Impact of Alternative Introductory Courses on Programming Concept Understanding

Allison Elliott Tew, W. Michael McCracken, Mark Guzdial
College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, GA 30332-0280
{allison.tew, mike.mccracken, mark.guzdial}@cc.gatech.edu

ABSTRACT
Computer science has long debated what to teach in the introductory course of the discipline, and leaders in our field have argued that the introductory course approach is critical to student development. We investigated the impact of alternative approaches to introductory computing by considering the questions of what students bring to their second class in computing and how the outcomes differ depending on the students' alternative first course. Our study showed significant differences in understanding of introductory concepts, such as iteration, conditionals, and arrays, at the beginning of the second course. However, by the end of the second course their understanding had converged.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – computer science education, curriculum

General Terms
Algorithms, Measurement, Experimentation, Languages.

Keywords
Programming, pedagogy, understanding, CS1, CS2.

1. INTRODUCTION
Computer science has struggled since its inception with what to teach in the introductory course of the discipline. Alan Perlis specified his list of critical topics in the first course and recommended that the course be taken by all university students as part of a liberal education [6]. In more recent years, leaders in our field have argued that the introductory course is so critical that students may be hopelessly lost if students do not have the right kind of experience from the start [4]. The most common debate is whether "objects-first" (teaching object-oriented programming from the very start) is effective or not [1].

ICER'05, October 1–2, 2005, Seattle, Washington, USA.

At the same time, we know that the human mind is amazingly plastic and can learn dramatically at any point in life [2]. Most programmers working today did not learn in an "objects-first" manner, and a great many did learn their first lessons in computing in exactly those curricula now considered "hopelessly lost." What is the real cost and impact of alternative approaches to introductory computing?

One reasonable place to begin studying the cost and impact of alternative approaches is in the second course in the computing curriculum. What do students know coming into their second course? Does the approach of the introductory course influence the outcomes from the second course?

At Georgia Tech, we teach three alternative introductory computing classes [5]: A traditional CS1 aimed at mostly CS majors; a course in MATLAB for engineers; and a third course for liberal arts, management, and architecture students oriented around media computation. In this study, we consider students from the first two courses in our traditional second course. The differences between these two classes are in the realm of *context*, rather than the more controversial issues of when objects are introduced. Both of these courses might be considered "objects-early" in that object-oriented programming is used in the first semester, but not in the first weeks of the semester. Our study considers the questions of what students from each of these classes bring to the second class and of how the outcomes from that second class differ depending on the students' alternative first course.

2. METHODOLOGY
We began by conducting a document analysis of the course materials for the two alternate introductory courses, CS 1321 – Introduction to Computing and CS 1371 – Computing for Engineers and the second course, CS 1322 – Object-Oriented Programming. We analyzed the course syllabi, lecture notes, assignments, and textbooks [3, 7] to determine content that was covered in the introductory sequence of courses.

CS 1321 is a traditional introductory computer science course, taught in Python, emphasizing simple data structures and the design, construction, and analysis of algorithms. The course was designed primarily for CS majors, but it is also a required course for undergraduate students in the College of Sciences, including Mathematics, Psychology, Chemistry and Physics majors. CS 1371 is the introductory course, taught in MATLAB and Java, tailored for students in the College of Engineering. Traditional introductory computer science content, such as data structures and design and analysis of algorithms, are taught in the context of engineering problem-

solving. CS 1322 is the second course in the introductory sequence. It is an object-oriented programming course, taught in Java, emphasizing data structures and an introduction to program design and software engineering.

Our document analysis revealed two sets of topics. A set of introductory topics that were covered in both of the first courses and a set of advanced topics that were covered in the second course. We identified seven introductory topics – conditionals, iteration, arrays, binary search trees, searching, sorting, and recursion. And we identified 6 advanced topics – object-oriented basics, polymorphism, dynamic binding, linked lists, hash tables, and GUIs. We informally evaluated content validity by having the list of identified topics reviewed by two expert faculty members.

We developed a set of pre- and post-test instruments to evaluate students' understanding of these topics. An ITiCSE 2004 working group, led by Raymond Lister, conducted a study of basic programming skills of beginning students by giving students a set of Multiple Choice Questions (MCQs) [8]. We adapted their approach and constructed a pre-test comprised of 13 MCQs, one for each of the topics identified above. There were two basic kinds of MCQs – tracing questions, where students were asked to trace the execution of a piece of code, and code completion questions, where students were asked to select and insert the missing lines of code to correctly execute a given algorithm. Examples of each kind of question are shown in Figures 1 and 2. We developed an analogous set of 13 MCQs for the post-test. The full text of the questions analyzed in this paper are in Appendix A, and the complete question sets are available at http://home.cc.gatech.edu/allison/2.

```
ARRAY

int array1 = { 4, 5, 3, 6, 2, 7, 1 };
int array2 = { 7, 4, 2, 1 };

array1[ 3 ] = array1[ 5 ];
array1[ 2 ] = array2[ 2 ];
array1[ 4 ] = array2[ 3 ] + 5;
array1[ 6 ] = array1[ 3 ];
if ( array1[ 1 ] > array2[ 1 ] )
      array1[ 1 ] += 2;

What is the value of array1 after this code is
executed?

a)   { 4, 4, 2, 7, 2, 2, 1}
b)   { 4, 7, 3, 6, 7, 7, 7}
c)   { 4, 7, 2, 7, 6, 7, 7}
d)   { 4, 7, 2, 7, 2, 2, 1}
e)   { 4, 7, 2, 6, 6, 7, 6}
```

