The Influence of Problem Solving Abilities on Students' Performance on Different Assessment Tasks in CS1

Alex Lishinski
Michigan State University
College of Education
East Lansing, MI
lishinsk@msu.edu

Aman Yadav
Michigan State University
College of Education
East Lansing, MI
ayadav@msu.edu

Richard Enbody
Michigan State University
Department of Computer Science and Engineering
East Lansing, MI
enbody@cse.msu.edu

Jon Good
Michigan State University
College of Education
East Lansing, MI
goodjona@msu.edu

ABSTRACT

Previous research has suggested that cognitive tests, including instruments seeking to measure problem solving, are significant predictors of students' programming performance. This paper seeks to expand upon this previous research by using a more theoretically grounded approach to measuring problem solving as a means of predicting performance in an introductory undergraduate programming course. Programming course performance has typically been measured by overall course grades; however, in this paper we used a more fine-grained approach to measuring student programming performance. Specifically, we utilized different types of course assignments (projects and tests) to measure programming outcomes. Results from this study indicate that problem solving ability significantly correlates with performance on programming assignments, but does not correlate with performance on multiple-choice exams.

Keywords

CS1, Assessment, Problem Solving

1. INTRODUCTION

Problem solving is an important set of cognitive skills that are highly valued in education as a set of skills that are applicable to many different academic disciplines [8]. Researchers in computer science education have long been interested in the relationship between problem solving skills and programming abilities [16, 15]. Previous research has suggested that there exists some connection between the generic set of skills that constitute problem solving, and the specific set of skills that is involved in solving programming problems; however, previous research is ambiguous about this relationship.

Previous research on the cognitive processes involved in learning to program has considered generic problem solving skills to be a sort of *telos* of programming instruction [12]. Generic problem solving skills are seen as emerging from the specific declarative and procedural knowledge that is acquired through learning to program, as one moves from novice to expert [16]. However, the existing research has generally not presented a theoretically well-grounded picture of problem solving. Without such a framework, it is difficult to justify a choice of instrument, much less any causal inferences [16].

The other side of the relationship between programming and problem solving is the notion that problem solving ability is predictive of success in programming [15]. However, there is limited research on using problem solving as a predictor of programming performance, compared to the research that has examined improved problem solving abilities as an outcome [19]. Nevertheless, the studies that have looked at problem solving as a predictor are located within a much more extensive body of literature on predictors of programming performance [5]. This body of research presents a relatively uniform picture of the positive predictive value of different sorts of cognitive measures (for example: the embedded figural relations test, the iconic pattern recognition test, and various other reasoning tests) that could reasonably be classified alongside problem solving as measures of "thinking skills" [9, 11, 4]. These results suggest that a measure of generic problem solving would likely have a similar relationship with programming performance.

Type of outcome measure is one element of the relationship between problem solving and programming that has not been substantially explored. Previous studies on predictors of programming ability have generally not discussed the implications of the choice of outcome measure. The outcome typically used is the grade obtained in a programming course or the score on a programming exam. The choice of course grades and exam scores as outcome measures has generally not been described in detail nor justified [15, 4, 22]. This lack of attention to choice of outcome measure persists de-

SIGCSE '16, March 02 - 05, 2016, Memphis, TN, USA

ISBN 978-1-4503-3685-7/16/03…$15.00

DOI: http://dx.doi.org/10.1145/2839509.2844596

spite evidence that different types of assessment can reflect student understanding and ability to substantially different degrees [20]. In particular, research has suggested that the use of multiple choice as opposed to constructed response assessments can greatly affect the validity of assessment in computer science courses [3]. The choice of outcome measure in studies of predictors of programming performance, therefore, becomes critically important.

The focus on the type of assessment draws its significance from the notion of a hierarchy of programming skills, which ranks programming related competencies from easy to difficult. Research into such a hierarchy is relatively new [14] and the results have suggested that a hierarchical model of programming skills is a viable one [17, 21, 13, 18]. Drawing from this research, this study attempted to account for the connection between problem solving ability and programming outcomes in such a way that is consistent with the hierarchical model.

