A User-Acceptance Evaluation of Two Web-based Computer Programming Teaching Tools

Jens O. Liegle
jliegle@cis.gsu.edu
Peter N. Meso
Dept. of CIS, Georgia State University
PO Box 4015
Atlanta, GA 30303

ABSTRACT

Learning computer programming through an online course is inherently difficult. This study presents results from the evaluation of two online tools that can be employed in teaching an online computer-programming course on structured programming. One of the tools, PROGSIM, allows code execution in a trace-like mode on the client's browser, while the other implements a question-answer system that allows students to self-test their comprehension of programming code semantic, and not just syntax. Using the Technology Acceptance Model, the two tools are compared to traditional online text-only instruction. Results are presented and discussed.

1. INTRODUCTION

The value of online and computer-mediated learning continues to be of great interest to both academia and practitioners. For areas where the knowledge to be taught is predominantly declarative or "factual know-that" (Alavi and Leidner, 2001), online learning resources are easily implemented. However, intellectual disciplines that rely heavily on procedural or "know-how" type knowledge such as computer programming, are much harder to teach within an online or computer-mediated environment. One reason for this is that students require interactive tasks such as debugging, validation, and testing to master the concepts being taught. It is no wonder that few tools exist that support learning in these type of environments.

There are several reasons that speak for the development of online-tools for teaching structured programming, even when such functionality comes bundled with Integrated Development Environments (IDE) such as Visual Studio. Among these are

- Can be integrated with other online-content such as lecture material
- Do not require an IDE
- Load quickly
- Do not require any training to use
- Access can be monitored
- Data from logfiles can be used for research

In this paper, we test the usefulness and ease of use of two prototype-teaching tools designed specifically for learning structured programming. In addition, we examine the comparative learning performance of students that use these tools versus a control group that uses plain web page tutorials. We expect that students that have the benefit of using interactive examples of running-code, as provided by the teaching tools, to perceive these tools to be more useful and to eventually obtain a better understanding of the learning material than the control group. Using the tools will take more time, and since no training was given to the subjects, we expect a lower perception of the tools'

ease of use compared to plain web page tutorials.

Therefore, our hypotheses are as follows:
Students who use a learning module enhanced with prototype learning tools will
- H1: rank the perception of usefulness higher
- H2: rank the perception of ease of use lower
- H3: perform better in the learning assessment
- H4: need more time to complete the learning task

than the control group

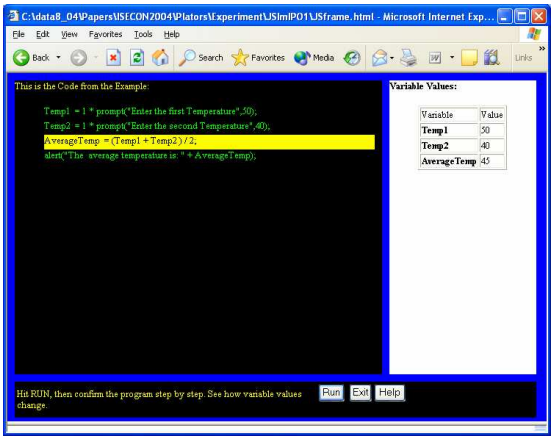
2. DESCRIPTION OF THE TOOLS

PROGSIM

While a number of computer programming tools exist (besides the obvious compiler, interpreter, and debugger) that allow real-time code execution, only a few of them are web-based, and these use a programming trick for real-time web-based programming: they execute the code on the server-side (Emurian, 2004; Janicki and Liegle, 2001A, 2001B). This technique only works reasonably well for programming languages like LISP (Corbett et al., 1992) because LISP can be embedded in a web server, and such a server can also execute LISP code. The use of LISP in the real world, however, is fairly limited, and LISP cannot be used to teach structured programming. While C or Java programs could be embedded in a web server, the code that the user sends to the server is not C or Java executable code; it is text that needs to be compiled into machine language. Thus, a server receiving this user code in text form is unable to interpret it, unless it is a LISP server that can interpret LISP in runtime.
There is a way to work around this problem. Code-execution capabilities can be built into an online-interpreter by submitting code from the user's browser to a server-side program, which then in turn starts a compiler/interpreter, executes the code, and simply displays the result to the user. Such tools are frequently used in computer science departments to test homework assignments. Here, programs are automatically compiled, and then started with either commandline arguments, or programs are written to get input from pre-specified files and write their output to again pre-specified files, which can then be compared to the correct solution. Since these systems usually cannot support real-time user input or program output due to the additional layer of the WWW in between, these types of systems are limited to simple calculations or file manipulations.

