The Effectiveness of Visualization for Learning Expression Evaluation: A Reproducibility Study

Amruth N. Kumar
Ramapo College of New Jersey
Mahwah, NJ 07430, USA
1 201 684 7712
amruth@ramapo.edu

ABSTRACT

A study was conducted to reproduce the results of an earlier study on the effectiveness of visualization for learning expression evaluation in a problem-solving software tutor on arithmetic expressions. In the current reproducibility study, data was collected from a software tutor on assignment expressions over six semesters. ANOVA analysis of the amount and speed of learning was conducted with treatment, sex and racial groups as fixed factors. Results include that visualization helped the students learn significantly more concepts, whether the students needed to use the tutor or benefited from using the tutor. However, it only benefited the less-prepared students. It did not help the students learn faster. It benefited both the sexes and traditionally represented as well as underrepresented groups. The current study confirmed almost all the results from the previous study, albeit for a harder topic. One reason why visualization was found to be effective in both these studies may be that the same visualization scheme was used by the students to both view feedback and construct their answers.

1. INTRODUCTION

Program visualization deals with visualizing programs at lower levels of abstraction [10]. It may be static or dynamic, the latter also referred to as program animation. Program visualization systems may be specialized or generic: the former visualize specific programming constructs, whereas the latter visualize entire programming languages [10]. The subject of the current study is static program visualization, specialized for expression evaluation.

A systematic review of generic program visualization and animation systems cataloged a mix of systems that were never evaluated, those that did not yield positive results, those whose results were not statistically significant and those with significant positive results [10]. Another survey of successful evaluations of visualization systems found that about half of the evaluations were only about usability; and a third were informal, "with little contribution to future improvements" [11]. A meta-study of algorithm visualization, the other type of software visualization found similarly equivocal results [2].

Few of the specialized visualization systems, i.e., those built for specific programming constructs, have been evaluated. Among those that have been evaluated, one study found that animation was no more effective than text explanation for learning the semantics of C++ pointers [3]. Another study found that graphic visualization with text explanation was better than graphic visualization alone when learning expression evaluation [4].

A recent large-scale study of a specialized visualization system for arithmetic expression evaluation found that visualization indeed helped students learn more concepts, but that the "benefits primarily accrued to less-prepared students." [6]. Arithmetic expressions, the subject of the reported study, are arguably one of the easiest topics in introductory programming. Could the results of the study be **reproduced** with assignment expressions, arguably one of the harder expressions in programming languages thanks to prefix, postfix, and compound assignment operators? This was the question addressed by the current study.

Reproducibility is a core principle of scientific research. A recent study of reproducibility of 100 results published in 2008 in three top Psychology journals highlights its importance: It found that only 35 of the 100 results could be reproduced at a statistically significant level [8]. The reproduced results were weaker than the claims made in the original paper for *all* 100 studies, although no claim was disproven. A hyper-competitive scientific culture that prizes novelty, and provides little incentive to reproduce earlier findings or publish results of such studies was found partly to blame for this state of affairs. Reproducibility as an issue is increasingly being addressed by numerous Computer Science research communities, as revealed by a search of the ACM digital library (e.g., Human Computer Interaction, Software Engineering, Recommender Systems, Databases, Simulation, Systems Research, Data Mining), but Computer Science Education research is not one of them.

Reproducibility is not replicability. Whereas reproducibility refers to the ability to draw the same results using different instruments, methods, protocols and/or participants, replicability refers to repeating the original experiment with exactly the same instruments, methods, protocols and participants to see if the same results can be obtained. Reproducibility is desirable, whereas replicability is not even "good science" [1].

Being able to reproduce the results of the earlier study on arithmetic expressions [6] would not only provide additional support to the results of the earlier study, but also extend the

ITiCSE '16, July 09 -13, 2016,Arequipa, Peru
© 2016 ACM. ISBN 978-1-4503-4231-5/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2899415.2899427

results to harder topics such as assignment expressions. In this context, the current study was conducted to evaluate the effectiveness of visualization in a software tutor on assignment expressions and possibly reproduce or refute results from the earlier study [6].

2. METHODOLOGY

2.1 Participants

The participants of the study were students in introductory programming courses from 41 institutions: 1721 students from baccalaureate institutions, 154 from community colleges and 104 from high schools over six semesters: Fall 2011-Spring 2014. Students were given the option to identify their sex and race. 1348 students identified themselves as male, and 487 as female. 1275 students identified themselves as Caucasians or Asians, the traditionally represented groups in Computer Science, and 256 identified themselves as belonging to underrepresented racial groups. Since this was a controlled study, institutions were randomly assigned to control or experimental group each semester.

