CAN USING A FORMAL SYSTEM FOR TRACING COMPUTER PROGRAMS HELP STUDENTS LEARN INTRODUCTORY COMPUTER SCIENCE?

Tom M. Warms, Pennsylvania State University Abington College; Qiang Duan, Pennsylvania State University Abington College; Kavon Farvardin, Pennsylvania State University

Abstract

Teachers of computer science have known that the ability to trace a computer program is in some sense a necessary condition for being able to write original programs. The concept of tracing has not been given much specificity in the literature. In the research reported upon here, a formal concept of tracing is used to present material to an introductory computer science class and, in a computer software representation, is made available to the students as a resource. Neither the tracing method nor the software is made available to a control section. The results of the research suggest that programming students feel that the tracing technique and software are useful tools, and inexperienced students respond more positively to the method and the software than experienced students.

Introduction

Tracing a computer program is an imprecise term that is used to describe the activity of following the statements of the program, step-by-step, and predicting the results of executing the statements. The literature describes experiments that relate the ability of a student to trace programs or program segments accurately to the student's ability to write original programs. One study [1] examined behaviors of students as they tried to solve programming problems, and found that students who accurately formed certain kinds of traces had a high probability of getting the correct answer to a variety of programming-related questions. Conversely, the study found that students who did poorly on tracing had a fragile grasp of basic programming principles, and suggested that an early emphasis on tracing and program comprehension might liberate the student to concentrate on the more creative aspects of programming. Another study [2] suggested that "the combination of tracing and explaining questions, more so than each skill independently correlates highly to code writing skills, supporting the ... notion of a hierarchical development of skills in learning to program."

In the present study, a formal method for tracing the execution of computer programs [3], [4] is introduced by the instructor to an experimental section; a computerization [5] of the system is used for lecturing and is made available to the students in that section as a resource. The study seeks to determine whether or not the student's overall performance in the course is enhanced by this availability.

Method

The experiment was conducted in two sections of the CMPSC 121 course in the Fall 2010 semester at a campus of Pennsylvania State University. CMPSC 121 introduces the fundamental concepts of computer programming and teaches students basic skills for designing and implementing structured programs. This course uses C++ as the programming language and employs the procedural paradigm almost exclusively. This is the first course in a three-course programming sequence for computer science and computer engineering majors at Penn State. There is no computer programming prerequisite for the course. Topics covered in this course include data types, mathematical expressions and calculations, basic I/O and files, control structures, looping structures, user-defined functions, arrays, simple searching and sorting algorithms, and the mechanisms of running, testing, and debugging a program.

There were two sections of CMPSC 121 taught by the same instructor in Fall 2010. The experimental section in which the tracing method was taught and for which the tracing software was made available had 16 students. The control section had 7 students who were not exposed to the tracing method. Students enrolled in the two sections included computer science and computer engineering majors as well as students in a variety of engineering and other majors.

Before taking this course, some students learned programming either by taking courses in high school or college or by teaching themselves. For other students, this was their first programming experience. In the experimental section, the instructor introduced the tracing method at the beginning of the third week of the semester. At that time the tracing software was provided to students by posting it on the class bulletin board for download.

From the time of introduction, the instructor used the trac-

ing software as a tool to explain new concepts. Typically, the instructor used the software to trace through example programs to demonstrate how those programs are executed. By doing that, the instructor also demonstrated how to use the tracing software. Example programs for the following subjects were analyzed by using the tracing software: basic mathematical expressions and calculations, if and if-else statements, while, do while, and for loops, user-defined function calls and returns using both call-by-value and call-by-reference, arrays, and linear search. Students in the experimental section had opportunities to use the tracing method and the software both in class and after class. The instructor also demonstrated how to use the tracing method manually without running the software, and asked students to practice it in a homework assignment.

