Report on Pilot Offering of *CS1315 Introduction to Media Computation*

May, 2003

Mark Guzdial
Rachel Fithian
Andrea Forte
Lauren Rich

Executive Summary

<Mark writes at the end>

Table of Contents

1. Introduction: What we collected and why

During the Spring 2003 semester, Georgia Tech offered four introductory computing courses.

- CS1321 *Introduction to Computing* is the traditional CS course at Georgia Tech. It is taught in sections of 200 students and more, using the programming language Scheme.
- CS1361, sections A and B, *Introduction to Computing for Engineers* is an introductory computing course aimed at engineers. It is taught in sections of 20-25, using the programming language MATLAB.
- CS1361, section C, is a course aimed at covering the same concepts as CS1321, but in the programming languages MATLAB and Java, in a 75 person lecture.
- CS1315 *Introduction to Media Computation* was trialed in the spring 2003 semester. It's a course aimed at non-CS and non-engineering majors. It was taught in a section of 120 using the programming language Python.

The main focus of our assessment effort during the Spring 2003 semester was to understand CS1315, but we felt that we couldn't really do that without comparing it to the other three CS courses. As a result, we gathered information not only about our new course for non-CS majors, but also about the potential of specialized computing courses.

We were interested in four main questions:

- What happened in the class? That is, what content was covered, how were students assessed, and what was the overall success rate? In particular, we were interested in the overall WFD (withdrawal, F, or D grades) rate. Our "best-practice" comparison is to the work by Laurie Williams and colleagues at North Carolina State University[1] where they found that specialized collaboration strategies led to improved retention. For non-CS majors, they were able to achieve a 66.4% success rate (WFD 33.6%) compared to a 55.9% success rate in a traditional CS1.
- Did the students learn computing? A great retention rate does not necessarily imply good learning.
- What did students think of the class? One of our goals was to improve students' attitudes toward computer science. We're interested not only in the attitudes toward the class overall, but also toward the role of the specific design features: how motivating were the relevance of the domain of media computation, support for collaboration, and the value of the technologies we implemented?

[1] Nagappan, N., Williams, L., Miriam, F., Weibe, E., Kai, Y., Miller, C., and Balik, S. 2003. "Improving the CS1 Experience with Pair Programming," *Twenty-fourth SIGCSE Technical Symposium on Computer Science Education*, 359-362. ACM: New York. See also Williams, L. and Kessler, R.R., 2000, "Experimenting with industry's 'pair-programming' model in the computer science classroom," *Journal on Software Engineering Education*, December.

- Did the class appeal to women? We designed the course explicitly to appeal to women by attempting to address concerns expressed in studies of the gender bias in computer science courses.
- How do CS1315, CS1321, and COE1361 compare?

To address these questions, we gathered the following data:

- Initial, midterm, and final surveys were given to consenting students in CS1315, one section of CS1321, and COE1361 section B. We were also able to administer midterm surveys in COE1361 section A.
- We examined the homework assignments of consenting students.
- There was a set of problems in common between CS1315, CS1321, and COE1361 on some of the exams and quizzes to give us some points of comparison. A classically difficult question, the Rainfall Problem, was used in order to compare results to those achieved by other CS1 courses.
- Interviews were conducted with female volunteers from CS1315 at both the middle and the end of the semester.

An important feature of the population we studied is that it did not include a significant number of computer science majors in *any* of the courses. Because most CS majors take CS1321 in the fall semester and we collected our data in spring, only five CS majors were enrolled in the course; of these, only a couple filled out surveys. This lack of data from CS majors allows us to better compare the three courses in terms of non-majors; however, it will be interesting to see what happens when CS students' voices are included in future surveys and interviews.

2. What happened in the class?

Overall, the class went very well. Students did well on the homework and exams, but more importantly, took advantage of the creative aspects of the course. For example, the third homework asked students to create a collage where the same image appeared at least three times, with *some* different visual manipulations each time—but that was a minimum. More images and more manipulations were encouraged. As can be seen by the examples in Figure 2 (which were all showcased by public posting on the CoWeb collaborative website), students did go above-and-beyond. Some of these programs were over 100 lines in length!

