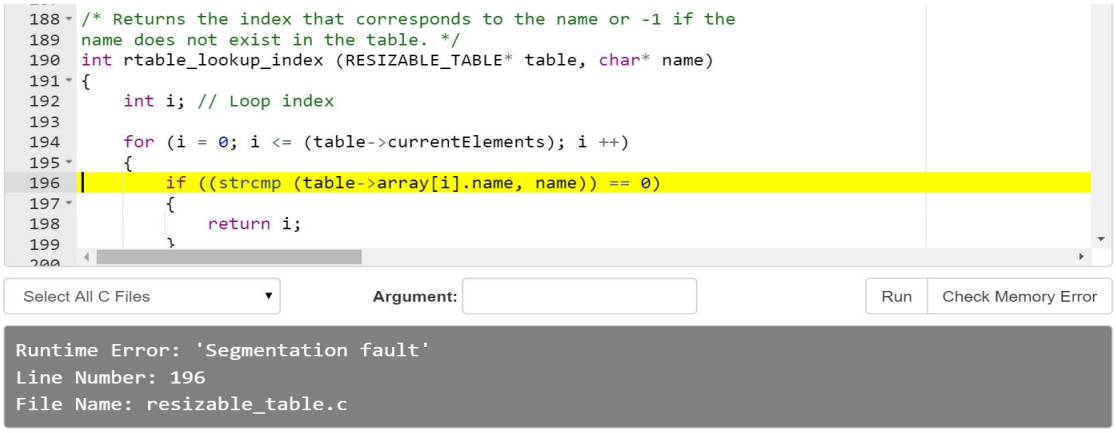

Figure 3: Runtime Error Detection

If a student clicks the "Check Memory Error" button, the system runs the program using `Valgrind` in order to detect potential memory errors. If there is any memory error, the system parses the original verbose outputs of `Valgrind`, eliminating repeating messages and displaying only the necessary debugging information in a concise and understandable way to the student, as shown in Figure 4. The original `Valgrind` output of Figure 4 is more than 3000 of lines.

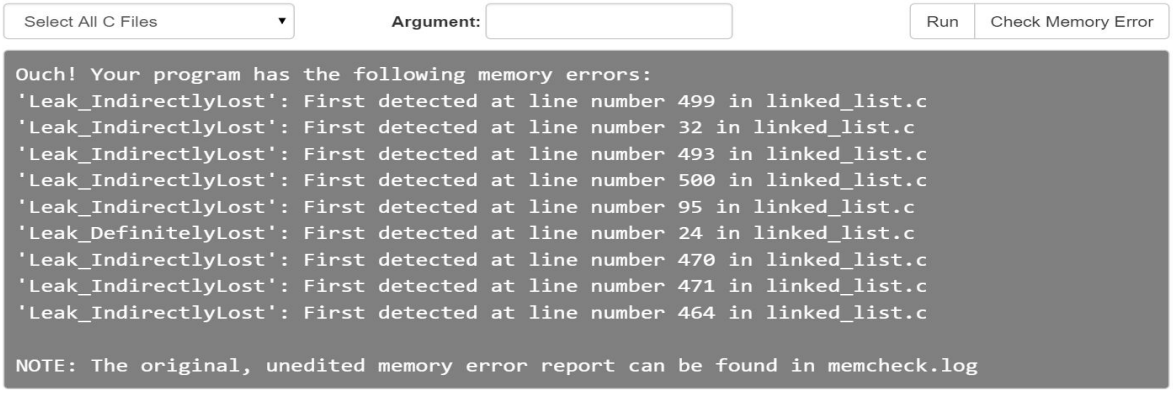

Figure 4: Memory Error Detection

Embedded Version Control

This system has a built-in version control tool. A student can create multiple versions of one file and add comments for each version. In the "History" viewing mode, the right side of the panel displays the available versions of the file and the left side shows the preview of the selected version. As shown in Figure 5 and 6, the student can browse all previous versions. If the student clicks the "Rollback" button, the system restores the selected version. Thus, the student takes the advantage of using version control without any knowledge about the underlying mechanism.