The same set of topics was taught in both sections. Students in the control section were not exposed to either the tracing method or the tracing software. In order to compare the student performances in the two sections, identical homework, test questions and programming activities were assigned to both sections.

```
// This program prompts the user for three integers
// and calculates and prints the largest and smallest
#include <iostream>
using namespace std;
void getExtremes(int, int, int, int &, int &);
int main()
{
      int x1, x2, x3, large, small;
      cout << "Enter three integers: ";
      cin >> x1 >> x2 >> x3;
      getExtremes(x1, x2, x3, large, small);
      cout << "The largest integer is  " << large << endl
           << "The smallest integer is " << small << endl;
      return 0;
}
void getExtremes(int x, int y, int z, int &lge, int &sml)
{
      lge = x;
      sml = x;
      if (y > lge)
            lge = y;
      else if (y < sml)
            sml = y;
      if (z > lge)
            lge = z;
      else if (z < sml)
            sml = z;
}
```

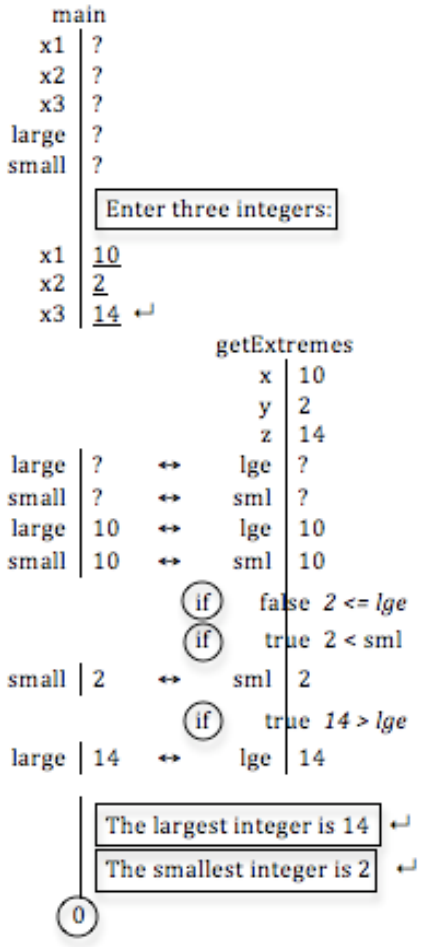

Figure 1. A Program to Prompt the user for Three Integers; Calculates and Prints the Largest and Smallest; using a User-Defined Function for the Calculation

Tracing

The tracing method is described in detail by Warms [3] and Warms & Drobish [4]. According to the method, the student writes down in a specified manner the result of the computer carrying out each of the executable statements in a program. While tracing any program statement in this formal system, the results of executing previous statements are always available. Uniquely, this method provides notations for tracing programs that contain more than one function. There are notations, for example, for statements that transfer control to other functions using call-by-value or call-by-reference, and for statements that return control to the calling function. Figure 1 shows a program that prompts the user for three integers and, using a function, calculates and prints the largest and smallest of the integers.

The Software

The software program, called RandomLinearizer, presents a list of programs to trace. Once one has been chosen, the selected program appears in the center of three panels, followed by an input set. The left panel then contains a randomized list of the elements that make up the trace of the program from that input set. The right panel is initially empty. The student is expected to click on the trace steps in the correct order so that the complete trace unfolds in the right panel. At the same time, the contents of the console window are updated at the bottom of the middle panel.

Figure 2a shows the setup for the trace of a program that uses a function to calculate the sum of two numbers, while Figure 2b shows the completed trace. When the user makes an error, RandomLinearizer displays a window that provides the location and, at times, the nature of the error. Figure 3 shows a diagnostic comment as a user makes a wrong choice in tracing a program.

Student Reaction to the Tracing Method and the Software

Figure 4 contains a questionnaire that was administered to students in the experimental section at the time of the final examination. Questions 1–10 dealt with the tracing method, 11–17 with the software, and 18–20 with the students' programming background. Questions 18–20 were administered to students in the control section as well as in the experimental section, and the responses in both sections were used to determine whether the student was inexperienced or experienced in computer programming. Table 1a shows the results for the responses of the entire experimental section on questions 1–10; Table 1b shows the same results for questions 11–17.

The results of the questionnaire show that students in the experimental section overwhelmingly responded positively toward the method and the software. The students agreed emphatically that tracing helped them learn the material of the course (question 2: Mean = 4.00 on a scale of 1 to 5). When the course material was broken down into general topics, agreement was strongest that the software was helpful when the students were trying to learn how to write programs with functions (question 15: Mean = 4.00). The students responded strongly in the negative to questions which stated that tracing didn't help the student at all (question 4: Mean = 1.64), and the software was not at all helpful (question 14: 1.50).