PROGSIM (PROgramming SIMulator), a web-based structured programming language interpreter that allows the real-time demonstration and practice of JavaScript and Visual Basic Script code, provides the user with real-time programming capabilities. PROGSIM makes use of the fact that one of the freely available web-browsers, Microsoft Internet Explorer, has the capability to execute both of these languages. The client-side tutor can use PROGSIM to allow users to trace their own (or provided) code while watching the values of the variables change in a debugger-style watch window. Since the user could code endless-loops and actually "hang" the system, a step-wise execution of programming lines was required. See Figure 1 for a screenshot of the PROGSIM prototype.

Figure 1: Screenshot of PROCSIM



The PROGSIM programming environment is used to support the following teaching strategies:
- *Demonstration and Evaluation*. Sample code is shown to the user.
- *Simulation*. User can execute sample code. Input is user driven, and output is immediately visible to the user in the form of message boxes.

- *Completion.* User receives partial code to a problem and has to complete it. Correctness is verified by the user and in simple cases by the system, e.g. by comparing the value of an output variable to an expected value.
- *Generation and Problem Analysis.* User receives a problem statement and has to generate the code that solves this problem. PROGSIM does not provide a grading functionality for this type of teaching, since the potential variations of code are too complex to be interpreted and graded at this point in time. However, the users can run and debug their code to see whether it produces the expected results.

PROGSIM requires extensive use of inter-frame communication (i.e., between HTML frames) in order to allow the execution of user/demo code. The code and the variable definitions are stored in a form in one frame (A). This code is read by a function in another frame (B) and then parsed. For each variable, two array entries are created, one for the name of the variable, and one for the current value of it (only numeric values are supported at this time). Each occurrence of a variable is replaced with a line of code referencing the value-slot of the corresponding array, e.g.

```
Sales= 3
```

becomes

```
ValueArray[get("Sales")] = 3.
```

Since all the values with the corresponding names are stored in an array, PROGSIM can display the names and values of all variables in a <DIV> area at runtime after a line of code is executed. After each line of code, a `If(wait(n)){return 0;}` function is added with "n" being the current line number (JavaScript version). The *wait(n)* function highlights the current line "n" and displays a message to the user asking whether the next line should be executed or the program stopped. The condition around the function call allows the termination of endless loops.

This manipulated code is then written with `A.document.writeln(`*codeline*`)` into frame A in form of a *function mycode()* (JavaScript version) or *Sub mycode* (VBScript version). Once the user clicks on the RUN button, this dynamically created function is called. The user can go back to the source code at any time and make changes until the program works as intended.

PROGCDT

Besides showing the learner a lecture on the syntax of commands and code samples, exercises are needed that allow learners to practice their newly acquired knowledge.

Figure 2: PROGCDT in Tutorial Mode

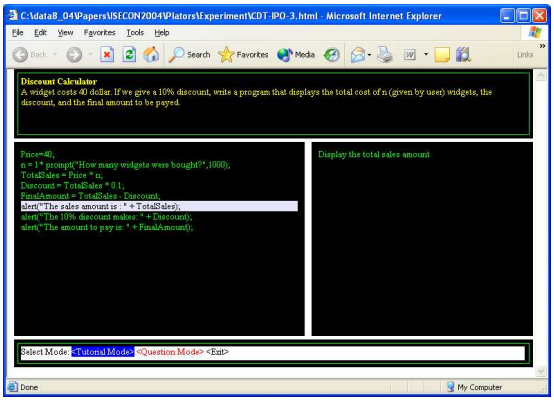
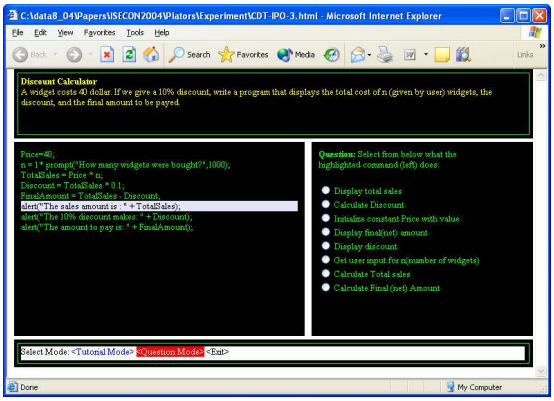