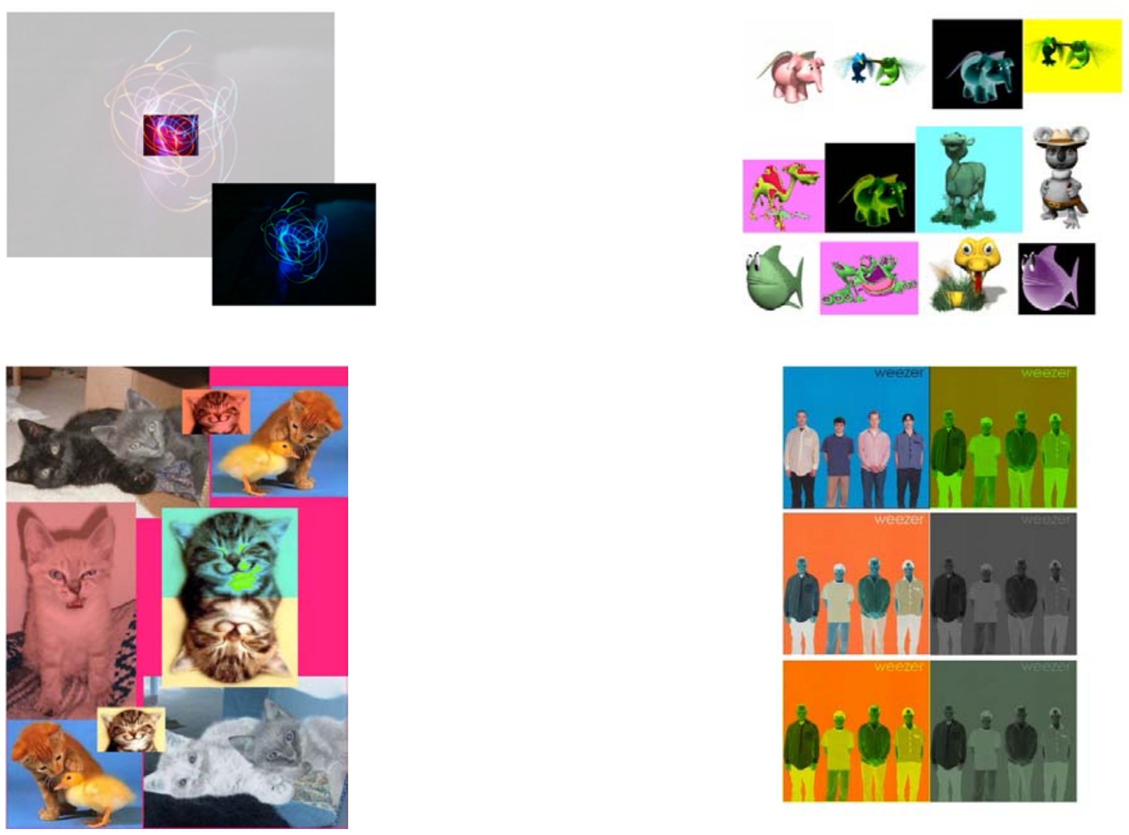


Figure 2: Student collages for Homework 3

2.1. Who took the class?

120 students enrolled in the course. From the class roll, we know that the major distribution was as shown in Figure 1:

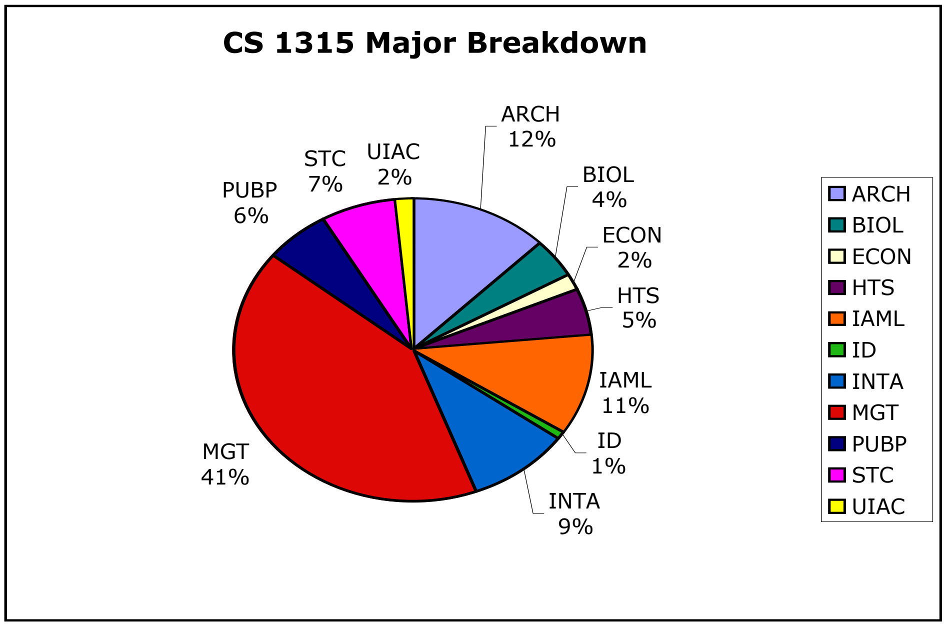


Figure 1: CS 1315 class breakdown by major

Of the 120 students in the course, 71% consented to participate in our study. From consent forms, we determined that the gender distribution was 33% male and 67% female, and the ethnicity distribution was 84% Caucasian, 8% Asian, 4% blank, 2% African-American, and 1% other.

2.2. What content was covered?

The syllabus for the course walks through each media type, with some repetition of concepts so that conditionals and loops can be re-visited in different contexts. A rough description of the syllabus is provided here.