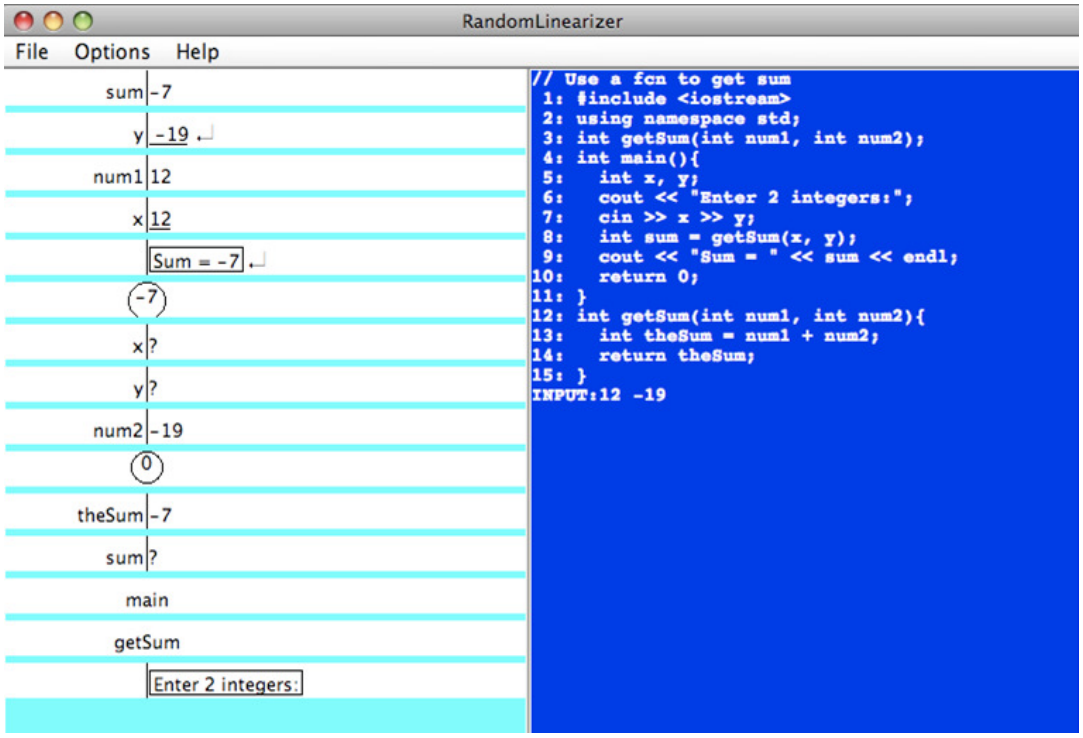

Figure 2a. Setup for Tracing a Program that uses a Function to Calculate the Sum of Two Numbers