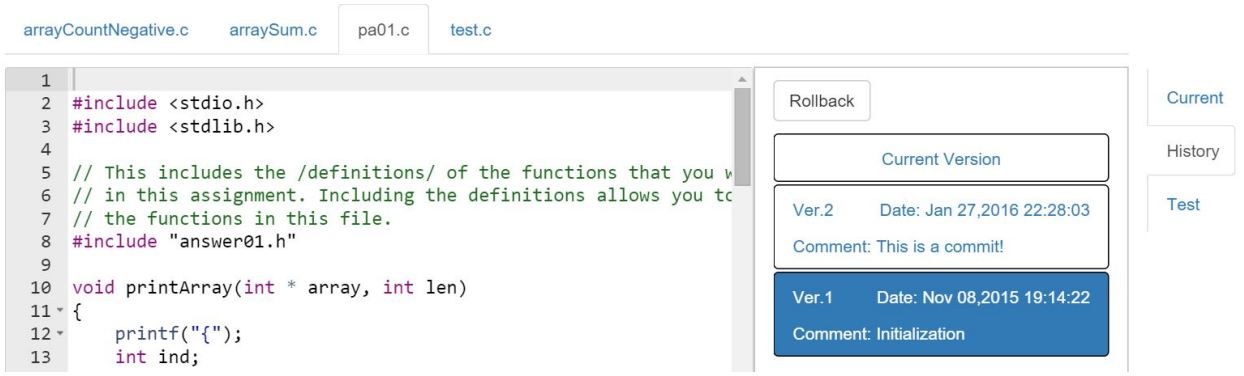

Figure 5: The Initial Version of the File "pa01.c"

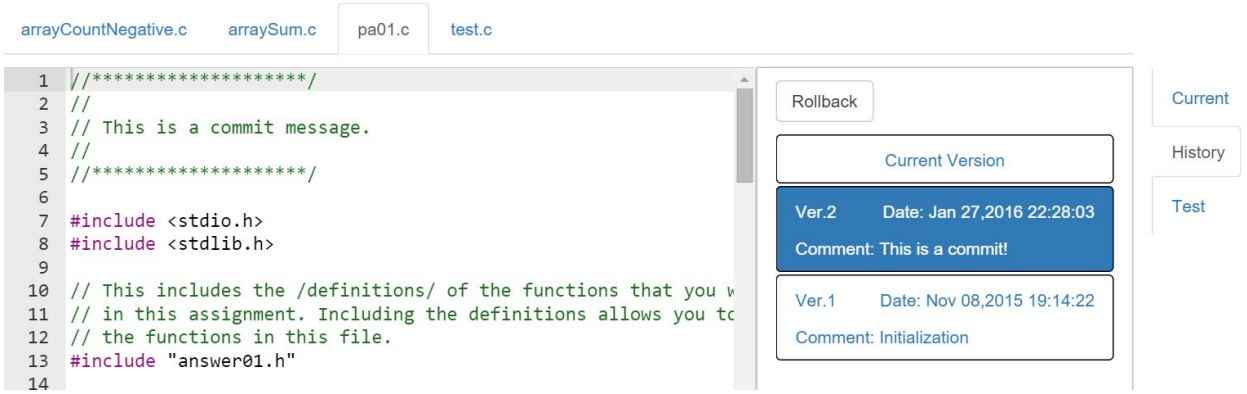

Figure 6: The Second Version of the File "pa01.c". The green comments in the top of the editor are the differences between this version and the initial version. The student can browse multiple versions at the same time.

Modular Programming and Testing

A program consists of many functions. If one function fails, the program fails. In a traditional coding classroom, identifying the failing functions can be difficult. Students are expected to write each function correctly to run their programs. However, without a complete program, testing could be difficult and this could be a frustrating experience. This system provides a new way for testing. The system has correct solutions hidden in the backend. When a student finishes only one part of the program, the program can run and be tested after replacing the unfinished parts by the correct solutions. The student has the freedom to choose which functions are replaced to debug the program. This feature allows students to finish programming assignments step by step, and it brings the following three benefits for students.