2.2 Instrument – The Software Tutor

The instrument used for this study was a software tutor on assignment expression evaluation. The tutor presents expressions to the student, has the student evaluate each expression one operator at a time, grades the student's answer and provides feedback. The student evaluates each operator by dragging the mouse across the operator and appropriate operands to draw an underbrace across them, and entering the intermediate result in the dialog box presented for the underbrace. The feedback includes whether the student's answer is correct and step-by-step explanation of the correct answer, which has been shown to help students learn [5].

The tutor covers the following concepts: simple assignment, compound assignment, prefix and postfix increment and decrement operators, precedence and associativity of assignment operators, and narrowing and widening coercion during assignment. The tutor is accessible over the web – students can use it on their own time, and at their own convenience. It is part of a suite of problem-solving tutors for introductory programming topics, available for free for educational use called problets (problets.org).

The feedback provided by the tutor is in two forms:

- **Text explanation** feedback explaining each step in the evaluation of the expression. For example, the text explanation provided for the expression `-- weight` is:

```
The value of weight is 14
weight is decremented to 13
-- weight returns 13
Prefix -- operator returns the value of
the variable after decrementing it
```

- **Graphic visualization** feedback in the form of an underbrace spanning the operator (`--` in the above example) and operands (`weight` above), with the intermediate result (`13` above) drawn centered underneath the underbrace.

Figure 1 shows a snapshot of the two forms of feedback.

Graphic visualization explains the order of evaluation of operators, but not concepts such as coercion. So, it was provided in addition to rather than instead of text explanation. The benefits of simultaneous presentation of the same information in text and visual forms is explained by dual coding theory [9], which postulates that visual and verbal information are processed differently, in separate channels, to organize the information and create separate mental representations, either of which can be used later to recall the information.

2.3 Protocol

The software tutor administered pre-test-practice-post-test protocol as follows:

Pretest – During the pretest, the tutor presented one problem per concept. If a student solved a problem correctly, no feedback was provided to the student, and no more problems were presented to the student on the concept. On the other hand, if the student solved a problem incorrectly, or opted to skip the problem because the student did not know the answer, feedback was presented to the student and additional problems on the concept were scheduled to be presented during the subsequent stages.

Adaptive practice – Once a student had solved *all* the pretest problems, practice problems were presented to the student on only the concepts on which the student had solved problems incorrectly during the pre-test. For each such concept, the student was presented multiple problems until the student mastered the concept. After solving each problem, the student received feedback explaining the correct answer. Since this was a controlled study, students in the control group received only text explanation whereas those in the experimental group received both text explanation and graphic visualization.

Post-test - During this stage, the student was presented test problems on the concepts that the student had mastered during the adaptive practice.

Demographics - Students were provided the option to identify their demographic information, including sex and race.

The entire protocol was limited to 30 minutes and was administered back-to-back, entirely over the web. A concept was considered to have been learned during this session if the student solved the problem on that concept incorrectly during the pre-test, solved enough problems during the adaptive practice to master the concept, and proceeded to solve the problem on the concept correctly during the post-test.

2.4 Design

The dependent variables of this study were all between-groups:

- Pre-test mean score per problem, calculated as the total pre-test score divided by the number of pre-test problems solved – this normalized the variation in the number of problems solved by the students (10-17). Since the students used the tutor after classroom instruction on assignment expressions, pre-test score was a measure of their *prior knowledge*;
- Number of concepts learned;
- Number of practice problems solved per learned concept, calculated as the number of problems solved during practice, divided by the number of concepts learned, once again, to normalize the variation in the number of concepts learned by the students. Since the tutor was adaptive, and presented practice problems on a concept until the student

had mastered the concept, the more the practice problems a student needed to learn a concept, the slower *the pace of learning*.