In this study, we examined the relationship between CS1 students' problem solving skills and their programming outcomes as measured by programming projects and multiple choice exams. In order to assess problem solving skills, we used items from the problem solving section of the Programme for International Student Assessment (PISA). These items were chosen because the PISA test focuses on generic, content-agnostic problem solving and because they have not been previously used to examine the relationship between problem solving and programming performance. Furthermore, the PISA test is a well tested and validated instrument based on a theoretical foundation of problem solving, making it ideal for this study.

The contributions of this work include:

- We found that there was a significant connection between problem solving ability and programming performance as measured by grades on programming assignments.
- We found no significant connection between problem solving ability and programming performance as measured by grades on multiple choice exams.
- We found a significant, but moderate, statistical connection between student performance on multiple choice exams and programming project performance, consistent with hierarchical accounts of programming knowledge.
- We found that the variance in programming project performance is uniquely explained by problem solving performance, which suggested that problem solving ability contributes to programming skill at the highest level of the hierarchy of programming skills.
- We piloted the use of an assessment of problem solving that had not previously been used in CS education, which addresses the need for a generic measure that is theoretically well-grounded.

2. THEORETICAL BACKGROUND

Fredericksen (1984) presented a review of the cognitive theoretical framework for problem solving[8]. Drawing largely from Newell and Simon's (1972) account of problem solving, Fredericksen laid out the prevailing key theoretical elements. The first key element Fredericksen discussed was *Problem representation*, which includes the notion of a *task environment* and a *problem space*. Problem representation includes the structure of the problem and the information and concepts that make up the problem definition. Another key element discussed by Fredericksen was problem solving procedures, which included *means-end analysis*, *hypothesize and test*, and *pattern recognition*. Means-end analysis is the process of comparing the present state to the goal state, and devising a series of subgoals to transform the current state into the goal state. Hypothesize and test, drawn from Simon (1980), is the process of looking for confirmatory (and disconfirmatory) evidence for suppositions about the task environment. Lastly, pattern recognition, drawn in this context from studies by Simon and Chase (1973) and Reitman (1976) on chess and go players, refers to the ability to combine disparate elements of the problem space into discrete units for more efficient mental processing.

2.1 Programming and Problem Solving

Previous research on the relationship between problem solving and programming has largely focused on the potential causal relationship between instruction in programming and generic problem solving ability, and less on the predictive power of generic problem solving ability on programming course outcomes. Palumbo (1990) reviewed the literature on the potential causal relationship extensively, and found that overall, the causal connection between problem solving and programming was not well supported empirically[16]. However, certain instructional factors, such as greater programming exposure and deliberate instruction in metacognitive strategies, have been found to influence problem solving.

Research on problem solving as a predictor of programming performance has been less common. Many studies have examined predictors of programming performance based on cognitive abilities of various sorts (see for example [9, 11, 4]) and results have been generally been significant. Fewer studies have examined problem solving *per se* as a predictor. Mayer (1986) examined problem solving as a predictor of programming performance, and vice versa[15]. The results of that study highlighted the significance of problem solving as a predictor. However, the problems chosen to indicate problem solving ability in his study were math word problem translation tasks, which is not aligned with the framework adopted in this study of problem solving as a generic, content-agnostic skill set.

2.2 Hierarchy of Programming Skills

The BRACElet project was a series of workshops and associated studies conducted between 2004 and 2010 with the goal of understanding how students learn various programming skills. The studies established a number of results related to the establishment of a hierarchy of programming skills. Lopez et al. (2008) formally established the hierarchy of skills, developing a model of programming skills that included basics like syntax at the low end, code tracing in the moderate level, explaining at an upper moderate level, and code writing as the highest level [14]. Other results have corroborated the existence and ordering of the hierarchy; Philpott et al. (2007) found that code tracing failures predicted code explaining failures [17].