- Week 1: Introduction to the course and the argument for the importance and usefulness of media computation. Introduction to variables and functions, in the context of playing sounds and showing pictures.
- Weeks 2–3: Pictures as a media type, including psychophysics (why don't we see 1024x768 dots on the screen?), looping to change colors with a simplified **for** loop, conditionals to replace specific colors, then indexing by index numbers to implement mirroring, rotating, cropping, and scaling.
- Weeks 4–6: Sound as a media type, including psychophysics (how human hearing limitations make MP3 compression possible), looping to manipulate volume, then indexing by index numbers to do splicing and reversing of sounds. Include discussion of how to debug and how to design a program, as those issues arise. One lecture on additive and FM sound synthesis.
- Week 7: Text as a media type: Searching for text, composing text, reading text from a file and writing it to a file. An example program parses out the temperature from a downloaded weather page.

- Week 8: Manipulating directories. Manipulating networks, including making the temperature-finding program work from the “live” Web page. Introduction to HTML.
- Week 9: Discuss media transitions. Moving from sound to text and back to sound again. Using Excel to manipulate media after converting it to text.
- Week 10: Introduction to databases: Storing media in databases, using databases in generating HTML.
- Week 11: Movies: How persistence of vision makes animations and movies possible, generating frames using the various techniques described earlier in the semester.
- Week 12: “Can’t we do this any faster? Why is Photoshop faster than Python? ” Introduction to how a computer works (e.g., machine language), and the difference between an interpreter and a compiler. Algorithmic complexity and the limits of computation.
- Week 13: “Can we do this any easier?” Decomposing functions, modularity, and functional programming (map, reduce, filter, and simple recursion).
- Week 14: “Can’t we do this any easier?” Introduction to objects and classes.
- Week 15: “What do other programming languages look like?” Brief overview of JavaScript and Squeak.

2.3. How were students assessed?

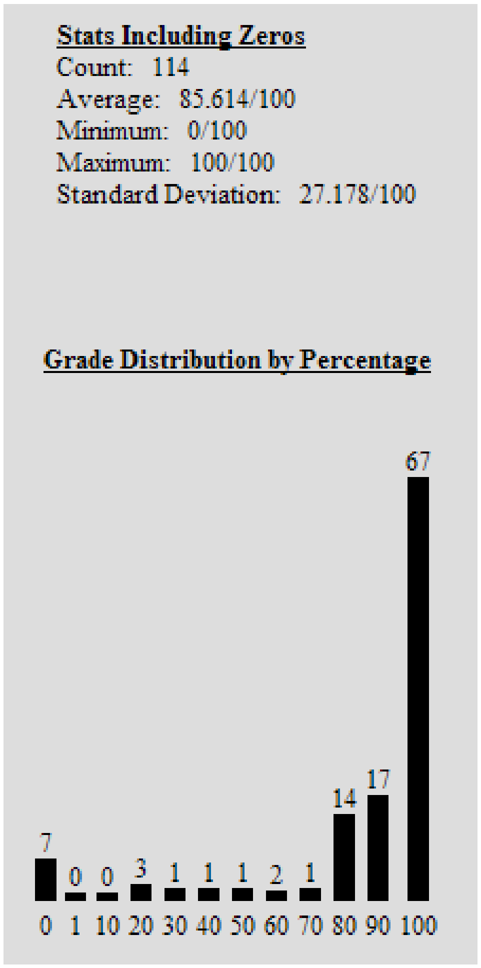
Student work consisted of both activities whose goal was student learning and activities whose goal was assessment of learning. The student learning activities were graded and counted toward the final grade, but were weighed less heavily than the assessment activities. Students were encouraged to collaborate on the student learning activities.

- There were five take-home lab-like activities to help students learn basic productivity software and skills such as email, Web browsing, Word, Excel, PowerPoint, and HTML. These were worth 15% of the total grade.
- There were three quizzes, each of which was preceded by a pre-quiz. The pre-quizzes were distributed on Mondays and were open for collaboration. The quizzes were given in-class for 20 minutes on Wednesdays and looked very similar to the preceding pre-quizzes. Each pre-quiz was worth 20 points and each quiz was worth 80 points. Overall, the pre-quiz/quiz grade was worth 15% of the total grade.
- There were six homework assignments, all involving programming, and all of which could be worked on collaboratively. These assignments appear in Appendix A, and the grades for these are summarized in Table 1. They were worth 15% of the total grade.
- There were two take-home exams, which were programming assignments like the homework, but on which students were asked not to collaborate at all. They were asked to enter a statement that they neither gave nor received aid

on the exam. These appear in Appendix B, and the grades for these are summarized in Table 2. These were worth 20% of the total grade.

- There were two in-class exams and one final exam, which made up 35% of the total grade. Students were not allowed to collaborate on these exams. These appear in Appendix C, D, and E, and the average grades and distributions for these appear in Tables 3, 4, and 5[2].

While the numbers are impressive, the distributions are even more striking. Figure 3 is the distribution for Homework #3 (the collage assignment) described earlier. The student performance on this homework was overwhelmingly positive.