Figure 3: PROGCDT in Question Mode



Merrill (Merrill et al., 1996; Merrill, Li, & Jones, 1991) worked on developing a set of instructional strategies that can be applied to different types of knowledge. PROGCDT (PROgramming Component Display Theory) is based on Merrill's work on Component Display Theory and Instructional Design. This module uses DHTML to implement the following information strategies (See Figures 2 and 3):

- *What-is-this?* User clicks on code and receives explanation of its functionality.
- *Where-is-the-code-that-does…?* User sees a question and has to locate the corresponding code line by clicking on the code segment that performs the specified action.
- *What-does-this-code-segment-do?* A random code segment is marked. The user then has to select from a list of choices the corresponding action that the marked code segment performs.

3. TECHNOLOGY ACCEPTANCE MODEL (TAM)

According to pedagogical research, effective teaching tools and technologies enhance the learning capability of students and make the mastery of difficult principles much more simpler (Liddle, Brown et al., 1995; Janicki and Liegle, 2001). Research in this area also points out that the teaching tools and technologies that prove to be effective in most cases are those that (1) are easy to use and easy to learn, (2) map a clear and direct path from the problem to its correct solution, allows for hand-on-learning or learning-by-doing rather than passive learning such as demonstrations by an instructor, and (3) minimize the technological barriers between student and the core-knowledge or principles being disseminated to the student (Janicki and Liegle, 2001).

According to IT innovation diffusion theories, these very same principles propel the rapid diffusion of a new information technology and are commonly referred to as the Technology Acceptance Model (TAM) (Davis, 1989, Davis et al. 1989). The TAM is the leading theory used to explain adoption of information technology by individuals in business and industry (Gallivan, 2001; Chircu et a., 2000; Straub et al., 1997). Research abounds on the use of TAM in explaining individual adoption and acceptance of IT and the antecedent and consequent factors that propel such diffusion within groups and organizations (Davis, 1989, Davis et al. 1989; Szanja, 1996; Agrawal and Prasad, 1997; Thompson, Higgins and Howell, 1991; Moore and Benbassat; 1991; Karahana, Straub and Chervany, 1999; Gallivan, 2001).

TAM theory posits that those technologies that bear properties that enhance their adoption will be more readily embraced and accepted by any type of user-group, including students. This holds true even in the context of an IS course. This means that the theories of IT innovation adaptation may be beneficial and effective at identifying the suitability and fit of particular emerging information technology as a pedagogical tool for a select type or group of IS courses.

The Technology Acceptance Model states that the factors that propel the diffusion of an Information technology are its ease-of-use and its usefulness (Davis, 1989; Gallivan, 2001). TAM has been used to explain diffusion using user-perception measures as well as actual usage measures. Within the context of an IT course, we expect that students will be attracted to that technology that is actually easy to use and directly relevant to the course requirement tasks that they must complete, or to that technology that they perceive as bearing these traits. Therefore, assessing the reactions of students toward a particular technology can determine the effectiveness of that technology as a pedagogical tool for the course in question.

4. PILOT TEST

A class of experienced programmers who were enrolled in a Java programming class received extra credit for testing the tools. The emphasis of this pilot test was to identify weaknesses of the system, problems with the actual lecture content, server performance issues, and possible interface design problems. All participants already knew the principles of Java and had, as a prerequisite, knowledge of another programming language (mostly Visual Basic).