Figure 1. Sample Tracing Question - Arrays

3. DATA & ANALYSIS

Participants in our study were Georgia Tech undergraduate students enrolled in the second course of the introductory computing sequence, and their participation was voluntary. The pre-test was administered during the first week of the semester to 177 total participants, and the post-test was administered during the last week of the semester to 88 total participants. The voluntary nature of the population could bias the sample, but our analysis, based on the demographic

```
SORTING

The following method "isSorted" should return true if
the array " x " is sorted in ascending order.
Otherwise, method should return false:

public static Boolean isSorted( int[] x )
{
         //missing code
}

Which of the following code fragments is the missing
code?

a) boolean b = true;
   for ( int i = 0; i < x.length – 1; i++ )
   {
         if ( x[ i ] > x[ i + 1 ] )
               b = false;
         else
               b = true;
   } return b;

b) for ( int i = 0; i < x.length – 1; i++ )
   {
         if ( x[ i ] > x[ i + 1 ] )
               return false;
   } return true;

c) boolean b = false;
   for ( int i = 0; i <  x.length – 1; i++ )
   {
         if ( x[ i ] > x[ i + 1 ] )
               b = false;
   } return b;

d) boolean b = false;
   for ( int i = 0; i <  x.length – 1; i++ )
   {
         if ( x[ i ] > x[ i + 1 ] )
               b = true;
   } return b;

e) for ( int i = 0; i <  x.length – 1; i++ )
   {
         if ( x[ i ] > x[ i + 1 ] )
               return true;
   } return false;
```

Figure 2. Sample Code Completion Question – Element Comparison for Sorting

information we collected, suggests that the sample is representative of the class as a whole. Our analysis will focus on the participants who completed one of the introductory courses (CS 1321 or CS 1371) at Georgia Tech. We had a small number of participants (n=13 pre-test, n=9 post-test) who had placed out of the introductory course via the high school CS Advanced Placement (AP) Exam or had earned credit by transferring coursework from a previous institution. During the administration of the post-test we discovered an error in the polymorphism MCQ, so that question has been excluded from the pre- and post-test data analysis.

3.1 Pre-Test

Participants were asked to complete a short survey about their major, prerequisite course information, and previous programming experience and were asked to answer 13 MCQs on the introductory and advanced topics. Our analysis of the pre-test data focuses only on the introductory topics that all the students have some familiarity with from their first CS course. Data about their performance on the advanced topics is presented as a baseline for comparison with the post-test. 55

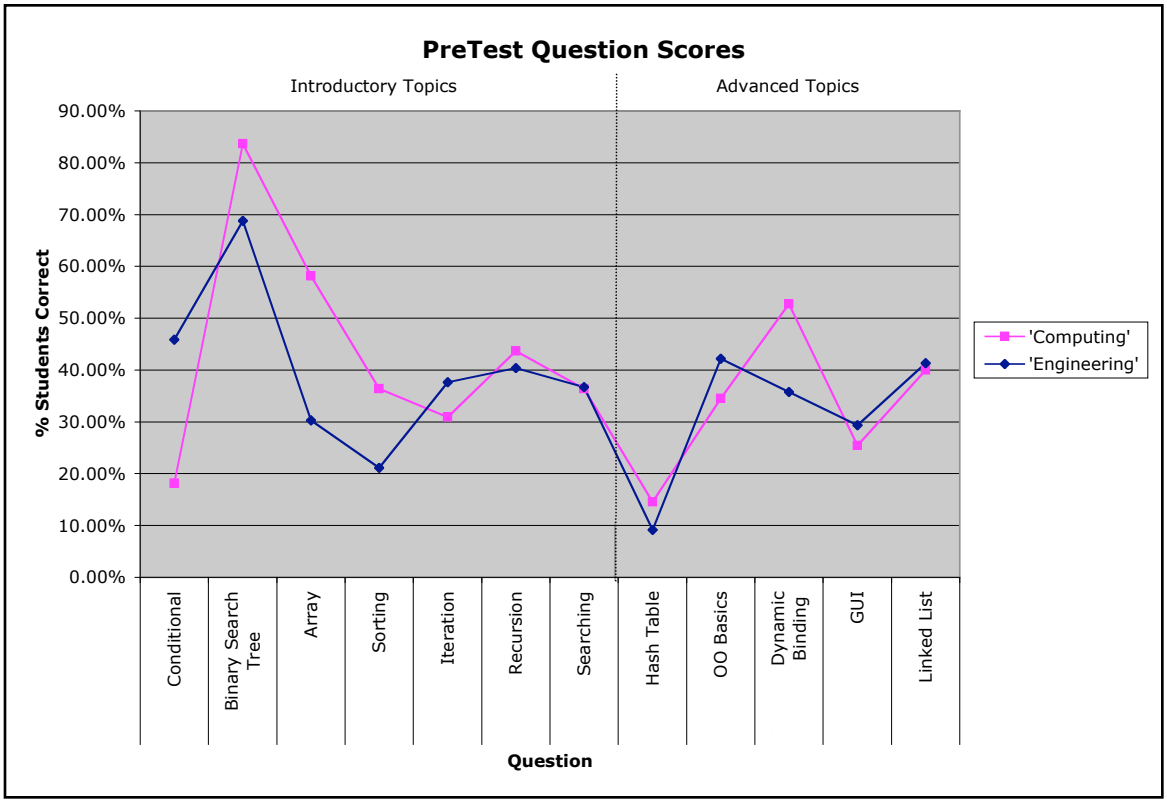


Figure 3. Percentage of students in each participant group answering pre-test questions correctly.