The independent variables of this study were:

- Treatment – text explanation only versus text explanation with graphic visualization.
- Sex: male or female.
- Representation – traditionally represented (Caucasians and Asians) versus underrepresented (the other racial groups, viz., Black/African American, Hispanic/Latino, Native American, Native Hawaiian/Pacific Islander and Other races)

2.5 Data Collection

The pre-test contained 17 problems for C++ and 16 problems for Java. Since students could use the tutor as often as they pleased, if a student used the tutor multiple times, only the first attempt when the student solved all the pre-test problems was considered. If the student never solved all the pre-test problems, the attempt with the most number of pre-test problems solved was considered. In order to eliminate trial or frivolous attempts by students, only those sessions were considered where students solved at least 10 pre-test problems. After this sifting, the control group contained 1112 students and experimental group, 867 students – this was the group for which statistics were reported earlier in Section 2.1.

Although students from high schools, community colleges and undergraduate institutions used the tutor, in order to maintain homogeneity of participant population, we considered only students from undergraduate institutions.

2.6 Data Analysis

A typical expression contains one or more operators. The grade awarded for a problem was calculated by the software tutor as the number of operators correctly evaluated, divided by the total number of operators in the expression. Therefore, the score on each problem was normalized to 0 $\rightarrow$ 1.0 regardless of the number of operators in the expression.

Univariate ANOVA analysis was conducted for the dependent variables listed in Section 2.4, with treatment, sex and representation as the fixed factors.

3. RESULTS

We first analyze the data of all the students. Thereafter, we consider subsets of students based on programming language, need, benefit and preparation to see if we can localize the results obtained for the entire population. We list only significant results (i.e., main effects), except for treatment.

3.1 All the undergraduate students

The results of analyzing the pre-test score per problem were:

- *No difference in the prior knowledge of the control (N=751) and test (N=565) groups*. So, any subsequent difference between the groups can be attributed to the use of the tutor.
- Male students scored significantly higher than female students and traditionally represented students scored significantly higher than students from under-represented groups as shown in the table below. *So, male students and students from traditionally represented groups had significantly greater prior knowledge than their counterparts*.

Pre-test Score	N	Mean
Sex [F(1,1315) = 8.845, p = 0.003]		
Male	965	0.872 ± 0.013
Female	351	0.835 ± 0.021
Representation [F(1,1315) = 13.601, p < 0.001]		
Traditional	1121	0.877 ± 0.01
Underrepresented	195	0.83 ± 0.023

Analysis of the number of concepts learned found that the experimental group learned significantly more concepts than the control group [F(1,787) = 3.831, p = 0.051] as shown in the table below, i.e., *students learned more with visualization than without.*

Concepts Learned	N	Mean
Control	446	2.679 ± .288
Test	342	3.104 ± .314

Analysis of the number of practice problems solved per learned concept yielded no significant results.

3.2 C++ Students

We considered C++ students who had solved all 17 pre-test problems (N=352). The results of analyzing the pre-test score per problem, summarized in the table below, were that the control group students scored significantly *more* than the experimental group students; male students scored significantly higher than female students; and traditionally represented students scored significantly higher than students from under-represented groups. In other words, the control group students, male students and students from traditionally represented groups had significantly greater prior knowledge than their counterparts.

Pretest Score	N	Mean
Treatment [F(1,351) = 9.778, p = 0.002]		
Control	275	0.882 ± 0.027
Test	77	0.800 ± 0.043
Sex [F(1,351) = 13.222, p < 0.001]		
Male	285	0.889 ± 0.026
Female	67	0.794 ± 0.044
Representation [F(1,351) = 27.607, p < 0.001]		
Traditional	296	0.91 ± 0.023
Underrepresented	56	0.773 ± 0.046

The results of analyzing the number of concepts learned, as summarized in the table below, were that the experimental group learned significantly more concepts than the control group, i.e., *students learned significantly more with visualization than without;* and female students learned significantly more concepts than male students.

Concepts Learned	N	Mean
Treatment [F(1,160) = 5.233, p = 0.024]		
Control	127	2.456 ± .471
Test	34	3.621 ± .889
Sex [F(1,160) = 4.932, p = 0.028]		
Male	128	2.473 ± 0.709
Female	33	3.604 ± 0.715

The results of analyzing the number of practice problems solved per learned concept, as summarized in the table below, were that the experimental group solved significantly more problems per learned concept than the control group, and female students solved significantly more problems per learned concept than male students. In other words, *the pace of learning was significantly slower with visualization than without, and for female students as compared to male students.* .