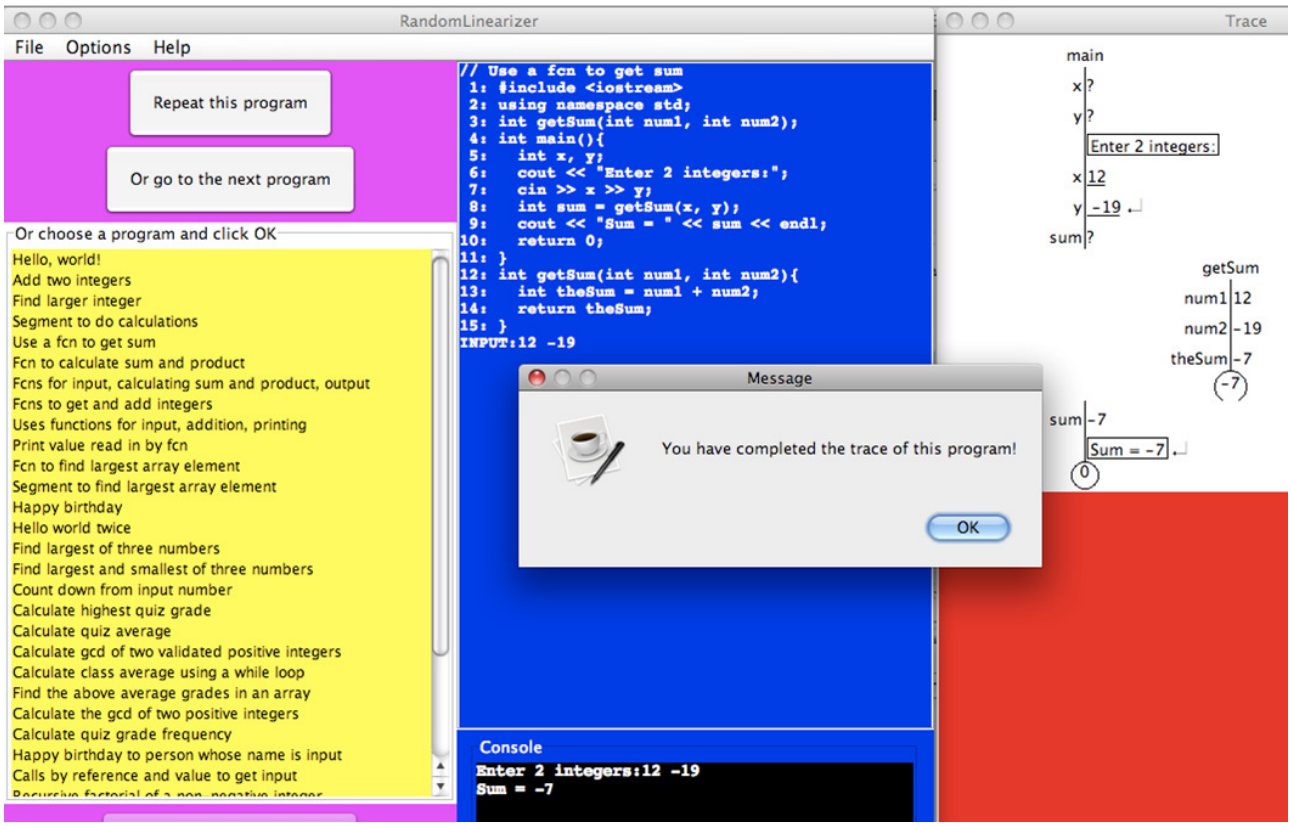

Figure 2b. The Completed Trace for the Program of Figure 2a

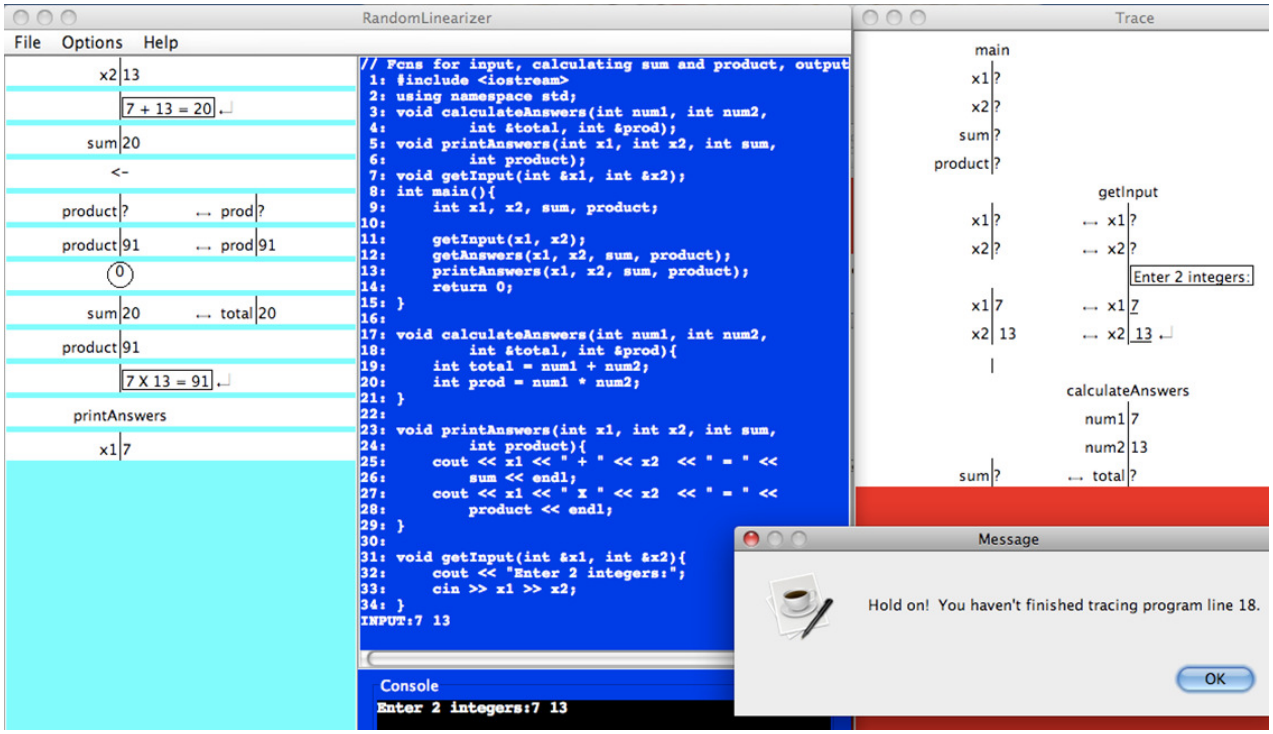

Figure 3. A Diagnostic Comment Appears when the User Selects the Wrong Trace Step

Student ID _____

Please fill in this questionnaire even if you decided not to participate in the research on the tracing method. If you decided not to participate your responses will not be used in research.

CMPSC 121
Tracing

Please encircle the degree to which you agree with each statement
(1 = Disagree, 5 = Agree).

1. I understand tracing 1 2 3 4 5
2. Tracing helped me learn the material of this course 1 2 3 4 5
3. Tracing is easy to learn 1 2 3 4 5
4. Tracing didn't help me at all 1 2 3 4 5
5. Tracing helped me follow how a program is executed 1 2 3 4 5
6. Tracing helped me understand C++ 1 2 3 4 5
7. Tracing helped me write programs 1 2 3 4 5
8. Tracing is confusing 1 2 3 4 5
9. I'm glad I learned how to trace programs 1 2 3 4 5
10. It was hard to learn tracing 1 2 3 4 5
11. At times, I used the software to help me understand material 1 2 3 4 5
12. The software was helpful to me when I was learning elementary programs 1 2 3 4 5
13. The software was helpful to me when I was learning how to write programs with loops 1 2 3 4 5
14. The software was not at all helpful 1 2 3 4 5
15. The software was helpful to me when I was learning how to write programs with functions 1 2 3 4 5
16. The software was easy to use. 1 2 3 4 5
17. The software was helpful to me when I was learning how to write programs with arrays 1 2 3 4 5
18. This is the first programming course I have taken in college T ___ F ___
19. I took one or more programming courses when I was in high school T ___ F ___