participants had completed the "computing" introductory course, CS 1321, and 109 participants has completed the "engineering" introductory course, CS 1371. Participants in the 'computing' group were majoring in Computer Science (n=41), Industrial & Systems Engineering (n=4), Computer Engineering (n=3), Computational Media (n=2), Electrical Engineering (n=2), Math (n=1), Discrete Math (n=1), and Psychology (n=1). Participants in the 'engineering' group were majoring in Electrical Engineering (n=56), Computer Engineering (n=27), Industrial & Systems Engineering (n=16), Computer Science (n=5), Aerospace Engineering (n=4), and Mechanical Engineering (n=1).

The students who took the pre-test demonstrated a better understanding of the introductory topics than the advanced topics. An average of 42.29% of the students answered the introductory topic MCQs correctly while an average of only 32.32% of the students answered the advanced topic MCQs correctly. The overall performance of the students on an individual question basis is shown in Figure 3.

On the pre-test the average individual student score was 4.58 ($\partial = 2.55$), with a median score of 4 out of a total of 12 possible points. Participants in the 'engineering' and 'computing' groups had similar overall scores on the test. The 'computing' group average individual student score was 4.75 and the 'engineering' group average was 4.39. One student earned a perfect score and 3 students did not answer any questions correctly. The question that was answered correctly most often was the binary search tree MCQ with 74.01% of the students answering this question correctly. Out of the introductory topics, the sorting question proved to be most difficult with only 29.76% of the students answering this question correctly.

3.1.1 'Engineering' Performance

Students who had completed the 'engineering' introductory course demonstrated significantly better understanding of the conditional topic on the pre-test.

The conditional MCQ asked students to trace a sequence of boolean and math conditional operations to perform a set of calculations on the variables x and y. (see appendix A.1) Students who correctly traced the code evaluated two nested if-else statements followed by another if statement to reach the correct answer of E (see Table 1.) Students in the 'engineering' group were over 25% more likely to answer the question correctly. Our chi-squared analysis yielded a significance at the $\alpha = 0.001$ significance level.

Table 1. Pre-Test Conditional MCQ

	Q2					
Class	A	B	C	D	E	blank
'Computing'	10.91%	9.09%	54.55%	1.82%	18.18%	5.45%
'Engineering'	5.50%	1.83%	44.04%	1.83%	45.87%	0.92%
Total	7.32%	4.27%	47.56%	1.83%	36.59%	2.44%

Our analysis shows that students in both participant groups were likely to chose incorrect choice C – 44% of the 'engineering' group and 55% of the 'computing' group did so. Distracter C was designed to catch those who incorrectly evaluated the first mathematical conditional, when x and y had been initialized to 2 and 16 respectively,

```
if ( x < y )
        x = x * x;
else
        y = x + y;
```

to false and executed the else clause. Distracter A was also chosen by almost 11% of the students in the 'computing' group. Distracter A was designed to catch those who incorrectly evaluated

```
if ( (true && false ) || true )
```

to false and did not execute the statements included in the if clause.

Students in the 'engineering' and 'computing' groups showed different patterns in answering the question, most noticeably by how frequently the correct answer was chosen. A chi-squared analysis revealed a significant difference at the α = 0.001 significance level.

3.1.2 'Computing' Performance

Students who had completed the 'computing' introductory course demonstrated significantly better understandings of the binary search tree, array, and sorting topics.

The binary search tree MCQ presented a class definition of a binary search tree with an accompanying figure representing an instance of a binary search tree with numerical nodes (see appendix A.2). Students were asked to trace code representing a pre-order traversal of the tree, although the specific traversal was not identified by name. Students in the 'computing' group were almost 15% more likely to choose the correct answer of D over the students in the 'engineering' group (see Table 2). Our chi-squared analysis yielded significance at the α = 0.05 significance level.

Table 2. Pre-Test Binary Search Tree MCQ

	Q3					
Class	A	B	C	D	E	blank
'Computing'	1.82%	1.82%	12.73%	83.64%	0.00%	0.00%
'Engineering'	1.83%	3.67%	19.27%	68.81%	3.67%	2.75%
Total	1.83%	3.05%	17.07%	73.78%	2.44%	1.83%

Our analysis shows that choice C was the most common incorrect answer for students in both participant groups – 13% of the 'computing' group and 19% of the 'engineering' group selected this answer. Distracter C was designed to catch those who incorrectly evaluated the traversal and simply listed the nodes of the tree as read logically from top to bottom in the diagram.

Students in the 'computing' and 'engineering' groups showed similar patterns in answering the question and a chi-squared analysis revealed no significant difference.

In the array MCQ, as seen in Figure 1, students were given instances of two arrays – array1 and array2, and were asked to perform a set of calculations and manipulations on individual array elements and determine the value of array1 after all the commands had been executed (see appendix A.3). Students in the 'computing' group, were over 25% more likely to answer the question correctly, choice C (see Table 3). Our chi-squared analysis yielded a significance at the α = 0.001 significance level.