Stats Including Zeros
Count: 114
Average: 85.614/100
Minimum: 0/100
Maximum: 100/100
Standard Deviation: 27.178/100

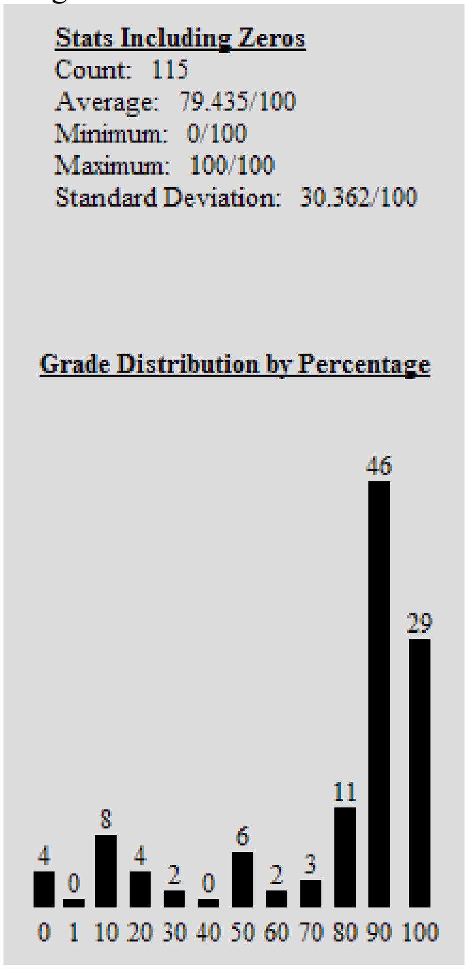
Grade Distribution by Percentage

Figure 3: Grade statistics and distribution for Homework #3

Obviously, from the collaboration policy described, students might have achieved much of this performance through collaboration with one another. Literally, every one of the 63 students who got 100% could have handed in the same assignment. Though this explanation is possible, the creativity exhibited in the collages suggest students worked individually and take some pride in their individual accomplishment.

[2] Grades on the final exam were somewhat lower than expected because several of the seniors handed in their finals after less than an hour in the final exam with the explanation (sometimes verbal, sometimes written on the exam), “I only needed X points for my A/B/C, so I think that that’s all I need.”

We also have the take-home exam performance to support the conjecture that student performance wasn't merely a matter of collaboration. Figure 4 provides the grade statistics and distribution from Take-Home Exam #2. While the grades aren't quite as high as on Homework #3, over half the class got an A (90% or better) on this assignment.



Stats Including Zeros
Count: 115
Average: 79.435/100
Minimum: 0/100
Maximum: 100/100
Standard Deviation: 30.362/100

Grade Distribution by Percentage

Figure 4: Grade statistics and distribution for Take-Home Exam #2

Homework	Average	Standard Deviation
1: Manipulating Color	92.0	22.1
2: Splicing Sound	89.1	24.5
3: Image Collage	86.3	26.0
4: HTML Generation	91.3	22.6
5: Animation	89.1	27.6
6: Ticker Tape	84.6	29.2

Table 1: Homework Grades for CS1315

Take Home Exam	Average	Standard Deviation
1	92.1	15.3
2	81.2	27.8

Table 2: Take Home Exam Grades for CS1315

Final Exam Breakdown			
	Average	**Points Possible**	**Standard Deviation**
Problem 1 Search function	6.2	10	2.7
Problem 2 Random comment generator	10.3	15	4.0
Problem 3 Vocabulary	32.0	36	8.3
Problem 4 Concept questions	7.4	12	3.0
Problem 5 Short essay	9.6	12	2.5
Problem 6 Algorithm explanation	8.8	15	4.0
Total	68.8	100	25.9

Table 3: Final Exam Grades for CS1315

Midterm #2 Breakdown			
	Average	**Points Possible**	**Standard Deviation**
Problem 1 Code explanation	12.6	20	4.8
Problem 2 Rainfall problem	11.4	25	8.7
Problem 3 Short essay	24.0	28	4.3
Problem 4 Search function	18.1	27	6.2
Total	62.7	100	22.0

Table 4: Midterm #2 Grades for CS1315

Midterm #1 Breakdown			
	Average	**Points Possible**	**Standard Deviation**
Problem 1 If statements	12.6	14	1.8
Problem 2 Code tracing	17.1	20	5.1
Problem 3 Code matching	18.7	21	3.2
Problem 4 Concept questions	15.9	20	3.6
Problem 5 Sound function	23.0	25	3.7
Total	85.8	100	15.6

Table 5: Midterm #1 Grades for CS1315

2.4. What support for collaborative learning was used in the class?

The students made extensive use of a collaborative website (CoWeb) (at http://coweb.cc.gatech.edu/cs1315).

- Each of the assignments (homework, labs, and take-home exams) had Q&A pages associated with them where extensive discussions took place.
- For each exam, a review page was posted where students could answer sample questions, ask questions about the questions or concepts, and critique each other's remarks.
- Each week, a new general "Comment on the week" page was put up for general feedback and discussion.
- A number of special topics pages were provided to encourage discussion (favorite movies, music, Atlanta-area restaurants).
- Pages encouraging anonymous feedback were provided.
- A "Soapbox" section at the top of the CoWeb pages encouraged students to make public comments, which often led to lively discussion.