1. It quickly identifies the incorrect functions thus assigning partial credits is easier.
2. It allows students to test each function individually before finishing the entire program.
3. It finds out which functions are replaced the most as an indication that students have difficulty completing that part of the program.

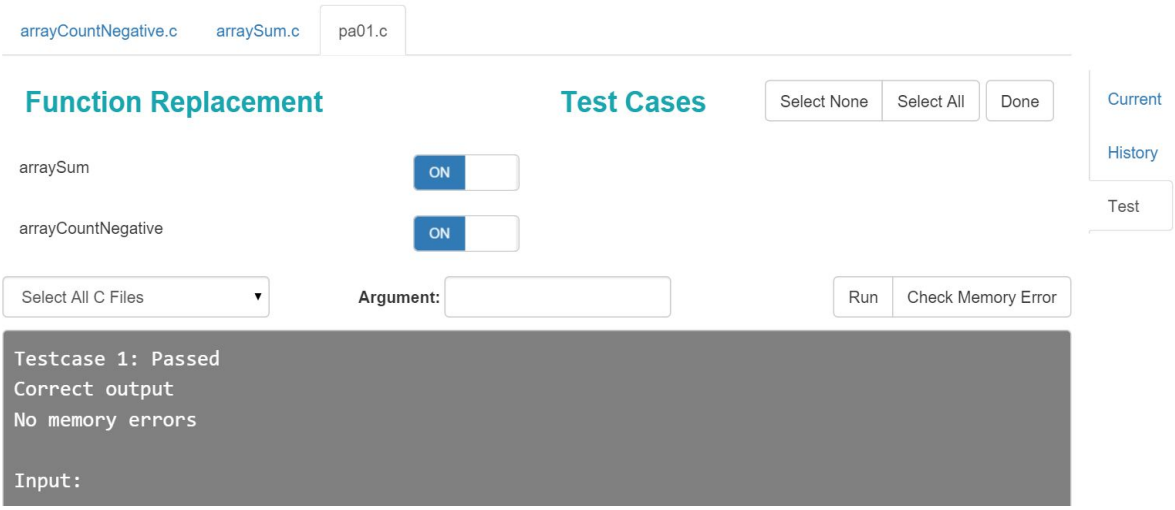

Figure 7: The User Interface for Modularly Testing. Both functions, "arraySum" and "arrayCountNegative", are replaced by the hidden solutions as their switches are on. Therefore, the program is compiled and generates correct results even though neither of them is completed.

As shown in Figure 7, in the "Test" viewing mode, each function is like a switch. The system monitors the development process of students by recording how the switches are toggled, and provide instructors with insights into students' view of programming. As students become more proficient, this feature can be partially or completely disabled by the instructor and students have to write the entire programs.

Assignments Management

The system also contains an assignments distribution hub. Instructors can upload their assignments to our system directly. Alternatively, they can maintain an online repository that links to our system. Furthermore, instructors can configure the resource of each assignment under which each program can use. For example, how much memory, how many cores, and how much storage space are provided for a program. Students can also go to the assignment hub, as shown in Figure 8, select assignments, and click the "Import" button to import assignments to their personal folders. This system is convenient to both students and instructors and thus can create a learning community beyond the boundaries of the classroom.