Additional previous research has empirically examined the relationships between these higher and lower level program-

ming skills. Venables et al. (2009) found relationships between code-writing tasks and lower level tracing and explaining tasks that were consistent with the hierarchical model [21]. Lister et al. (2009) similarly looked at the relationship between these dimensions, finding significant relationships between tracing, explaining and writing [13]. Porter et al. (2014) examined clicker data as a predictor of student performance, and found that the clicker data was equally predictive for multiple choice and code-writing questions, which is consistent with the hierarchical model [18].

2.3 Outcome Measures: Types of Assessment

Previous research on types of assessment has distinguished between multiple choice (MC) assessments and constructed response (CR) assessments. This prior research has shown a significant difference between student results on these two types of assessment, with MC assessments being more suited for assessing students' understanding of factual content with unambiguous correct answers, and CR assessments being better suited for measurement of students' thinking and problem solving processes. Azalov et al. (2004) examined the differences between student results on MC and CR assessments in an undergraduate introductory CS course and found that students' scores on a CR assessment predicted final course grades to a much greater extent than an analogous MC assessment [3]. Kuechler and Simkin (2003) also examined the relationship between multiple choice questions and code-writing questions in an introductory programming course and found significant, but moderate, connections between MC questions and code-writing questions, which is consistent with the hierarchical model [10].

Therefore, the purpose of this study was to examine the relationship between problem solving and different types of assessments (multiple choice exams vs. programming projects) in order to establish whether there is a relationship with one or both. The next section discusses the methodology and data collection to address the research questions.

3. METHODS

Participants: 41 undergraduate students enrolled in a CS1 class at a large midwestern university participated in the study. The sample included 29 males and 9 females (3 non-responses) with an average GPA of 3.19.

Setting: Participants were recruited from an online section of a CS1 course at a large Midwestern University in the Spring Semester 2015. The online course followed a standardized format consisting of the use of pre-recorded video lectures, in lieu of in-person class meetings, to cover the course content. The course exams were conducted in-person and the course used Python as the programming language.

3.1 PISA Problem Solving Framework

The PISA problem solving items used in this study were based on an account of problem solving as generic and content-agnostic. The authors of the PISA problem solving framework contended that a problem is essentially a situation where a novel strategy is required to arrive at a solution [2]. The content-agnostic nature of problem solving thus emerges from the notion that previous knowledge must not be directly applicable in problem solving. The authors contrasted a problem with an exercise, which is a situation that allows for the application of prior knowledge and previously used strategies to generate a solution.

To develop the problem solving items, the authors first developed a problem solving theoretical framework drawn in large part from Mayer (1992). The PISA framework centers around the following definition of problem solving ability: *An individual's capacity to engage in cognitive processing to understand and resolve problem situations where a method of solution is not immediately obvious.* The framework lists deductive, inductive, analogical, and combinatorial as the primary types of reasoning that are used to solve the PISA problem solving tasks. Furthermore, the framework contains a list of cognitive processes involved in problem solving, which include:

- *Exploring and understanding* the information associated with the problem.
- *Representing and formulating* the problem with various representations and formulating hypotheses about the relevant factors and relationships.
- *Planning and executing*: devising a plan, setting goals and subgoals and executing the steps of the plan.
- *Monitoring and reflecting*: monitoring progress, reacting to feedback, reflecting on the solution, the strategy used, or information associated with the problem.

This framework of the processes involved in problem solving maps neatly onto the account given by Fredericksen (1984). Exploring and understanding the problem corresponds to Newell and Simon's notion of problem representation. Representing and formulating the problem corresponds to pattern recognition and hypothesize and test strategies. Planning and executing corresponds to means-end analysis and hypothesize and test strategies.