A majority of the students (57%) of the participants in the 'engineering' group chose incorrect choice A. Distracter A was designed to catch those who incorrectly indexed the arrays beginning with 1, not 0. Choices B and E were the incorrect answers most commonly given by students in the 'computing' group. Distracter E was designed to catch those who did not evaluate the first assignment statement

Table 3. Pre-Test Array MCQ

	Q4					
Class	A	B	C	D	E	blank
'Computing'	9.09%	10.91%	58.18%	7.27%	10.91%	3.64%
'Engineering'	56.88%	1.83%	30.28%	4.59%	3.67%	2.75%
Total	40.85%	4.88%	39.63%	5.49%	6.10%	3.05%

```
array1[ 3 ] = array1 [ 5 ];
```

and distracter B was generated semi-randomly.

Students in the 'computing' and 'engineering' groups showed different patterns in answering the question, most noticeably by which distracters were most frequently chosen. A chi-squared analysis revealed a significant difference at the α = 0.001 significance level.

The sorting MCQ (see in Figure 2 and appendix A.4) is a code completion question. The question focused on the first part of the sorting algorithm – comparing the value of elements in an array. Students are asked to select the correct code to evaluate whether an array, passed as an input parameter, is sorted in ascending order. Students who correctly answered this question iterated over the entire array as follows.

```
for (int i = 0; i < x.length – 1; i++ )
{
        if ( x[ i ] > x[ i + 1 ])
                return false;
} return true;
```

While neither student group performed particularly well on this question, the 'computing' students were over 15% more likely to answer the question correctly, choice B (see Table 4). Our chi-squared analysis yielded a significance at the α = 0.025 significance level.

Table 4. Pre-Test Sorting MCQ

	Q6					
Class	A	B	C	D	E	blank
'Computing'	43.64%	36.36%	7.27%	5.45%	7.27%	0.00%
'Engineering'	51.38%	21.10%	8.26%	11.93%	4.59%	2.75%
Total	48.78%	26.22%	7.93%	9.76%	5.49%	1.83%

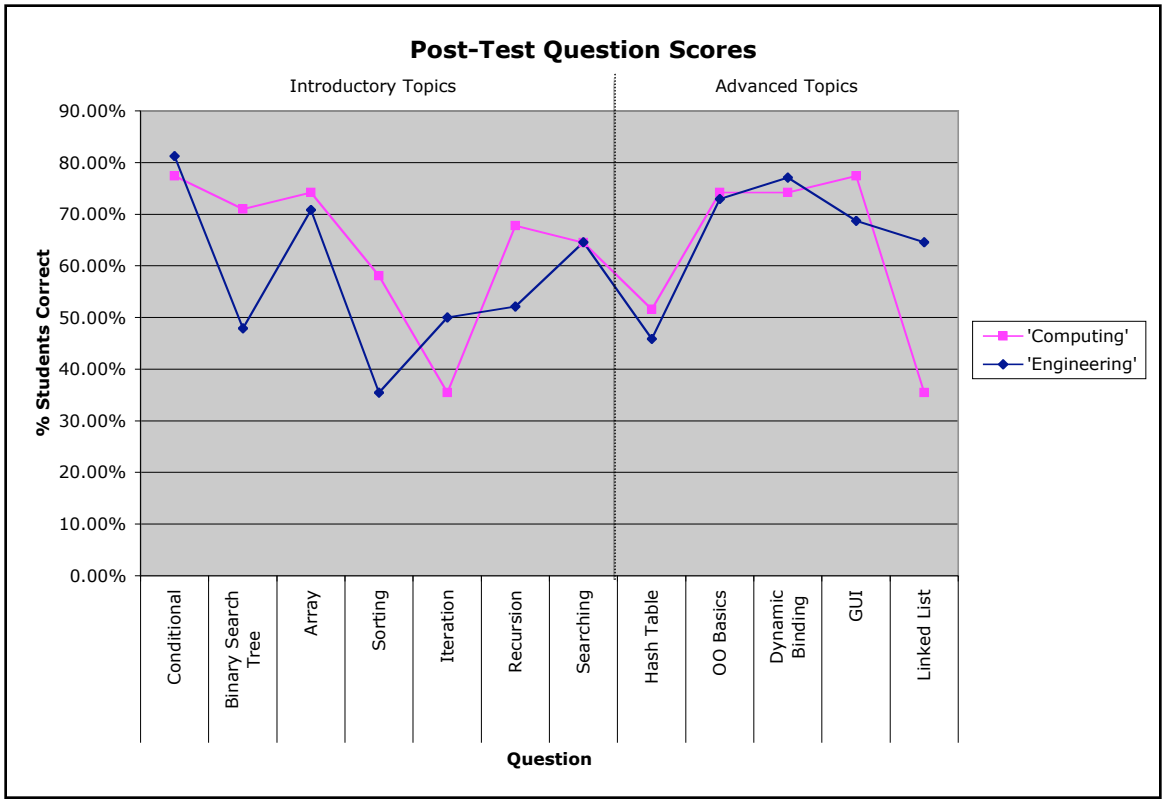


Figure 4. Percentage of students in each participant group answering post-test questions correctly.