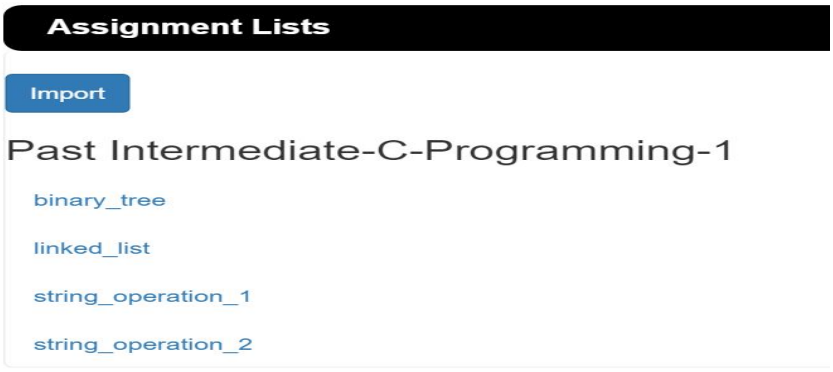

Figure 8: Assignments Hub and Several Embedded Assignment

Code Analysis

Code analysis is another core function of this system. When students run their programs, the system saves the compile information, execution results, memory violation logs, and other outputs for further analysis. The system aims to record every single operation of students, such as checking out an old version or using standard solutions to test. The system is configurable by each individual instructor for generating the desired report of the progress of students and the statistics about common mistakes. According to the reports, instructors can determine which topic they need to elaborate during lectures.

Implementation

The user interface of the system is responsive on various browsers, majorly built under an open source HTML, CSS, and JavaScript framework Bootstrap. Most communications between the client and our server are completed by asynchronous JavaScript (Ajax) for loading data dynamically without refreshing the entire page. We chose Django as the web framework, taking advantages of its user authentication system to save developing time. The system uses MySQL for the database. Each user program is compiled and executed in an independent Docker container. A Docker container is a lightweight virtual machine. The server is currently deployed on Apache Httpd and Ubuntu Operating System.

Evaluation

Overview

The evaluation process is intended to answer three primary research questions.

1. Does the system provide a superior environment for the improvement of debugging skills, beyond options like IDE and Codeacademy?
2. Is the system helpful for intermediate and beginning coders alike?
3. Dose frequency and regularity of using the system influence performance on coding assignments?

The initial evaluation process for the system and its effectiveness will include both analyses of student use of the system and student self-report data. The research population for the current study will consist of approximately 65 students enrolled in an Advanced C Programming course. The students in the course typically have a wide range of programming skill. In order to avoid comparison of non-representative sample sets, we will use several initial class assignments to determine baseline competency in skills. We will explore skill gains among individual students. Since the first implementation of the system alongside a classroom environment involves students in the second programming class, our analysis focuses on seven programming skills essential for this level of students, such as String Modification/Replacement; String Deletion; Text Compression; Array Count; Array Sort/Order; Array Search; Mathematical Operation. Of the seven, Mathematical Operation is the skill least emphasized in the course, but it is important to intermediate and advanced C programming.

Research Design

We will use an A-B-A-B research design to compare the system against traditional methods of coding practice and feedback. Students will be randomly assigned to one of two groups, alternating between experimental and control. Beginning in the second week of classes, and continuing for the next four weeks, both groups will be asked to complete two out-of-class assignments that measure some aspect of programming skills. The assignments are designed to be identical in difficulty and concept. Only one group will be allowed to use the system at a time for each assignment. For example, during the first experiment, Group A will use the system for the assignment in the beginning of the week, and Group B will use the system for the assignment at the end of the week. The following week, the order will switch. This experimental framework will allow comparisons between use of the system while providing some control against possible improvements that would come with the repeated practice. Since the system is designed to provide targeted feedback on code writing, the group alternation is done to avoid disadvantaging Group B, as Group A might perform better on both assignments if they were also first to use the system each week.

Because the use of the system will be voluntary, we expect the students will fall into one of three groups: those who engage consistently with the system for each assignment; those who engage partially with the system; those who ignore the system entirely. We will compare each of these three groups to determine if regular use of the system improves performance on coding assignments, and if the level of engagement is influenced by skill level (i.e., students who begin with demonstrable, advanced programming skills may not engage with the system because they have already acquired the necessary competency for coding and debugging).