The NCES (National Center for Education Statistics) technical report (2007) [6] on the assessment of problem solving in two major international assessments (TIMSS and PISA) assessed the validity of the approaches to problem solving used in these two assessments. Drawing from problem solving theory, the authors of the report identified a set of factors that define problem solving, which largely align with the framework used to develop the PISA problem solving assessment. In particular, whether or not a strategy is easily found is key, according to the authors. If a strategy that the individual has previously used in similar situations works in this situation, then it cannot be said to be an instance of problem solving. The NCES report therefore corroborated the theoretical approach to problem solving used by the authors of the PISA problem solving items.

The PISA problem solving items are divided into 3 subtypes. Four items are termed *Decision Making*; given the constraints of a problem situation, the student must select which of a number of outcomes are acceptable or optimal given those constraints. Four items are termed *System Analysis and Design*; given some examples or information about the way that a system is structured, the student must predict how a system will behave in a given situation, or determine what will satisfy the relations and constraints of the system. The remaining two items address *Troubleshooting*; given a situation in which certain characteristics of how a system should behave are described, the student must identify what is causing a given problem.

To measure participants' problem solving skills, we selected items from the released problem solving items from

the 2003 PISA test (see Appendix A for example item) [1]. The released items consisted of 10 distinct problem situations, each of which was accompanied by 1-3 questions. Of the 10 released items, 4 items were chosen for this study. Of the 4 items, two were systems analysis and design, one was decision making, and one was troubleshooting. The items were chosen on the basis of difficulty level and skill relevance. Specifically, item difficulty was identified using the percentage of students who answered the items correctly on the 2003 PISA test. Skill relevance was selected based on the framework of computational thinking. Specifically, items were chosen that were judged to best link generic problem solving with the more generically applicable elements of computational thinking, such as algorithms, pattern recognition, and debugging.

3.2 Study Design and Data Collection

The course grade consisted of multiple choice midterm exams that included mostly code tracing and syntax questions, a cumulative final exam with the same format, programming projects, and weekly lab assignments (see Appendix B for archive of exams and programming projects). The programming projects allowed students to write a substantive piece of code to meet a set of specified assignment requirements in service of a coherent larger goal. For example, one project required students to write a program to open and process text files to extract particular information. These programs were scored by a teaching assistant using a multidimensional rubric based upon the requirements for each assignment.

Of the assignment types used in the course, we chose to focus on the multiple choice exams and programming projects for two main reasons. These two assignment types were the most significant components of students' overall grade, and the programming projects were judged to be the best exemplar of a code-writing assignment in the course.

3.3 Data Analysis

To analyze the data, each of the PISA items were weighted and scored individually in order to create a composite problem solving score for each participant. Specifically, individual question results from actual PISA administrations were used to identify difficulty level of each item and more difficult items were weighted more heavily than the easier items in participants' composite problem solving score. For example, the library system question awarded 1 point to students who correctly answered the initial 1 blank question and a total of 4 points to students who correctly answered the flowchart design question, which had 8 blanks.

To analyze the connection between the problem solving assessment and course outcomes, scores on the programming projects were combined and used as one measure of course performance, and scores on the 2 midterm exams were combined and used as a second measure of course performance. Correlation and regression analysis were used to examine these connections. To avoid bias in the coefficient estimates, students who recorded a zero, indicating that no assignment was submitted, for any of the programming projects or either of the exams were excluded from the data analysis.

4. RESULTS

After removal of those students with missing data, the final analysis included 28 students who completed all of programming projects and exams. The distribution of student scores on programming projects was significantly negatively skewed, and the distribution of students' scores on the problem solving assessment deviated significantly from normality. In order to meet the assumptions of the Pearson product-moment correlation coefficient and linear regression, both variables were transformed using the standard square root transformation. The square root transformation resulted in an approximately normal distribution of values for both variables. The distribution of total exam scores satisfied the normality assumption without transformation. Cumulative GPA was also included in the analysis, and the distribution was negatively skewed, so it was transformed to approximate normality with a natural log transformation.