Our analysis shows that students in both participant groups were likely to chose incorrect choice A – 44% of the 'computing' group and 52% of the 'engineering' group did so. Distracter A was designed to catch those who correctly evaluated the elements of the array for each iteration but failed to aggregate the evaluations over the entire array.

```
for (int i = 0; i < x.length – 1; i++ )
{
        if ( x[ i ] > x[ i + 1 ])
                b = false;
        else
                b = true;
} return b;
```

Answer A simply returns the value of the evaluation for the last two elements of the array. Incorrect answer D was also chosen by 12% of the participants in the 'engineering' group. Distracter D was designed to catch those who incorrectly set the boolean to true upon the following evaluation of the elements of the array.

```
if ( x[ i ] > x[ i + 1 ])
        b = true;
```

The boolean should have been initialized to true and then set to false upon the evaluation above. Our results are consistent with the findings on the same question in the original Lister study. [8]

Students in the 'computing' and 'engineering' groups showed similar patterns in answering the question and a chi-squared analysis revealed no significant difference.

3.1.3 Similar Performance

Students demonstrated similar understandings of the iteration, recursion, and searching introductory topics. On the iteration MCQ, 30.91% of the participants in the 'computing' group answered the question correctly and 37.61% of the 'engineering' group answered it correctly. 43.64% of the 'computing' participants and 40.37% of the 'engineering' participants answered the recursion MCQ correctly. The searching MCQ question was answered correctly by 36% of the students in both participant groups. Our chi-squared analyses revealed no significant differences in the levels of understandings on any of these topics.

3.2 Post-Test

On the post-test, participants were asked to complete a short survey about their major and pre-requisite course information and were asked to answer 13 MCQs on the introductory and advanced topics. There were 31 participants in the 'computing' group majoring in Computer Science (n=17), Electrical Engineering (n=6), Math (n=3), Computer Engineering (n=2), Computational Media (n=2), and Industrial & Systems Engineering (n=1). There were 48 participants in the 'engineering' group majoring in Electrical Engineering (n=29), Computer Engineering (n=10), Industrial & Systems Engineering (n=5), Aerospace Engineering (n=3), and Computer Science (n=1).

Students demonstrated improved understanding on almost all of the topics when compared to their pre-test levels. An average of 61.53% of the students answered the introductory topic MCQs correctly and an average of 64.77% of the students

answered the advanced topic MCQs correctly. The overall performance of the students on an individual question basis is shown in Figure 4.

On the post-test the average individual student score was 7.55 (∂ = 2.74), with a median score of 8 out of a total of 12 possible points. Participants in the 'engineering' and 'computing' groups had similar overall scores on the post-test as well. The 'computing' group average individual student score was 7.61 and the 'engineering' group average was 7.31. One student earned a perfect score and one student only answered 1 question correctly. The question that was answered correctly most often was the conditional MCQ with 79.55% of the students answering this question correctly, while the loops and hash table questions proved to be most difficult with only 46.59% of the students answering these questions correctly.

Our analysis of the post-test results revealed that there is now no significant difference in understanding on any of the introductory topics between the 'computing' and 'engineering' participant groups.

3.2.1 'Engineering' Performance

Students who had completed the 'engineering' introductory course demonstrated significantly better understanding on the linked list topic on the post-test.

For the linked list MCQ students were given the class definition of a linked list and instances of "head" and "position" nodes (see appendix A.5). In a code completion question, students were asked to select the correct piece of code to insert "position" into the beginning of the list, while maintaining head as the pointer to the beginning of the list. Students who correctly answered the question selected the following code

```
position.next = head;
head = position;
```

Students in the 'engineering' group were almost 30% more likely to choose the correct answer of C over the students in the 'computing' group (see Table 5). Our chi-squared analysis yielded significance at the α = 0.001 significance level.

Table 5. Post-Test Linked List MCQ

	Q14					
Class	A	B	C	D	E	blank
'Computing'	6.45%	3.23%	35.48%	19.35%	32.26%	3.23%
'Engineering'	4.17%	6.25%	64.58%	6.25%	16.67%	2.08%
Total	5.06%	5.06%	53.16%	11.39%	22.78%	2.53%

Our analysis shows that more students in the 'computing' group chose incorrect choice E than in the 'engineering' group – 17% of the 'engineering' group and 32% of the 'computing' group did so. Distracter E was designed to catch those who incorrectly reset the position pointer as follows

```
position = head.next
```

without realizing the need to maintain the position's pointer to its original node. Distracter D was also chosen by almost 19% of the students in the 'computing' group. Distracter D was designed to catch those who incorrectly reset the position pointer as above and also incorrectly set the head pointer as follows.

```
head = position.next
```

Students in the 'engineering' and 'computing' groups showed slightly different patterns in answering the question, most noticeably by how often the correct answer was selected. However, a chi-squared analysis revealed no significant difference in their answer pattern.

3.2.2 Similar Performance

Students demonstrated no significant differences in their understandings of the remaining introductory and advanced topics on the post-test. Students in the 'engineering' group were slightly more likely to answer the iteration question correctly and students in the 'computing' group were slightly more likely to answer the binary search tree, sorting, and recursion questions correctly. Our chi-squared analyses revealed no significant differences in the levels of understandings on any of these topics.

The binary search tree MSQ was analogous to the pre-test question. In this tracing question, students were asked to trace code representing a post-order traversal of the tree, although the specific traversal was not identified by name. On the BST MCQ, Q3, more students (70.97%) in the 'computing' group chose the correct answer than in the 'engineering' group (47.92%) (see Table 6). Choice D was the most common incorrect answer for both participant groups. Distracter D was designed to catch those who incorrectly evaluated the traversal and performed a pre-order traversal on the tree.