20. I learned how to program on my own in high school T ___ F ___

Please use this space and the other side for any comments you may have about tracing and the software.

Figure 4. Questionnaire Administered to the Students in the Experimental Section of the Course at the Time of the Final Examination

Table 1a. Mean and Standard Deviation for Questions 1–10 of the Questionnaire

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Mean	4.14	4.07	3.64	1.64	4.14	3.79	3.5	2.93	3.93	3.14
N	14	14	14	14	14	14	14	14	14	14
SD	1.027	0.730	0.929	0.745	0.663	0.802	1.286	1.207	0.829	1.099

Table 1b. Mean and Standard Deviation for Questions 11–17 of the Questionnaire

	Q11	Q12	Q13	Q14	Q15	Q16	Q17
Mean	3.00	3.50	3.64	1.50	4.00	3.71	3.64
N	14	14	14	14	14	14	14
SD	1.414	1.225	1.082	1.019	0.784	0.914	0.929

Response to Method and Software by Prior Experience in Experimental Section

Some of the students in both sections had previously taken computer science courses in high school or college, or had taught themselves how to write programs. Table 2a shows the results of questions 1–10 for the experimental section, broken down by the experience of the user; Table 2b shows the results for questions 11–17.

Table 2a. Mean and Standard Deviation for Questions 1–10 of the Questionnaire as a Function of Experience

Experienced		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
yes	Mean	4.00	4.13	3.50	1.63	4.13	3.50	3.63	3.13	3.75	3.00
	N	8	8	8	8	8	8	8	8	8	8
	SD	1.195	0.835	1.069	0.744	0.641	0.756	0.916	1.126	0.886	1.069
no	Mean	4.33	4.00	3.83	1.67	4.17	4.17	3.33	2.67	4.17	3.33
	N	6	6	6	6	6	6	6	6	6	6
	SD	0.816	0.632	0.753	0.816	0.753	0.753	1.751	1.366	0.753	1.211

Table 2b. Mean and Standard Deviation for Questions 11–17 of the Questionnaire as a Function of Experience