A total of 28 students completed the tutorial and filled out a report sheet. For each micro-lesson, students had to report errors, typos and bugs that they found. In addition, open-ended questions asked them what they liked and disliked about each module. Furthermore, they rated each module on several statements using a Lickert scale from 1

(strongly agree) to 5 (strongly disagree). For a summary of the results, see Table XXX.

A number of problems with the system were identified. First, students complained that the mouse pointer would not change into a "hand" symbol on words or symbols that looked and acted like hyperlinks. These words or symbols were in fact "hot" words. However, they were using <DIV> or <SPAN> tags with `onClick()` methods instead of anchor tags. The problem of the missing "hand" mouse pointer is a Browser implementation problem and could not be solved directly. However, the instructions for the system were improved and pointed out this potential problem.

Another problem was that once a student wrote a JavaScript program in PROGSIM which had an error, the Browser automatically started the Visual Studio Debugger (if installed). This totally confused people, took a long time to load and quit, and did not help in debugging the user's code, since in fact the entire PROGSIM was loaded into the debugger. There was no solution to this problem due to a lack of an `OnErrorGoto` function in JavaScript. The instructions were upgraded to point out this problem.

Overall, the system was very well received by the class as these selected comments from the pilot group show: "I liked how you let the user try out the PROGCDT", "Code show a plus", "It was a good intro, "Good use of many examples", "Demonstration was done well", "The quiz really made me think about what I read", "Simple, but to the point", "Like the examples; they are understandable because they relate to the real world", "Cool tools".

5. EXPERIMENT

A total of 40 Students participated in a short (20min) experiment. Expert programmers and people with prior experience with the subject matter (simple input/processing/output in java script) were excluded from the study, and some students did not complete all parts of the experiments. A total of 32 valid subjects remained, which were randomly assigned to one of two groups. Using a browser interface, Group 1 (control group) received a text-only version of the tutorial material (See Appendix), while Group 2's version of the web-page contained additional links to PROGSIM and PROGCDT tools for each of the 4 examples that were given. Both groups filled out a pre- and a post-test on the subject matter, and in addition completed a version of Kolb's Learning Style inventory survey (Romero et al, 1995) and a short questionnaire about the ease of use and usefulness, as measured by TAM (technology acceptance model) (Davis, 1989). The results are presented in Tables 2-4.

Since Levene's Test for equal variance was greater than .05 (0.651 and 0.096, not shown) for both variables, equal variance could be assumed.

While the results indicated that the performance of the treatment group was slightly higher than that of the control group (6.35 vs. 5.93, see Table 2), the difference was not statistically significant (p=0.591, see Table 4). H3 is therefore not supported. The treatment group did, however, take nearly twice as long to complete the tutorial due to the usage of the additional tools (11.94 min. vs. 7.53 min, p=0.000, see Tables 3 and 4). H4 was therefore supported.

We originally had intended to examine whether the preferred learning style of a user had any impact on the perceived usefulness of the tools; however, since everyone but 2 observers in the control group were of the explorer type, this analysis could not be conducted.

The perceived usefulness and ease of use were measured by a customized TAM questionnaire (see Appendix). The questions used in this instrument were directly derived from the original TAM questionnaire (Davis, 1989), therefore, there was no need to validate the instrument.

Questions T1-T4 measured the perceived usefulness of the respective tutorial. Although in all items (T1-T4), the perceived usefulness of the control group was higher than the treatment group, this difference was never statistically significant (see Table 5). The same was true for perceived ease of use (T5-T8, see Table 6). Neither H1 nor H2 were therefore supported.

6. DISCUSSION

The goal of this limited experiment was to get an understanding of the effectiveness of the two learning tools PROGSIM and PROGCDT. The results are initially not very encouraging. Although the treatment group performed slightly better than the control group, they needed significantly more time to use the tools, and in the TAM evaluation, they did not perceive the tools to be of significantly higher usefulness. A positive point is that the ease of use rankings of the treatment group were fairly close to the rankings of the static-web based tutorial of the control group.