2.5. What was the WFD rate?

By drop day, only two students dropped the course. At the end of the course, 3 students were given D's, 7 were given F's, and three were given I's (two for pending academic misconduct cases and one for hardship supported by the Dean of Students' office). W's, F's, and D's made up 12 of the 120 originally enrolled students, for a 10% WFD rate.

2.6. Future work on the curriculum and course materials

The course notes that were developed for the course only covered about half of the semester. The rest of the course notes are being developed during Summer 2003. We are also improving the labs during the summer—they were only modified slightly from CS1321's labs, and we'd now like to make more substantial revisions to fit them into the media computation context.

We also plan to revise the technology considerably during the Summer 2003. The technology developed for the course (a programming environment called JES and a set of media manipulation tools called MediaTools) performed admirably—e,g, there was literally not a single report of the environment crashing during the entire semester. Nonetheless, several bugs were noted and two significant enhancements are desired:

- First, there is no support for debugging in the programming environment. We need to figure out what support is useful and understandable to non-CS majors who are creating loops that iterate tens of thousands of times (e.g., processing samples or pixels). Simple breakpoints won't work—no one wants to click **Continue** 80,000 times.
- Second, two separate environments exist for viewing media and for writing programs; this separation between the media tools and the programming environment was a problem. Students who have just created a sound want to visualize it *then*, not after saving it to WAV file, opening MediaTools, and then opening the WAV file.

There is considerable interest in the course outside of Georgia Tech. A contract for a book based on these course notes has been signed with Prentice-Hall. Kennesaw State University is teaching a summer camp for high school students based on the course during Summer 2003, while Gainesville College is offering the course to a small number of students during Summer 2003. With Marion Usselman of CEISMC, we are exploring the creation of a high school version of the course.

3. Did the students learn computing?

3.1. Did the students think that they learned computing?

We asked the question of whether students felt that they were learning to program on both the midterm and final surveys. In general, students reported that they *did* feel that they were learning to program.

At the midterm survey, the majority of students in all four CS1 classes reported that they felt that they were learning to program (Table 6). Surprisingly, between the midterm and final surveys, CS1315 students' confidence in their learning dropped (Table 7). This may be because the material at the beginning of the semester was less challenging than that presented at the end, or because as students learn more, they begin to realize how little they know.

Course		Yes	No	Total
CS 1321	Raw count	29	4	33
	Percentage of class	**88%**	**12%**	100%
CS 1315	Raw count	84	3	87
	Percentage of class	**97%**	**3%**	100%
COE 1361b	Raw count	26	5	31
	Percentage of class	**84%**	**16%**	100%
COE 1361a	Raw count	20	1	21
	Percentage of class	**95%**	**5%**	100%

Table 6: Midterm Survey Responses: Are you learning to program?

Course		Yes	No	Blank	Total
CS 1321	Raw Count	23	3	0	26
	Percentage of class	**88.5%**	**11.5%**	0.0%	100.0%
CS 1315	Raw Count	42	7	5	54
	Percentage of class	**77.8%**	**13.0%**	9.3%	100.0%
COE 1361b	Raw Count	19	2	0	21
	Percentage of class	**90.5%**	**9.5%**	0.0%	100.0%

Table 7: Final Survey Responses: Did you learn to program?

Students were asked to rate their programming expertise before entering the course, and again at the end of the course. Figures 5 and 6 represent the distribution of self-reported expertise before and after in each of the three courses. Figure 4 represents the average difference between students' before and after ratings and indicates that the amount of self-reported learning was comparable across all three courses.