Table 6. Post-Test Binary Search Tree MCQ

	Q3				
Class	A	B	C	D	E
'Computing'	70.97%	0.00%	0.00%	25.81%	3.23%
'Engineering'	47.92%	4.17%	16.67%	29.17%	2.08%
Total	56.96%	2.53%	10.13%	27.85%	2.53%

The post-test element comparison for sorting MCQ asked students to select code that correctly evaluates whether an array, passed as an input parameter, is sorted in descending order, analogous to the ascending order question from the pre-test. Students continued to struggle with this topic with 58.06% of the 'computing' students answering the question correctly and 35.42% of the 'engineering' students answering it correctly (see Table 7). As in the pretest, choices A and D were the most common mistakes, this time for both participant groups.

Table 7. Post-Test Sorting MCQ

	Q6					
Class	A	B	C	D	E	blank
'Computing'	22.58%	58.06%	3.23%	12.90%	3.23%	0.00%
'Engineering'	29.17%	35.42%	6.25%	22.92%	4.17%	2.08%
Total	26.58%	44.30%	5.06%	18.99%	3.80%	1.27%

In the iteration MCQ students were asked to trace the execution of a while loop while computing a "limit" variable and were asked to return the number of times the loop was executed (see appendix A.6). Students in the 'engineering' group were slightly more likely to select the correct answer, C – 50.00% of the 'engineering' students answered this question correctly while only 35.48% of the students in the 'computing' group did so (see Table 8).

Table 8. Post-Test Iteration MCQ

	Q8				
Class	A	B	C	D	E
'Computing'	3.23%	29.03%	35.48%	6.45%	25.81%
'Engineering'	2.08%	22.92%	50.00%	10.42%	14.58%
Total	2.53%	25.32%	44.30%	8.86%	18.99%

Choices B and E were common mistakes for both participant groups. Distracter B was designed to catch an "off-by-one" error where students forgot the last iteration of the loop and distracter E was designed to catch students who confused the loop counter variable "i" with the computed "limit" variable and returned the "limit" variable instead.

In the recursion MCQ students were asked to trace the execution of a recursive function that counted the number of "e"s in a string and returned a computed mathematical value associated with the string (see appendix A.7). Students in the 'computing' group were slightly more likely to select the correct answer, A – 67.74% of the 'computing' students answered this question correctly while only 52.08% of the students in the 'engineering' group did so (see Table 9). Choice D was a common mistake for both participant groups, representing a distracter where the first else clause in the recursive call is always evaluated and the second mathematical calculation is never performed, regardless of whether the input character is an "e" or not. The 'engineering' group also commonly chose incorrect answers B and C. Distracter B was designed to catch students who switched the mathematical operations (one for when the character was an "e" and another when it was not) and distracter C was designed to catch students who initialized the mathematical value to 0 instead of 1, as was specified in the function parameter list.

Table 9. Post-Test Q9 - Recursion

	Q9					
Class	A	B	C	D	E	blank
'Computing'	67.74%	3.23%	6.45%	16.13%	6.45%	0.00%
'Engineering'	52.08%	10.42%	16.67%	10.42%	4.17%	6.25%
Total	58.23%	7.59%	12.66%	12.66%	5.06%	3.80%

Our chi-squared analyses revealed no significant differences in the levels of understandings on any of these topics. Students in the 'engineering' and 'computing' groups showed similar patterns in answering each of these individual questions. Additional chi-squared analyses revealed no significant differences in the patterns of answers on any of these questions.

Students in both participants groups demonstrated nearly identical understandings of the remaining topics. Our analysis showed less than a 10% difference in the percentage of students answering the following MCQs correctly – conditional, arrays, searching, hash tables, object-oriented basics, dynamic binding, and GUIs. Our chi-squared analyses revealed no significant differences in the levels of understandings on any of these topics.

4. DISCUSSION

Our analysis shows that 'engineering' and 'computing' participants have significantly different understandings of four of the seven introductory topics as they begin the second course in the introductory CS sequence: conditionals, binary search trees, arrays, and sorting. In this section, we begin to explore possible explanations for these differences.

On the pre-test 'engineering' participants demonstrated a better understanding on the conditional topic. Our document analysis reveals that while the textbooks contain comparable coverage of conditional statements, and for this particular question the related topic of booleans and logical operators. the 'engineering' course emphasizes boolean logic more explicitly in lecture materials. In particular, the use of standard engineering flow-chart notation reinforces the evaluation of conditional clauses in a representation familiar to the students. The majority of students in the 'engineering' group are also majoring in areas, such as Electrical and Computer Engineering, that are likely to emphasize boolean logic in other areas of their curricula, such as digital circuit design. We believe a combination of the explicit connections to representations used in their major and more experience with boolean logic can explain the differences in performance.

Participants in the 'computing' group demonstrated a better understanding of the binary search tree topic on the pre-test. We believe the key issue here is one of motivation. Unlike many of the other topics presented in the 'engineering' course, trees are presented in the course in a de-contextualized manner. Students are shown the definitions, diagrams, and appropriate algorithms, but in a traditional family tree model. Our analysis shows no effort to connect this topic to any engineering problem solving application, thus the 'engineers' may be less motivated to learn the necessary algorithms to master this topic. Almost 75% of the students in the 'computing' participant group are CS majors, and we expect they already understand the value of learning about more complex data structures.