4.1 Correlation Analysis

The Pearson product-moment correlation coefficient was calculated for the relationships between problem solving score, total exam score, and total project score (see table 1). The results of these correlations shed light on the connection between problem solving and the 2 course outcomes. The relationship between problem solving and programming project scores was moderately strong and significant (r = .4099, p = .0152), but the relationship between problem solving score and exam grade was insignificant (r = -.0088, p = .5176). However, exam and project scores correlated substantially with one another (r = .427, p = .0117), which suggested that, while the two course assessments were related, problem solving ability accounts for performance on programming projects above and beyond exam scores.

4.2 Regression Analysis

To further explore these results, linear models were created, regressing total project scores on problem solving scores and total exam scores, both individually and in combination. An R^2 value of .1824 (Adj: .1509) was observed for the regression of project score on exam score, and an R^2 value of .168 (Adj: .136) was observed for the regression of project score on problem solving score. The combined model had an R^2 value of .3534 (adj: .3017). The ΔR^2 for both variables was calculated for the combined model (see table 2), with a ΔR^2 of .1711 being observed for problem solving scores, and a ΔR^2 of .1855 being observed for combined exam scores. The η_p^2 value of 0.21 for problem solving indicates a medium effect size for problem solving as a predictor. This pattern of results supports the notion that MC and CR assessments in this course are hierarchical, because problem solving only explains performance on higher level skills, but it makes a unique contribution to this explanation. The knowledge assessed by the multiple choice test (tracing, comprehension, etc.) is, therefore, foundational for the skills assessed by the programming projects (writing code). To put it another way, the lower level skills are necessary but not sufficient for the upper level skills.

Previous data from this CS1 class showed that the best predictor of student success in this course is cumulative GPA [7]. For this reason, the connections between cumulative GPA, the two course outcome scores, and problem solving scores were examined. Of particular interest is the relationship between problem solving scores and GPA, because we wanted to know whether the correlation between problem solving and programming project scores was independent or subsumed by another known covariate. The connection between GPA and course outcomes was verified by the fact that

	Cum. GPA (log)	Problem Solving (sqrt)	Prog. Project (sqrt)	Exam Score
Cum. GPA (log)	1.00	0.18	0.36	0.36
Problem Solving (sqrt)	0.18	1.00	0.41	-0.01
Prog. Projects (sqrt)	0.36	0.41	1.00	0.43
Exam Score	0.36	-0.01	0.43	1.00

Table 1: Correlation Matrix

	SSR	df	η_p^2	ΔR^2	SSE	SST
(Intercept)	0.30	1.00	0.00		81.72	126.39
Problem Solving	21.62	1.00	0.21	0.17	81.72	126.39
Exam Score	23.44	1.00	0.22	0.19	81.72	126.39

Table 2: Model change with Exam Score and Problem Solving

	SSR	df	η_p^2	ΔR^2	SSE	SST
(Intercept)	2.17	1.00	0.03		79.06	126.39
Problem Solving	18.03	1.00	0.19	0.14	79.06	126.39
Exam Score	15.30	1.00	0.16	0.12	79.06	126.39
Cum. GPA	2.66	1.00	0.03	0.02	79.06	126.39

Table 3: Model change with GPA, Exam Score and Problem Solving Score

cumulative GPA correlated significantly with both course outcome variables (r = .360, r = .358, see table 1). The regression model was recalculated, adding GPA as a predictor. The delta R-squared statistic for this additional predictor was small ($\Delta R^2 = .0210$), and the values for the other 2 predictors were reduced somewhat (exam score: $\Delta R^2 = .1211$, problem solving score: $\Delta R^2 = .1427$, see table 3). The η_p^2 value of 0.19 for problem solving indicates that the effect size is still medium for problem solving as a predictor, even accounting for the effect of GPA in the model. In the resulting model, problem solving score surpassed exam score as the predictor that explained the largest proportion of variance in programming project outcomes. Therefore, the predictive value of problem solving scores was largely independent of previous GPA, although previous GPA substantially contributes to explaining variation on both assessments. The results of this analysis, therefore, further support the conclusion that problem solving uniquely explains differences in higher level programming skills.