Experienced		Q11	Q12	Q13	Q14	Q15	Q16	Q17
yes	Mean	2.63	3.38	3.88	1.88	4.00	3.75	3.50
	N	8	8	8	8	8	8	8
	SD	1.598	1.408	0.991	1.126	0.926	0.886	1.069
no	Mean	3.50	3.67	3.33	1.00	4.00	3.67	3.83
	N	6	6	6	6	6	6	6
	SD	1.049	1.033	1.211	0.632	0.632	1.033	0.753

Students in the experimental section responded positively toward the method and the software regardless of their experience; the inexperienced students among them responded even more positively. The difference in means for the inexperienced and experienced students trended toward significance in their responses as to whether tracing helped them understand C++ (question 6) or was not at all helpful (question 14). The two questions are, in a sense, inverses of one another: question 6 deals positively with the tracing method and question 14 negatively with the software. The direction of the difference in both cases suggests that inexperienced students felt that the method and software were more helpful than did the experienced students.

Conclusions and Future Work

No more than very tentative conclusions can be reached on the basis of this experiment that was carried out with small sample sizes. On the whole, though, students felt that tracing was useful and the tracing software was a useful program. More specifically, a statistically significant result showed that inexperienced students were even more positively disposed to the tracing technique and software than the experienced students. A comparison between the control and experimental sections involved even smaller sample sizes; therefore, no conclusions, however tentative, could be drawn.

Elsewhere [6] it has been conjectured that inexperienced students may feel intimidated by experienced students. In addition, it has been found that the best predictor of success in introductory computer science courses is students' comfort level [7]. Repeating this study on a larger scale will provide better answers to the questions of whether or not a tracing is a tool that appeals to all students; whether or not it helps inexperienced students feel less intimidated by experienced students; and, whether students who learn in an environment in which tracing is taught learn the material better than others who learn in a non-tracing environment.

Acknowledgments

Kavon Farvardin worked on this project within the Penn State Abington ACURA undergraduate research program.

References

[1] Lister, R., Adams, E., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M. et al. (2004). A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bulletin*, 36(4), 119-150.

[2] Tan, G. & Venables, A. (2010). Wearing the assessment 'BRACElet'. *Journal of Information Technology Education: Innovations in Practice*, 9, IIP25-34.

[3] Warms, T. M. (2005). The power of notation: modeling pointer operations. *ACM SIGCSE Bulletin*, 37(2), 41-45.

[4] Warms, T. M. & Drobish, R. (2007). Tracing the execution of computer programs – report on a classroom method. *Proceedings of the ASEE Mid-Atlantic Section Conference*. Newark, NJ.

[5] Warms, T. M. (2010). Using the tracing method and RandomLinearizer for Teaching C++. *Proceedings of the FECS'10 International Conference on Frontiers in Education: Computer Science and Computer Engineering*, (pp. 16-22). Las Vegas, NV.

[6] Holden, E. & Weeden, E. (2005). Prior Experience and New IT Students. *The Journal of Issues in Informing Science and Information*, 2, 189-204.

[7] Wilson, B. C. & Shrock, S. (2001). Contributing to success in an introductory computer science course: a study of twelve factors. Proceedings of the SIGSCE symposium, (pp. 184-188). Charlotte, NC.

Biography

TOM M. WARMS is a faculty member in the department of computer science and engineering at Penn State Abington, where he has taught courses in mathematics, computer science, information sciences and technology, philosophy, and linguistics. He received S.B. and M.S. degrees, both in mathematics, from the Massachusetts Institute of Techology and New York University, respectively, and his Ph.D. in formal linguistics and mathematical logic from the University of Pennsylvania. He has published papers in pattern recognition, psycholinguistics and computer science pedagogy. Dr. Warms may be reached at t1w@psu.edu

QIANG DUAN is a faculty member in the department of information sciences and technology at Penn State Abington. He received his Ph.D. in electrical engineering from the University of Mississippi, and holds an M.S. in telecommunications and a B.S. in electrical and computer engineering. His teaching areas include computer networks and telecommunications, computer programming, system analysis and design, and network security. He has published papers in data communications and computer networking, network virtualization and the Internet, Service-Oriented Architecture and Web services, and Grid and Cloud computing. Dr. Duan may be reached at qxd2@psu.edu

KAVON FARVARDIN is an undergraduate student majoring in computer science at the Pennsylvania State University. He may be reached at kff5027@psu.edu