Practice Problems	N	Mean
Treatment [F(1,160 = 4.118, p = 0.044]		
Control	127	3.105 ± 0.276
Test	34	3.71 ± 0.52
Sex [F(1,160) = 9.787, p = 0.002]		
Male	128	2.941 ± 0.415
Female	33	3.874 ± 0.418

3.3 Java Students

We considered Java students who had solved all 16 pre-test problems (N=730). The results of analyzing the pre-test score per problem were:

- *There was no difference in the prior knowledge of the control (N=346) and test (N=384) groups.*
- Male students scored marginally higher than female students, and traditionally represented students scored significantly higher than students from under-represented groups, as shown in the table below.

Pretest Score	N	Mean
Sex [F(1,729) = 2.996, p = 0.084]		
Male	517	0.875 ± 0.017
Female	213	0.845 ± 0.029
Representation [F(1,729) = 5.259, p = 0.022]		
Traditional	629	0.88 ± 0.012
Underrepresented	101	0.84 ± 0.031

Analysis of the concepts learned and the number of practice problems solved per learned concept yielded no significant results.

3.4 Students who needed to use the Tutor

A normalized score of 1.0 represented ceiling effect. The students who scored less than 1.0 stood to benefit from using the tutor, since they had incorrectly solved one or more problems during the pretest. For our next analysis, we considered all the students whose normalized score was 0.95 or less – this included even those who had incorrectly solved exactly one problem.

The results of analyzing the pre-test score per problem were:

- *There was no difference in the prior knowledge of the control (N=438) and test (N=327) groups.*
- Traditionally represented students scored significantly higher than students from under-represented groups [F(1,764) = 7.34, p = 0.007] as shown in the table below.

Pretest Score	N	Mean
Traditional	636	0.799 ± 0.012
Underrepresented	129	0.759 ± 0.026

Analysis of the concepts learned found that the experimental group learned significantly more concepts than the control group [F(1,619) = 4.973, p = 0.026] as shown in the table below, i.e., *students learned significantly more with visualization than without.*

Concepts Learned	N	Mean
Control	264	3.019 ± .316
Test	356	3.547 ± .342

Analysis of the number of practice problems solved per learned concept yielded no significant results.

3.5 Students who benefited from the Tutor

The students who benefited from using the tutor were those who learned at least one concept. For our next analysis, we considered all the undergraduate students who had learned at least one concept.

The results of analyzing the pre-test score per problem were:

- *There was no difference in the prior knowledge of the control (N=446) and test (N=342) groups.*
- Male students scored marginally more than female students, and traditionally represented students scored significantly higher than students from under-represented groups, as shown in the table below.

Pretest Score	N	Mean
Sex [F(1,787) = 2.769, p = 0.097]		
Male	559	0.83 ± 0.16
Female	229	0.806 ± 0.23
Representation [F(1,787) = 8.7, p = 0.003]		
Traditional	677	0.84 ± 0.011
Underrepresented	111	0.797 ± 0.027

Analysis of the concepts learned found that the experimental group learned significantly more concepts than the control group [F(1,787) = 3.831, p = 0.051] as shown in the table below, i.e., *students learned significantly more with visualization than without.*

Concepts Learned	N	Mean
Control	342	2.679 ± .288
Test	446	3.104 ± .314

Analysis of the number of practice problems solved per learned concept yielded no significant results.

3.6 Less-prepared Students

The mean of normalized pre-test scores for the entire undergraduate cohort was 0.853. The students who scored 0.853 or lower on the pre-test were less-prepared as compared to those who scored more than 0.853. We next considered the less-prepared undergraduate students.

Analysis of the pre-test score per problem found *no difference in the prior knowledge of the control (N=281) and test (N=177) groups.*

Analysis of the concepts learned found that the experimental group learned significantly more concepts than the control group [F(1,379) = 6.859, p = 0.009] as shown in the table below, i.e., *students learned significantly more with visualization than without.*

Concepts Learned	N	Mean

Control	233	3.568 ± .377
Test	147	4.326 ± .427

Analysis of the number of practice problems solved per learned concept yielded no significant results.

3.7 Better-prepared Students

Finally, we considered the better-prepared students: those whose normalized score was over the cohort average of 0.853, but under 1.0. We excluded the students who had scored 1.0 since they knew all the concepts, and could not benefit from using the tutor.

Analysis of the pre-test score per problem found that *there was no difference in the prior knowledge of the control (N=249) and test (N=246) groups*.

Analysis of the concepts learned and the number of practice problems solved per learned concept yielded no significant results.