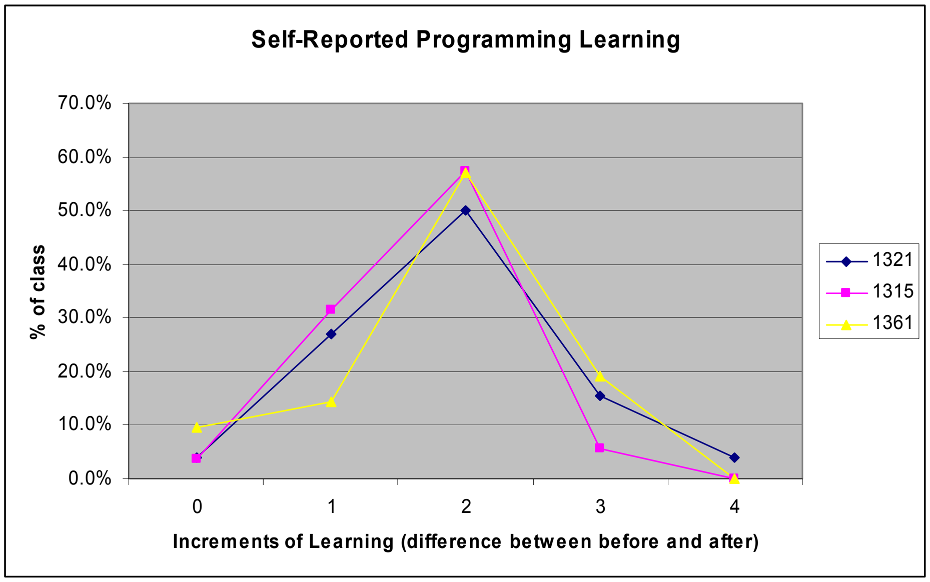


Figure 4: Self-Reported Programming Learning

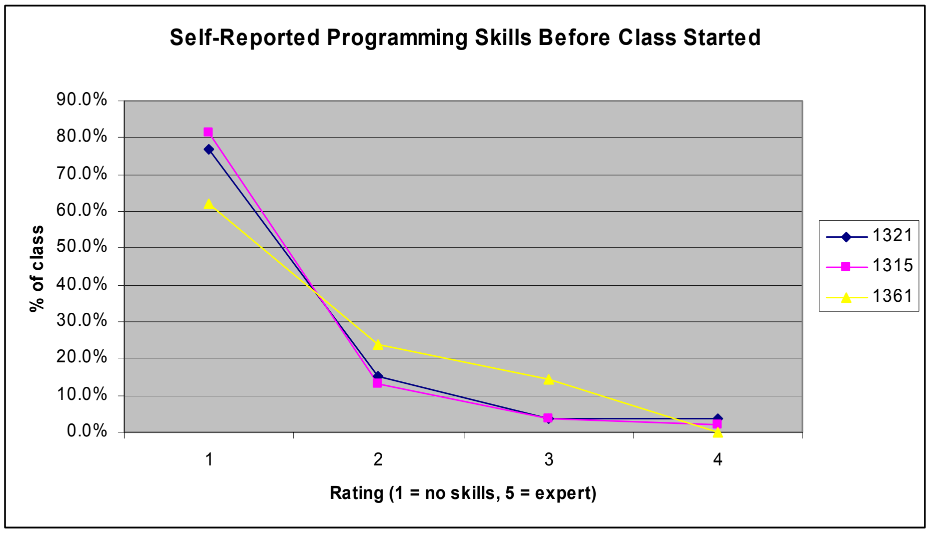


Figure 5: Self-Reported Programming Skills Before Class Started

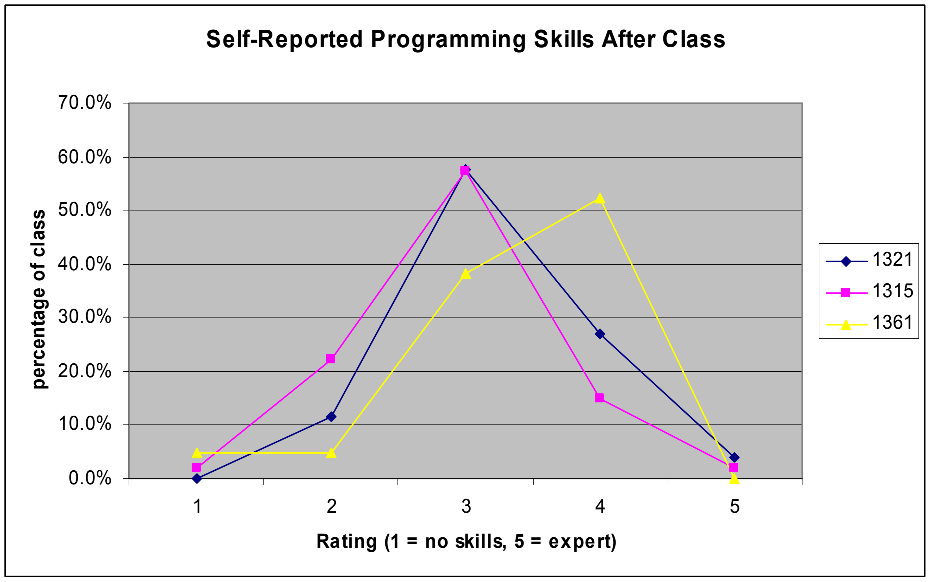


Figure 6: Self-Reported Programming Skills After Class

3.2. *How do CS1315 students perform on known problems in computing? How do they compare to CS1321 and COE1361 students?*

In order to compare CS1315 students' achievement to that of CS1321 and COE 1361, we developed several problems that we attempted to put on all three courses' exams and quizzes. Unfortunately, logistical considerations such as different rates of development and sequencing of concepts in each course and different times and numbers of exams and quizzes prevented us from distributing the problems as uniformly as we would have liked. In addition, those questions that did make it onto all three exams were modified to such an extent that it is difficult to compare them. In general, we found that the different programming languages used, the differences in directions given on exams, and the inclusion of model code on CS1315 exams created unique conditions for each course and rendered the results fundamentally incomparable.

A more reliable indicator of CS 1315 students' programming achievement can be found in students' attempt to solve a classically difficult problem: *The Rainfall Problem.* The rainfall problem is:

> "Write a program that repeatedly reads in positive integers, until it reads the integer 99999. After seeing 99999, it should print out the average. "

At Yale in the mid-80's, Elliot Soloway and crew gave this problem to several groups of students[3]:

- First-semester CS1 students after learning and using WHILE, REPEAT, and FOR, 3/4 of the way through the term.
- CS2 students 3/4 of the way through.
- Juniors and seniors in a systems programming course.