An explanation for the relative poor effectiveness of the tools could be that the subject matter taught was too simplistic to elucidate their full potential. It may be that the use of the tools in more complex tasks could yield significant difference in both perceived usefulness and actual learning outcome. We intend to carry out an experiment that involves a more complex learning task in the near future.

An inference from this finding is that while theory suggests that tool support for learning may be beneficial, there may be a threshold of complexity below which the value of tool support is marginal or even potentially harmful.

7. REFERENCES

Agarwal, R., and Prasad, J., A Conceptual and Operational Definition of Personal Innovativeness in the Domain of Information Technology, Information Systems Research, 9, 2, (1998), 204-215.

Alavi, Maryam and Dorothy E. Leidner: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues, Vol. 25, No. 1, March 2001

Chircu, Alina M, Kauffman, Robert J, Limits to value in electronic commerce-related IT investments, Journal of Management Information Systems, 17,2, (2000), 59-80

Corbett, A. and Anderson, J. R. "Student Modeling and Mastery Learning in a Computer-Based Programming Tutor," presented at Second International Conference, ITS ´92, Montreal, Canada, 1992.

Davis, F.D., "Perceived Usefulness, Perceived Ease-of-Use and User Acceptance of Information Technology," MIS Quarterly, Vol. 13 No. 3, 1989, pp. 319-339.

Davis, F.D., Bagozzi, R.P, and Warshaw, RR., User Acceptance of Computer Technology: Comparison of Two Theoretical Models, Management Science, 35, 8, (1989), 982-1003.

Emurian, H. "A programmed instruction tutoring system for JavaTM: consideration of learning performance and software self-efficacy," *Computers in Human Behavior* 20 (2004) 423–459

Gallivan, Michael, Organization Adoption and Assimilation of Complex Technological Innovations: Development and Application of a New Framework, Database for Advances in Information Systems, 32,3,(2001), 51-85

Gill, T.G. "Teaching Flowcharting with FlowC," *Journal of Information Systems Education*, (15:1), pp. 65-77

Janicki, T., Liegle, J.O. (2001A). Development and evaluation of a framework for creating web-based learning modules: a pedagogical and systems perspective, Journal of Asynchronous Learning Networks, Summer 2001.

Janicki, T., Liegle, J.O. (2001B). More than class notes? A features review of current web-based course management tools and their adherence to accepted learning pedagogy, Journal of Allied Academies, Summer 2001.

Kolb, D. (1976). Learning Style Inventory, Self-Scoring Test and Interpretation booklet. Boston, MA: McBer and Company.

Liegle, J. and Janicki, T. "The effect of learning styles on the navigation needs of Web-based learners" Computers in Human Behavior, In Press, 10 April 2004

Karahanna, E., Straub, D.W., and Chervany, N.L., Information Technology Adoption Across Time: A Cross-Sectional Comparison of Pre-Adoption and Post-Adoption Beliefs, MIS Quarterly, 23, 2, (1999), 183-213.

Romero, J.E.; Tepper, B.J.; Tetrault, L.A. (1992). Development and Validation of New Scales to Measure Kolb's Learning Style Dimensions. Educational and Psychological Measurement, 52, 171-180.

Straub, Detmar W., Mark Keil and Walter Brenner, Testing the Technology Acceptance Model across Cultures: A Three Country Study, Information & Management, 31, 1, (1997), 1-11

Szajna, B., Empirical Evaluation of the Revised Technology Acceptance Model, Management Science, 42, 1, (1996), 8592.

Thompson, R.L., Higgins, C.A., and Howell, J.M., Personal Computing: Toward a Conceptual Model of Utilization, MIS Quarterly, 15, 1, (1991), 124-143.