5. DISCUSSION

In this study, we examined the use of problem solving as a predictor of beginners' programming outcomes. In particular, we were interested in the extent to which problem solving might be related to the different types of assessment used in a CS1 course. We found that the problem solving scores measured at the beginning of the term are predictive of student outcomes, but only the outcomes on the constructed-response programming assignments, and not the outcomes on multiple choice exams. Similar to Porter et al. (2014), this study corroborated the association between code-writing and multiple choice grades. However, unlike Porter et al. (2014), the results of this study identified a predictor of programming success that is able to differentiate between the two types of assignment. Furthermore, this predictor is especially valuable because it is a preexisting characteristic measured at the beginning of the semester.

While Porter et al.'s observation that you must be able to trace code in order to write code appears true, it does not follow of course that the ability to trace entails the ability to code. There is an association between the lower and higher level programming skills, as this study confirmed, the question remains however, what can explain the differences in outcomes for higher level skills beyond the foundational effect of the lower level skills? This study found that problem solving ability is uniquely predictive of higher level skills above and beyond the effect of lower level skills. These results not only provide evidence for the notion that programming knowledge/ability is hierarchical, but they also suggest what abilities underlie the upper level skills and distinguish them from the lower level skills. Problem solving is a means by which we can explain this distinction between higher and lower level skills. Furthermore, the finding that controlling for previous GPA did not significantly detract from the predictive value of problem solving further supports the conclusion that problem solving ability has a unique predictive value for success on higher level programming tasks. These findings offer a basis for early identification of students who may encounter difficulties at higher level programming tasks.

6. CONCLUSION

In this paper, we have found evidence that programming ability is hierarchical, and goes beyond what can be assessed by a multiple choice test. We have observed that the variability of outcomes at the higher levels of the programming skills hierarchy is associated with generic problem solving skills. These results support the hierarchy of programming skills developed by Lopez et al. (2008) and the other BRACElet studies, and also provide a more substantive picture of what differentiates higher level skills like code writing from lower level skills like code tracing, using the theoretical framework of problem solving[14, 17, 21, 13, 18]. Problem solving ability helps explain success on the most difficult programming tasks.

7. REFERENCES

[1] PISA Problem Solving Items and Scoring Guides, 2003.

[2] The PISA 2003 Assessment Framework. Technical report, Organisation for Economic Co-operation and Development, 2003.

[3] P. Azalov, S. Azaloff, and F. Zlatarova. Work in Progress - Comparing Assessment Tools in Computer Science Education: Empirical Analysis. *Proceedings of 34th ASEE/IEEE Frontiers in Education Conference, 20-23 October 2004*, 2:18–19, 2004.

[4] R. J. Barker and E. a. Unger. A predictor for success in an introductory programming class based upon abstract reasoning development. *ACM SIGCSE Bulletin*, 15(1):154–158, 1983.

[5] S. Bergin and R. Reilly. Predicting introductory programming performance: A multi-institutional multivariate study. *Computer Science Education*, 16(February 2015):303–323, 2006.

[6] J. A. Dossey, S. S. Mccrone, C. O. Sullivan, P. Gonzales, and M. Schneider. Problem Solving in the PISA and TIMSS 2003 Assessments. Technical report, 2007.

[7] R. J. Enbody, W. F. Punch, and M. McCullen. Python CS1 as preparation for C++ CS2. *Proceedings of the 40th ACM technical symposium on Computer science education - SIGCSE '09*, page 116, 2009.

[8] N. Frederiksen. Implications of Cognitive Theory for Instruction in Problem Solving. *Review of Educational Research*, 54(3):363–407, 1984.

[9] D. C. Gibbs. The effect of a constructivist learning environment for field-dependent/independent students on achievement in introductory computer programming. *ACM SIGCSE Bulletin*, 32(1):207–211, 2000.

[10] W. Kuechler and M. Simkin. How well do multiple choice tests evaluate student understanding in computer programming classes? *Journal of Information Systems Education*, 14(4):389–399, 2003.