As is the case for novices in many fields, a student's self-perception of their ability has a significant influence on their in-class achievement [8]. A student's attitude towards programming can also influence their ability to problem-solve, particularly when working in teams [9]. With these considerations, evaluation of the system includes student self-report feedback. Borrowing from the NSF-funded Student Assessment of Learning Gains project (salgsite.org), we ask students to rate their improvement in coding skills as a result of their use of the system. Students will be asked near the end of the semester to complete an online survey regarding the utility of the system and its ability to improve their programming skills. Students will be asked what specific parts of the system are helpful, along with suggestions for future improvements of the system. Students will also be asked to rate the system's effectiveness at improving the 7 skills emphasized in the course (see appendix for full survey). We will compare student responses in a perception of learning gains with their actual performance in the course and engagement with the system.

Conclusion

This paper presents a system that helps students learn programming. The system automatically invokes many tools for software management and development. By providing the information from these tools without asking students to install, configure, and maintain these tools, our system can take advantage of these tools without adding burden to students. Instructors can understand students' progress while they are doing programming assignments.

Future Improvements

This system has many possibilities for the future improvements. We plan to add functions that allow collaboration and communication among students. The system can help instructors monitor contributions of each student in a group project. Plagiarism detection tools, such as Moss [10], may also be integrated for instructors as they become necessary. Moreover, besides record every single operation of students and program execution settings, it is possible to store copies of students' code to understand their progress. The information will be valuable for future studies on how students learn and debug their programs.

Acknowledgement

We want to thank the AWS Cloud Credits for Research and the Microsoft Azure for Research Program. This project is supported in part by NSF ACI-1535108. Any opinions, findings, and conclusions or recommendations in this materials are those of the authors and do not necessarily reflect the views of the sponsors. This system is built as one of the Vertically Integrated Projects [11] at Purdue University.

References

1. Association for Computing Machinery. ACM Hails New "Every Student Succeeds" Law. https://www.acm.org/media-center/2015/december/essa-epc
2. Pritchard, D., Vasiga, T. (2013). CS circles: an in-browser python course for beginners. *ACM technical symposium on Computer science education,* Pages 591 - 596.
3. P. Guo. Online Python Tutor: Embeddable web-based program visualization for CS education. In these proceedings, 2012. http://pythontutor.com/.
4. Christopher Douce, David Livingstone, and James Orwell. Automatic TestBased Assessment of Programming: A Review. Journal of Educational Resources in Computing, 5(3):4, September 2005.
5. Stephen H. Edwards. Improving Student Performance by Evaluating How Well Students Test Their Own Programs. Journal of Educational Resources in Computing, 3(3):1, September 2003
6. Michael T. Helmick. Integrated Online Courseware for Computer Science Courses. In SIGCSE Conference on Innovation and Technology in Computer Science Education, pages 146–150, 2007.
7. Ace - The High Performance Code Editor For The Web. https://ace.c9.io/#nav=about
8. Chalk, P., Boyle T., Pickard, P., Bradley C., Jones R., Fisher, K. (2003). Improving pass rates in introductory programming. *Proceedings of LTSN-ICS*.
9. McKinney, D., Denton, L. (2005). Affective assessment of team skills in agile CS1 labs: the good, the bad, and the ugly. *ACM SIGCSE Bulletin*, 37(1).
10. Saul Schleimer, Daniel Wilkerson, and Alex Aiken. (2003) Winnowing: Local Algorithms for Document Fingerprinting. *ACM SIGMOD international conference on Management of data, Pages 76-85.*
11. Vertically Integrated Projects, https://engineering.purdue.edu/vip/

APPENDIX - Survey Instrument

5-point Likert Scale, Strongly Disagree to Strongly Agree

1) The System helped me improve my ability to use the following functions in C programming:

    String modify/replace
    String delete
    Text compression

Array count
Array sort/order
Array search
Mathematical Operation

2) Was the system helpful to you: Yes/No

Open Ended:

3) [*If yes*] What about the system did you find helpful?
4) [*If no*] Why was the system not helpful?
5) [All] What would have improved your experience with the system?