TABLES

Table 1: Pilot Results:

Statement	Avg.	Changes made
Overall, the quality of the content was high	2.0	Typos and bugs were fixed
Overall, the difficulty level of the content was appropriate for you	2.6	No changes made.
I found the examples very helpful	1.7	Added more
I found the demonstrations very helpful	1.7	Added one to every code sample shown
Overall, the difficulty level of the quizzes was appropriate	2.4	Also, too easy. No change besides typos.
Overall, the quiz questions tested the current content	1.9	Some bugs were fixed.
The free-style programming tool was easy to use (PROGSIM)	2.0	Improved instructions.
Using this tool improved my learning of programming	2.3	Relatively low score. We assume that students did not use it as much
In general, I find such a tool to be useful	1.8	No problems here.
The code demonstration tool was easy to use (PROGSIM)	1.4	No problems here.
Using this tool improved my learning of programming	1.6	Added a simulation to every code sample we showed.
In general, I find such a tool to be useful	1.5	No problem here.

Note: 1 (Strongly Agree) to 5 (Strongly Disagree)

Table 2: Descriptive Statistics for Subjects

							Skewness		Kurtosis	
	N	Min	Max	Mean	Std. Dev	Var	Statistic	Std. Error	Statistic	Std. Error
Learning Improvement	32	1	10	6.16	2.16	4.652	-.483	.414	-.130	.809
Time	32	5	17	9.87	3.26	10.629	.345	.414	-.813	.809

Table 3: Descriptive Statistics For Treatment Groups

	Group	N	Mean	Std. Dev.	Std. Err
Learning Improvement	1	15	5.93	2.28	.59
	2	17	6.35	2.09	.51
Time	1	15	7.53	1.92	.50
	2	17	11.94	2.77	.67

Table 4: Independent Samples-T-Test

	T	df	Sig. (2-tailed)	Mean Diff.	Std. Error
Learning Improvement	-.543	30	.591	-.42	.77
Time	-5.157	30	.000	-4.41	.85

Table 5: ANOVA For Perceived Usefulness Questions

		Sum of Squares	df	Mean ^{2}	F	Sig.
T1	Between Groups	4.4E-02	1	.044	.041	.842
	Within Groups	32.675	30	1.089		
	Total	32.719	31			
T2	Between Groups	4.9E-04	1	.000	.000	.984
	Within Groups	37.875	30	1.262		
	Total	37.875	31			
T3	Between Groups	.344	1	.344	.273	.605
	Within Groups	37.875	30	1.262		
	Total	38.219	31			
T4	Between Groups	7.1E-02	1	.072	.069	.795
	Within Groups	30.122	29	1.039		
	Total	30.194	30			

Table 6: ANOVA for perceived Ease-Of-Use Questions

		Sum of Squares	df	Mean ^{2}	F	Sig.
T5	Between Groups	4.9E-02	1	.049	.054	.819
	Within Groups	27.451	30	.915		
	Total	27.500	31			
T6	Between Groups	.567	1	.567	.487	.491
	Within Groups	34.933	30	1.164		
	Total	35.500	31			
T7	Between Groups	7.8E-03	1	.008	.008	.928
	Within Groups	27.992	30	.933		
	Total	28.000	31			
T8	Between Groups	4.9E-02	1	.049	.058	.812
	Within Groups	25.451	30	.848		
	Total	25.500	31			

Appendix

Instructions – Version PL [WITH TOOL] Experiment ID#:_____

The following is an exercise-experiment that teaches you some basic programming principles in java script.

Since students receive different types of instructions, you will be seated in different sections and receive different instructions.

Please – treat this exercise like an exam – no open books, no discussion, no peeking.

Your participation does not directly influence your grade, i.e. your performance on pre-and post-tests are anonymously recorded.

The subject matter taught, JavaScript, however is very course related and I reserve the right to test you on that material in later exams.

Please answer honestly, read all questions carefully, and make sure to record the time (using either your watch or the computer's watch – just use the same in all cases).

When you have completed this exercise, please hand the material to the instructor and take a short break as instructed. This exercise involves the following steps:

1. **Take a pre-test**
2. **Fill out a questionnaire**
3. **Record the current time**
4. **Take an online tutorial**
5. **Record the current time (to allow for calculation on how much time you spent on the tutorial)**
6. **Evaluate the tutorial**
7. **Take a post test (to allow us to determine what you learned)**

Please: Do not go back to change any previous answers!