4. DISCUSSION

The following table summarizes the results of the various cases that we analyzed in the previous section. The rows correspond to the 7 cases we considered: **All** the students, **C++** students, **Java** students, the students who **need**ed the tutor, the students who **benefit**ed from using the tutor, the **less**-prepared students and the **better**-prepared students. For each of **Pretest Score**, **Concepts Learned** and **Problems per Learned concept**, the columns list Treatment (**T**), Sex (**S**) and Representation (**R**). A cell contains a check mark if a statistically significant result was found for it, e.g., when **all** the students were considered, a significant difference was found between the sexes (**S**) on the **Pretest Sore**.

	Pretest Score			**Concepts Learned**			**Problems per Concept**		
	T	**S**	**R**	**T**	**S**	**R**	**T**	**S**	**R**
All		√	√	√					
C++	√	√	√	√	√		√	√	
Java		√	√						
Need				√					
Benefit			√	√					
Less				√					
Better									

From the **Pretest Score (T)** column of the table, it is clear that the prior preparation of the control and experimental groups was statistically comparable in all but one case. Yet, from the **Concepts Learned** (**T**) column of the table, it is clear that *visualization helped the students learn significantly more concepts in most cases. It helped the students learn more concepts, whether the students needed to use the tutor or benefited from using the tutor. However, it only benefited the less-prepared students, not the better-prepared ones.*

From the **Problems per Concept** (**T**) column of the table, it is clear that *visualization did not help the students learn faster, as measured by the number of practice problems solved per learned concept.* In the one instance when a significant difference was found for treatment (C++), learning was slower with visualization than without!

Considering the lack of significant differences in **S** and **R** columns of **Concepts Learned** and **Problems per Concept**, we can state that 1) the tutor was not biased towards either sex or racial group; and 2) *both the sexes and racial groups benefited the same from using the tutor*. In the one case (C++) when a significant difference was found for sex, female students learned *more* concepts, although at a *slower* pace than male students.

From (**S**) and (**R**) columns of **Pretest Score**, we can summarize that regrettably, *female students were often less-prepared before using the tutor than male students and underrepresented students were often less-prepared than traditionally represented students*. In the prior study, arithmetic expressions were used [6]. Since many of the concepts covered as arithmetic expressions in programming languages are typically also covered in K-12 math, it is harder to localize the reason for the differences in the prior preparation of the sexes and racial groups, if any. Assignment expressions on the other hand are concepts unique to programming languages – it is safe to assume that they are not covered in K-12 math. Any prior knowledge of assignment expressions is acquired by the students in class or through programming activity. This leads to one of two explanations for the lower prior-preparation of female and underrepresented students:

- Although introductory programming courses typically assume that students will not have had any prior programming experience, this may not be entirely accurate. Male and traditionally represented students may be entering the introductory programming course with more exposure to programming concepts than their counterparts. If so, providing additional encouragement to female and underrepresented students in high school to engage in programming activities may redress this difference.
- Somehow, male and traditionally represented students learn better from classroom instruction of introductory programming concepts than their counterparts. If so, Computer Science educators and education researchers may want to isolate the instructional strategies for introductory programming that are effective across sexes and races and propagate them.

Since this is a reproducibility study, comparison of the results from this study with those from the previous study [6] are in order. Primary results of the prior study were that "visualization helped students learn more concepts; visualization did not increase the speed of learning; the benefits of visualization accrued primarily to less-prepared students; and visualization may affect different demographic subgroups differently". All the results were reproduced in the current study except the last one – the population size of the current study was not large enough to confirm/refute whether visualization affected different demographic subgroups (e.g., underrepresented female students) differently. Students find the subject of this study, viz., assignment expressions to be harder than arithmetic expressions, the subject of the prior study. So, in addition to confirming the results of the earlier study, this study extended those results to harder topics.

So, why was visualization found to be effective in both the studies, especially for the less-prepared students, when the form of visualization was "viewing", one of the least effective forms of engagement [7]? It could be because students also "constructed" their answer using the same visualization scheme before submitting it, and "constructing" is considered to be one of the highest levels of engagement [7]. Using the same visualization scheme to both view feedback and construct answers may hold the key to improving the effectiveness of visualization in software tutors. For instance, just as drawing underbraces was found to be effective for evaluating expressions, the following student-constructed visualization techniques might turn out to be effective for other program comprehension tasks: drawing boxes around code segments to clarify scope; drawing arrows across code to trace the flow of control and data; and superimposing state diagram on the code to track and debug the lifecycle of variables. Future work includes evaluating the effectiveness of these student-constructed visualization techniques in online tutors.