Students in the 'computing' group also demonstrated a better understanding of the array topic on the pre-test. The 'engineering' students performed poorly on this question and the majority of their errors are attributable to an array indexing misconception. Most 'engineering' students incorrectly attempted to index the first element of an array in Java using an index of 1 instead of 0. The 'engineering' introductory course begins instruction in MATLAB and then transitions to Java to introduce objects and work with more advanced data structures. In MATLAB arrays (i.e. vectors) are indexed beginning with 1. We believe this difference is attributable to the incorrect transfer of knowledge from MATLAB to their current programming language Java.

On the pre-test, the sorting topic was better understood by participants in the 'computing' group. Again we believe the problem to be one of motivation. Sorting is presented in the 'engineering' course in a de-contextualized manner, showing definitions, diagrams, and appropriate algorithms for sorting lists of numbers. Our analysis again shows no effort to connect this topic to any engineering problem solving application, thus the 'engineers' may be less motivated to learn the necessary algorithms to master this topic.

We did *not* find any differences in outcomes in introductory topic understanding on the post-test. Students come out of the

second course with the same basic concepts even when they came they from different introductory approaches. What can we conclude? There may still *be* differences that might appear later in the curricular sequence. For example, the students may have different perspectives on software development process and engineering. But finding convergence among students' understandings in the second course, despite different introductory courses, might suggest that later courses, beyond the introductory sequence, could also address and reduce such differences.

If choices in the introductory course approach are not critical for the learning outcome at the end of the program of study, then we should consider making design choices based on variables other than simply learning. We know that there are design choices that we can make in the introductory course that can have dramatic impacts on retention and diversity [5, 9, 10]. If students can learn from alternative approaches, we can choose approaches that are more inviting and retain more students, rather than those that perhaps are less appealing.

5. FUTURE WORK

Our results indicate that the traditional debates over the pedagogical approach (e.g. objects-first vs. objects-early) for the introductory CS course may not be as important as the community believes. We propose to continue our investigation of the impact of alternative introductory courses, maintaining our focus, for now, on programming concept understanding.

Beginning summer semester 2005, students from the 'media computation' introductory sequence will be taking the second course, and we can add a third alternative introductory course into our analysis. We will conduct an additional document analysis of the 'media computation' course materials and will re-evaluate the topic list accordingly.

Using the results of the document analyses and our preliminary work, including an analysis of the reliability of our initial pre- and post-test instruments, we will revise the pre- and post-test instruments to correct the previously mentioned errors and clarify the questions and distracters. After taking steps to ensure instrument validity, we will repeat the pre- and post-test study during Fall semester 2005.

Our pre-test/post-test study design using MCQs places constraints on the amount of information we can gather about students' understanding of introductory topics. We plan to develop additional instruments and study designs to gather a richer set of information about students' understanding, e.g. open ended questions, participant interviews, think-aloud protocols. These qualitative methods and analyses could be combined with our previous studies to provide a much richer explanation of the impact of an introductory course on programming concept understanding.

6. ACKNOWLEDGMENTS

We would like to thank Boris Goykhman for his assistance in desiging the pre- and post-tests and in collecting data and Leigh Waguespack for her assistance with the statistical analysis. We are also grateful for the cooperation of the faculty teaching the courses where we conducted our studies – John Stasko, Yannis Smaragdakis, and Ashwin Ram.

7. REFERENCES

[1] Bruce, K.B. Controversy on how to teach CS 1: a discussion on the SIGCSE-members mailing list. *SIGCSE Bulletin*, *36* (4). 29-34.

[2] Bruer, J.T. *Schools for thought : a science of learning in the classroom*. MIT Press, Cambridge, MA, 1993.

[3] Deitel, H.M., Deitel, P.J., Liperi, J.P. and Wiedermann, B.A. *Python How to Program*. Prentice-Hall, Inc., Upper Saddle River, NJ, 2002.

[4] Denning, P.J. A debate on teaching computing science. *Communications of the ACM*, *32* (12). 1397-1414.

[5] Forte, A. and Guzdial, M. Motivation and non-Majors in computer science: Identifying discrete audiences for introductory courses." *IEEE Transactions on Education*, to appear 2005.

[6] Greenberger, M. *Computers and the world of the future*. MIT Press, Cambridge, MA, 1964.

[7] Kaplan, D.T. *Introduction to Scientific Computation and Programming*. Brooks/Cole - Thomson Learning, Belmont, CA, 2004.

[8] Lister, R., Adams, E.S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B. and Thomas, L., A multi-national study of reading and tracing skills in novice programmers. in *Working group reports from ITiCSE on Innovation and technology in computer science education*, (Leeds, United Kingdom, 2004), ACM Press, 119-150.

[9] Margolis, J. and Fisher, A. *Unlocking the clubhouse : women in computing*. MIT Press, Cambridge, MA, 2002.

[10] McDowell, C., Werner, L., Bullock, H. and Fernald, J., The effects of pair-programming on performance in an introductory programming course. in *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, (Cincinnati, KY, 2002), ACM Press, 38-42.

APPENDIX A

In this section, we include the full text of the seven topic multiple choice questions presented in the paper. Questions were given to the participants one question per page. Formatting of some of the questions has been altered to fit the two-column layout.

A.1 Conditional MCQ (Pre-Test)

CONDITIONAL

```
int x = 2;
int y = 16;
if ( ( true && false ) || true )
       {
       if ( x < y )
              x = x * x;
       else
              y = x + y;
       }
if ( ( false || true ) && false )
       {
       if ( x <= y )
              x++;
       else
              y = y + y;
       }
else
       x = x * x;
if ( x <= y )
       y = x + y;
System.out.println("x = " + x);
System.out.println("y = " + y);
```

What is the output of this code fragment?