Pretest: This pretest assesses your current knowledge of Javascript.
Circle your response

Question 1: In what sequence should you analyze an Input-Processing-Output problem?
a) Input-Processing-Output
b) Processing-Output-Input
c) Output-Processing-Input
d) All of the above

Question 2: What is a constant?
a) a special type of variable
b) a non-changing value
c) its value is determined by the programmer
d) all of the above

Question 3: What is the Javascript command for user Input ?
a) enter()
b) prompt()
c) input()
d) read()

Question 4: What is the Javascript command for user output?

a) alert()
b) display()
c) output()
d) messagebox()

Question 5: What is the result of "1" + "2"?

a) "12"
b) 12
c) 3
d) "1+2"

Question 6: Write a small Javascript program that displays the total number of visitors based on the number of male visitors and female visitors as given by the user.

Personality Type Questionnaire

The following example shows you a pair of statements, with a scale from 1 to 7 in-between (see below). The numbers in-between can be seen as a range – whether you lean more towards one statement or the other statement.

Example:

1	I would describe myself as smart	1 ② 3 4 5 6 7	I would describe myself as generous

In the example above, the user decided that he/she is more smart than generous. These questions are taking from a commonly accepted instrument that measures your preferred learning style. **We will never make the results available to anyone else, there is no "best" score, and no learning style is known to be better than another** – all we want to find out through these questions is how we can present that tutorial material "better."

Please, take your time, think carefully, and circle your selection

1	I would describe myself as impartial (open minded)	1 2 3 4 5 6 7	I would describe myself as explicit (definite)
2	I would describe myself as reflective	1 2 3 4 5 6 7	I would describe myself as action-oriented
3	I like to be specific	1 2 3 4 5 6 7	I like to remain flexible
4	I value patience	1 2 3 4 5 6 7	I value getting things done
5	I like things to be varied and colorful	1 2 3 4 5 6 7	I like things to be exact and precise
6	I would describe myself as a doer	1 2 3 4 5 6 7	I would describe myself as an observer
7	I take a creative and imaginative approach to solving problems	1 2 3 4 5 6 7	I take a precise and calculated approach to solving problems
8	I feel good when I understand things	1 2 3 4 5 6 7	I feel good when I have an impact on things
9	I like to stay flexible (not get too focused)	1 2 3 4 5 6 7	I like to get as focused as possible
10	I am good at getting things accomplished	1 2 3 4 5 6 7	I am good at seeing things from many perspectives
11	I would describe myself as evaluative and logical	1 2 3 4 5 6 7	I would describe myself as receptive and accepting
12	I like to watch what is going on	1 2 3 4 5 6 7	I like to see the results of my actions
13	I strive for versatility	1 2 3 4 5 6 7	I strive for accuracy
14	I am reserved	1 2 3 4 5 6 7	I am prepared

Personal Information (Anonymous)

1) Business Work experience (yrs) ☐0 ☐1-2 ☐3-4 ☐5-6 ☐6-8 ☐9-10 ☐>10

2) Programming experience (yrs) ☐0 ☐1-2 ☐3-4 ☐5-6 ☐6-8 ☐9-10 ☐>10

3) GPA ☐<2.5 ☐2.5-3.0 ☐3.1-3.5 ☐3.6-4.0

4) Gender ☐Female ☐Male

5) Major: ☐MBA-CIS ☐MS-CIS ☐PHD-CIS ☐Other:_____

6) Age ☐<20 ☐20-25 ☐26-30 ☐31-35 ☐36-40 ☐41-45 ☐46-50 ☐>50

7) CIS programming courses currently or previously taken at GSU/other institutions

☐ 8110 - ASP
☐ C/C++ Intro
☐ C/C++ Intermediate
☐ C/C++ Advanced
☐ VB Intro
☐ VB Advanced
☐ Java
☐ JavaScript
☐ Other languages (specify):

Please read the following instructions carefully and then follow them one by one.