[11] B. L. Kurtz. Investigating the relationship between the development of abstract reasoning and performance in an introductory programming class. *ACM SIGCSE Bulletin*, 12(1):110–117, 1980.

[12] M. C. Linn. Cognitive Consequences Instruction in Classrooms Programming The. *Educational Researcher*, 14(5):14–16,25–29, 1985.

[13] R. Lister, C. Fidge, and D. Teague. Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *ACM SIGCSE Bulletin*, 41(3):161, 2009.

[14] M. Lopez, J. Whalley, P. Robbins, and R. Lister. Relationships between reading, tracing and writing skills in introductory programming. *Proceeding of the fourth international workshop on Computing education research - ICER '08*, pages 101–112, 2008.

[15] R. E. Mayer, J. L. Dyck, and W. Vilberg. Learning to program and learning to think: what's the connection? *Communications of the ACM*, 29(7):605–610, 1986.

[16] D. Palumbo. Programming Language / Problem-Solving Research : A Review of Relevant Issues. *Review of Educational Research*, 60(1):65, 1990.

[17] A. Philpott, P. Robbins, and J. L. Whalley. Assessing the Steps on the Road to Relational Thinking. In *20th Annual Conference of the National Advisory Committee on Computing Qualifications, Nelson, New Zealand.*, page 286, 2007.

[18] L. Porter, D. Zingaro, and R. Lister. Predicting student success using fine grain clicker data. In *Proceedings of the tenth annual conference on International computing education research - ICER '14*, pages 51–58, 2014.

[19] W. M. Reed, D. B. Palumbo, A. L. Stolar, W. M. Reed, and D. B. Palumbo. The Comparative Effects of BASIC and Logo Instruction on Problem-Solving Skills. *Computers in the Schools*, 4(3-4):105–118, 1988.

[20] M. G. Simkin and W. L. Kuechler. Multiple-Choice Tests and Student Understanding: What Is the Connection? *Decision Sciences Journal of Innovative Education*, 3(1):73–98, 2005.

[21] A. Venables, G. Tan, and R. Lister. A Closer Look at Tracing, Explaining and Code Writing Skills in the Novice Programmer. In *Proceedings of the Fifth International Workshop on Computing Education Research Workshop - ICER '09*, pages 117–128, 2009.

[22] B. C. Wilson and S. Shrock. Contributing to success in an introductory computer science course: a study of twelve factors. *ACM SIGCSE Bulletin*, 33(1):184–188, 2001.

APPENDIX

A. PISA TEST ITEM EXAMPLE

Adapted from PISA Problem Solving Items and Scoring Guides available at http://nces.ed.gov/surveys/pisa/pdf/items2_solving.pdf [1]

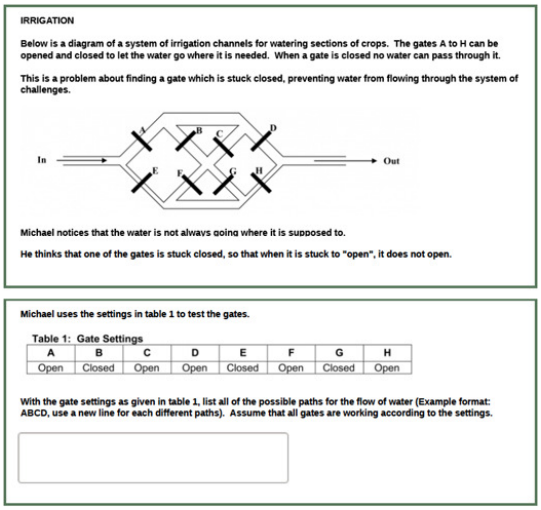

Figure 1: Troubleshooting Item: Irrigation (Question 1)

B. COURSE MATERIALS

Archive of programming assignment examples: http://www.cse.msu.edu/∼cse231/PracticeOfComputing UsingPython/

Archive of exam assignment examples: http://www.cse.msu.edu/ cse231/Exams/