5. ACKNOWLEDGMENTS

Partial support for this work was provided by the National Science Foundation under grant DUE 1432190.

6. REFERENCES

[1] Drummond, C. Replicability is not Reproducibility: Nor is it Good Science. Proc. Evaluation Methods for Machine Learning Workshop, 26th ICML, Montreal, Canada, 2009.

[2] Hundhausen, C.D., S.A. Douglas, and J.T. Stasko, A meta-study of algorithm visualization effectiveness. Journal of Visual Languages and Computing, 2002. 13(3): p. 259-290.

[3] Kumar, A.N., Data Space Animation for Learning the Semantics of C++ Pointers, in SIGCSE Technical Symposium2009: Chattanooga, TN. p. 499-503.

[4] Kumar, A.N., Results from the Evaluation of the Effectiveness of an Online Tutor on Expression Evaluation, in 36th SIGCSE Technical Symposium2005: St. Louis, MO. p. 216-220.

[5] Kumar, A.N., Explanation of step-by-step execution as feedback for problems on program analysis, and its generation in model-based problem-solving tutors. Technology, Instruction, Cognition and Learning (TICL) Journal, 2006. 4(1).

[6] Kumar, A.N. The Effectiveness of Visualization for Learning Expression Evaluation. Proc. 40th SIGCSE Technical Symposium on Computer Science Education. SIGCSE 2015. Kansas City, KS. 362-367.

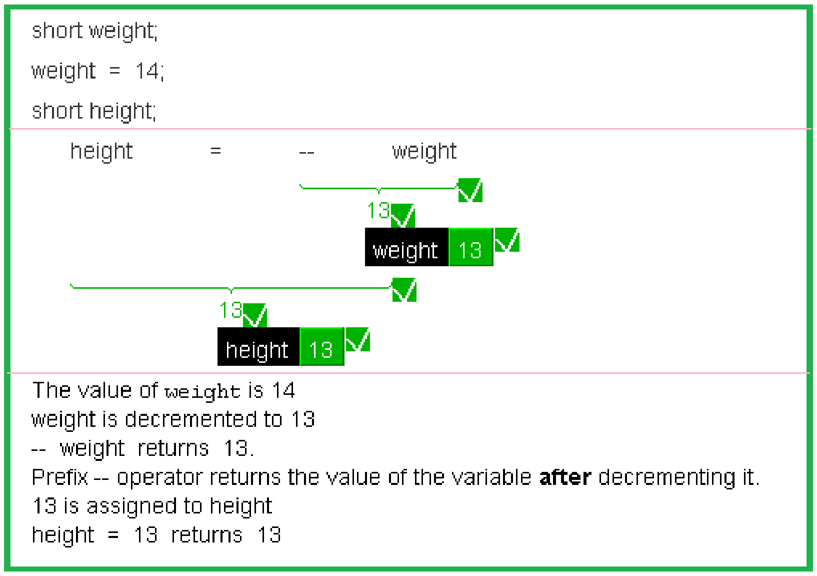
[7] Naps, T.L., et al., Exploring the role of visualization and engagement in computer science education, in SIGCSE Bulletin 2003. p. 131-152.

[8] Open Science Collaboration, Estimating the reproducibility of psychological science, Science, Vol. 349(6251), 28 August 2015

[9] Paivio, A., Mental representations: A dual coding approach, 1990, New York: Oxford University Press.

[10] Sorva, J., V. Karavirta, and L. Malmi, A Review of Generic Program Visualization Systems for Introductory Programming Education. Transactions on Computing Education, 2013. 13(4): p. 1-64.

[11] Urquiza-Fuentes, J. and J.Á. Velázquez-Iturbide, A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems. Transactions of Computing Education, 2009. 9(2): p. 1-21.


```
short weight;
weight = 14;
short height;
```

The value of `weight` is 14
weight is decremented to 13
-- weight returns 13.
Prefix -- operator returns the value of the variable **after** decrementing it.
13 is assigned to height
height = 13 returns 13

Figure 1 (© Amruth N. Kumar): Screen shot of the feedback provided by Assignment Expression Tutor. Both graphic visualization (between the two pink lines) and text explanation (after the second pink line) are shown. Courtesy: problets.org.