1. Record the **current time** using the computer/your watch: _____

2. Once you have recorded the time, please go to the following URL:

   `http://cis.gsu.edu/~jliegle/PL/IPO-PL.html`

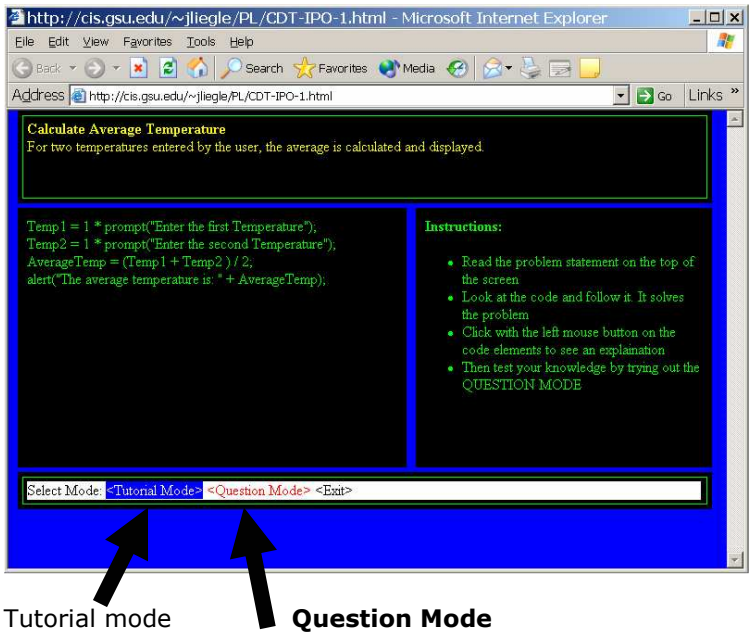
   (case sensitive. If you have problems, notify the instructor)

   This is a short tutorial. Please study the material within the next minutes, take your time. If applicable, go to linked material as well.

   !!! DO NOT MAKE CHANGES TO THE PRE-TEST !!!!

   Once you have completed the tutorial, **exit the browser.**

You will find two tools, a program simulator and a explaination/self test tool (see below)



Tutorial mode **Question Mode**

This tool has two modes – **Tutorial mode** (click on line and see explanation at the right), And **Question Mode** (See question on the right, answer appropriately. Open ended, you stop when you feel you understand what is happening)

Once you have completed the tutorial,

3. Record the **current time again** using the computer/your watch: _____

NOW EXIT THE BROWSER – NO PEEKING !!!

4. Continue on the next page

Did you exit the browser ???

Tutorial Evaluation:

Please evaluate the tutorial with the understanding
that the presented material was limited in scope.

Please use the score-table on the right of the statements below to evaluate the tutorial

		1-Strongly Disagree 5-Strongly Agree
1	Using the tutorial improves my effectiveness	(1) (2) (3) (4) (5)
2	Using the tutorial improves my performance	(1) (2) (3) (4) (5)
3	Using the tutorial increases my productivity,	(1) (2) (3) (4) (5)
4	I find the tutorial useful.	(1) (2) (3) (4) (5)
5	Learning to use the tutorial is easy for me	(1) (2) (3) (4) (5)
6	I find it easy to get the tutorial to do what I want	(1) (2) (3) (4) (5)
7	It would be easy for me to become skillful at using the tutorial	(1) (2) (3) (4) (5)
8	I find the tutorial easy to use	(1) (2) (3) (4) (5)

Continue on next page

Post-test: Please – do not open the browser again and do not look at the pre-test
Circle your response

Question 1: In what sequence should you analyze an Input-Processing-Output problem?
a) Input-Processing-Output
b) Processing-Output-Input
c) Output-Processing-Input
d) All of the above

Question 2: What is a constant?
a) a special type of variable
b) a non-changing value
c) its value is determined by the programmer
d) all of the above

Question 3: What is the Javascript command for user Input ?
a) enter()
b) prompt()
c) input()
d) read()

Question 4: What is the Javascript command for user output?
a) alert()
b) display()
c) output()
d) messagebox()

Question 5: What is the result of "1" + "2"?
a) "12"
b) 12
c) 3
d) "1+2"

Question 6: Write a small Javascript program that displays the total number of visitors based on the number of male visitors and female visitors as given by the user.