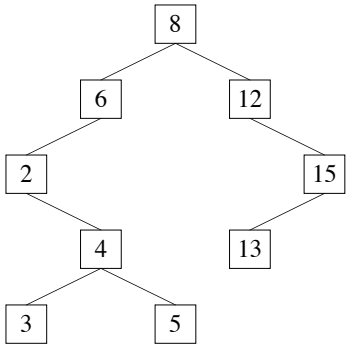
a) x = 4
   y = 22
b) x = 3
   y = 19
c) x = 4
   y = 20
d) x = 4
   y = 23
e) x = 16
   y = 32

A.2 Binary Search Tree MCQ (Pre-Test)

BINARY SEARCH TREE

```
public class Node
{
      private Node left;     // left child node
      private Node right;    // right child node
      private int data;

      public Node getLeft() {
            return left; }
      public Node getRight() {
            return right; }
      public int getData() {
            return data; }
      …
      …
}
public void Traverse (Node root)
{
      if ( root.getData() != null )
            System.out.println(root.getData() );
      if ( root.getLeft() != null )
            Traverse ( root.getLeft() );
      if ( root.getRight() != null )
            Traverse ( root.getRight() );
}
```



What is the output of this Traversal method on the above BST, where 8 is passed as the root?

a) 3, 5, 4, 2, 6, 13, 15, 12, 8
b) 2, 3, 4, 5, 6, 8, 12, 13, 15
c) 8, 6, 12, 2, 15, 4, 13, 3, 5
d) 8, 6, 2, 4, 3, 5, 12, 15, 13
e) 8, 12, 15, 13, 6, 2, 4, 5, 3

A.3 Array MCQ (Pre-Test)

ARRAY

```
int array1 = { 4, 5, 3, 6, 2, 7, 1 };
int array2 = { 7, 4, 2, 1 };

array1[ 3 ] = array1[ 5 ];
array1[ 2 ] = array2[ 2 ];
array1[ 4 ] = array2[ 3 ] + 5;
array1[ 6 ] = array1[ 3 ];
if ( array1[ 1 ] > array2[ 1 ] )
     array1[ 1 ] += 2;
```

What is the value of array1 after this code is executed?

a) { 4, 4, 2, 7, 2, 2, 1}

b) { 4, 7, 3, 6, 7, 7, 7}

c) { 4, 7, 2, 7, 6, 7, 7}

d) { 4, 7, 2, 7, 2, 2, 1}

e) { 4, 7, 2, 6, 6, 7, 6}

A.4 Sorting MCQ (Pre-Test)

SORTING

The following method "isSorted" should return true if the array " x " is sorted in ascending order. Otherwise, method should return false:

```
public static Boolean isSorted( int[] x )
{
       //missing code
}
```

Which of the following code fragments is the missing code?

a)
```
boolean b = true;
for ( int i = 0; i < x.length – 1; i++ )
{
      if ( x[ i ] > x[ i + 1 ] )
           b = false;
      else
           b = true;
} return b;
```

b)
```
for ( int i = 0; i < x.length – 1; i++ )
{
      if ( x[ i ] > x[ i + 1 ] )
           return false;
} return true;
```

c)
```
boolean b = false;
for ( int i = 0; i <  x.length – 1; i++ )
{
      if ( x[ i ] > x[ i + 1 ] )
           b = false;
} return b;
```

d)
```
boolean b = false;
for ( int i = 0; i <  x.length – 1; i++ )
{
      if ( x[ i ] > x[ i + 1 ] )
           b = true;
} return b;
```

e)
```
for ( int i = 0; i <  x.length – 1; i++ )
{
      if ( x[ i ] > x[ i + 1 ] )
           return true;
} return false;
```

A.5 Iteration MCQ (Post-Test)

ITERATION

```
int[] x = { 3 ,2, 5, 6, 8, 4 };
int limit = 11;
int i = 0;
while ( ( 0 < limit ) && ( i < x.length ) )
{
     limit -= x[ i ];
     i++;
}
```

What is the value of the variable " i " after the code is executed?

a) 0

b) 3

c) 4

d) 5

e) –5

A.6 Recursion MCQ (Post-Test)

RECURSION

```
public int Eval(String s, char c, int value)
{
   if (s.length == 0)
       return value;
   else if((s.charAt( 0 ).equals( c ))
   {
       value = value++;
       return Eval(s.substring( 1 ), c, value);
   }
   else
   {
       value = value * 2;
       return Eval(s.substring( 1 ) , c, value);
   }
}
```

What is the value returned by this method call?

```
       Eval( "remember" , 'e' , 1 );
```

a) 58

b) 25

c) 26

d) 9

e) 29

A.7 Linked List MCQ (Post-Test)

LINKED LIST

```
public class ListNode
{
     private ListNode next;
     private String data;
     …
}
```

Assume that a linked list exists and that the variable "head" is used to maintain the beginning of the list. Also assume that the variable "position" and "head" are objects of the class ListNode.

```
Listnode head = new ListNode();
Listnode position = new ListNode();
```

Which of the following code segments correctly adds "position" to the beginning of the list? The code must maintain the "head" variable as the beginning of the list.

a)
```
position = head;
head = position;
```

b)
```
position.next = head.next;
head = position.next;
```

c)
```
position.next = head;
head = position;
```

d)
```
position = head.next;
head = position.next;
```

e)
```
position = head.next;
head = position;
```