There were two groups in each set: One group got raw Pascal. The second group got Pascal with a **leave** (**break**) statement.

% Correct:	**Pascal**	**Pascal-with-Leave**
CS1	14%	24%
CS2	36%	61%
Advanced	69%	96%

Table 8: Rainfall Problem results at Yale in the mid-80's

Our students:

- Only saw WHILE once in class, but knew FOR really well.
- Only saw break once the day before the midterm—it wasn't required for the midterm.

Here's how we phrased the problem on the CS1315 midterm:

> Write a function **rainfall** that will input a list of numbers, some positive and some negative, e.g., [12, 0, 41, -3, 5, -1, 999, 17]. These are amounts of rainfall.

[3] Soloway, E., Bonar, J., and Ehrlich, K. 1983. "Cognitive strategies and looping constructs: An empirical study." *Communications of the ACM*, 26(11). 853-860.

Negative numbers are clearly a mistake. Print the average of the positive numbers in the list. (Hint: The average is the total of the positive numbers divided by the number of just the positive numbers.)
You may want to recall these examples from lecture on how to manipulate lists...

Extra credit (5 points): Write the function so that, if the number 999 appears in the list, *add in no later numbers in the list*. So, if the above example were input, the 17 would not be added into the average.

14 people out of 113 (12%) who took the test who "got it" –by Elliot Soloway's standards that means they had a correct solution (aside from syntactic errors). This doesn't count several people who got the extra credit but didn't get the main problem! Like Elliot's paper showed, students understand "break" really well. This number (14 out of 113) does count some students who nailed the main program but didn't attempt the extra credit; it is questionable whether this criteria fully lives up to Ellliot's standards. With partial credit, the average on the problem was 45%, low enough that we dropped the problem for those whose grade was improved by dropping it. We didn't want the experiment to hurt anyone's grades, and this problem was clearly for experimentation only.

CS1321 also included a variation of the rainfall problem on *two* of their tests. On the first test (which stipulated the use of tail recursion), the result was an average score of 53%. On the second test (which was open to any kind of iteration), the result was an average score of 60%.

Overall, the rainfall problem continues to be a hard problem for CS1 students. These results don't show that CS1315 students are learning programming remarkably well. But they do show that CS1315 students aren't out of the ballpark either. It was a bit unfair from an assessment perspective to include this problem on the midterm: the students had not done *any* list processing yet, and the rainfall problem was far from a communications context. Nevertheless, it does suggest that achievement in CS1315 isn't dissimilar from that of other CS1 courses described in the CS education literature.

3.3. Future work on the learning of computation

During Summer 2003, we plan to directly study the homeworks done by students who gave consent in order to categorize the difficulties and bugs that students faced and the strategies that they used.

4. What did students think of the class?

4.1. How hard was it?

Students did not find the class to be a cakewalk. On the midterm survey, students were asked to rate their expectations of course difficulty and to rate how well those expectations were being met. On a scale of 1 to 5 where 1 = easy, 5 = hard, the average expected level of difficulty was 3.5. No students reported expectations at the easiest level (1), while nearly 20% reported expectations of the hardest level (5) (see Figure 7).

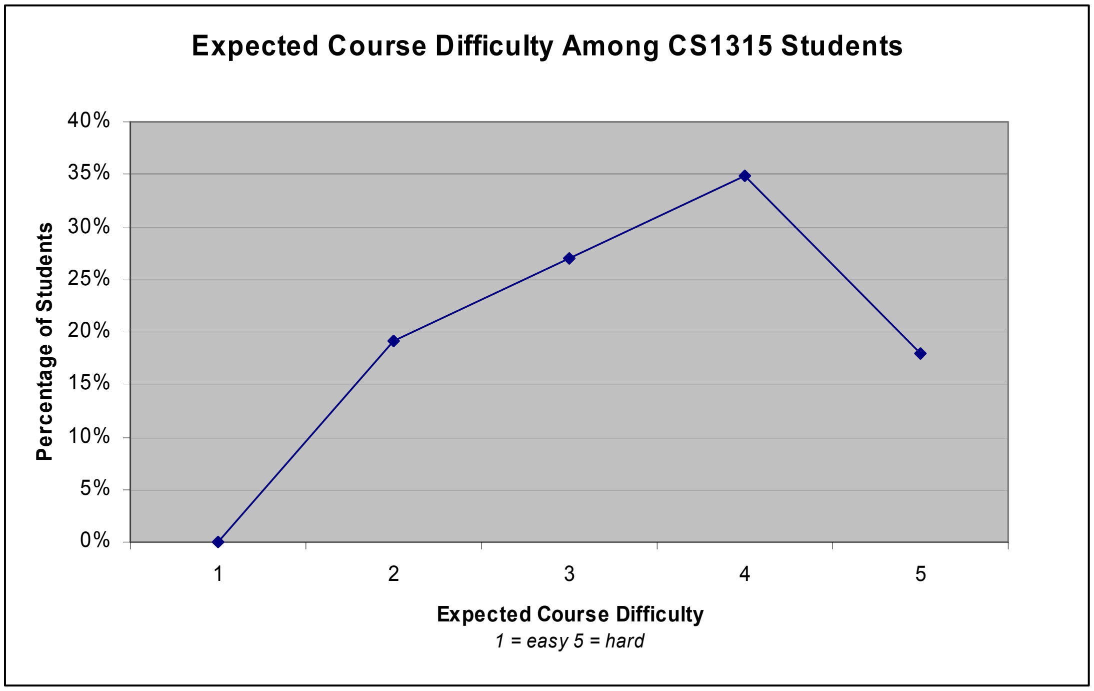


Figure 7: Expected course difficulty among CS1315 students

Figure 8 describes the same arc, with the average match to expectations overlaid. The number **three** on the right Y axis represents perfectly matched expectations. We see here that students who believed CS1315 would be somewhat easy (difficulty level 2) found that their expectations were not met: the class was harder than they expected. Students who expected the class to be somewhat hard (4) or hard (5), discovered that the course was not quite as difficult as they expected, with their average match to expectations hovering just below three.

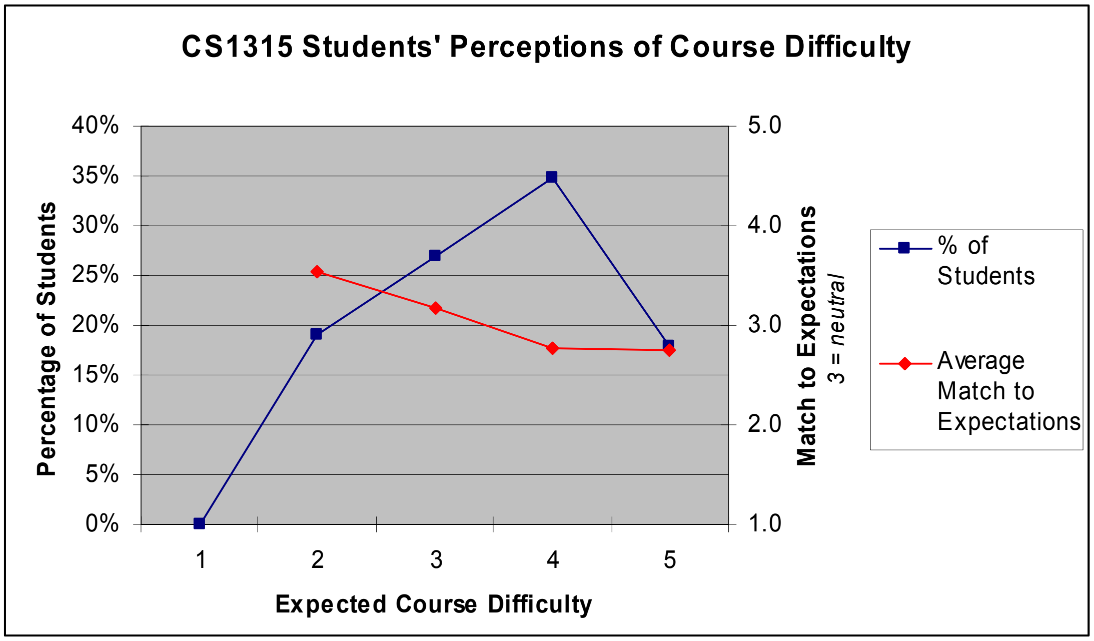


Figure 8: CS 1315 Difficulty expected versus difficulty perceived

4.2. Was the course relevant, interesting, and motivating?

When asked what they like about the class, the students affirm that we're succeeding at creating a course that students recognize for its applicability, particularly for non-CS majors: (All of the quotes below are from female students.)

- "I like the feeling when I finally get something to work."
- "Very applicable to everyday life."
- "I dreaded CS, but ALL of the topics thus far have been applicable to my future career (& personal) plans- there isn't anything I don't like about this class!!!"
- "When I finally get a program to work like I want it to."
- "The professor answers questions in class and online and is concerned about our success in the class. He also seems to understand that most of us are not engineers and most likely won't do straight programming in the future- just the way of thinking is important."
- "Collaboration! If I can't figure it out, I can ask for help."
- "Collaboration. It helps to have more than one person to explain concepts; helps them sink in more."

When we ask students "What is something interesting, surprising, or useful that you learned?" we found that students appreciated the relevance of the course and even found the computer science interesting (again, all female respondents):

- "The most useful things I have learned are the basics about computers and pictures/sound. I think when we learn HTML- that will be interesting and useful to real life applications."
- "Just general concepts about programming. It's pretty logical, sort of like in math, so it's understandable."
- "Programming is fun and ANYONE can do it!"

4.3. What role did collaboration play?

According to many students, the collaboration policy and the CoWeb each played significant roles in their success in CS1315. In fact, on the final survey, when asked what absolutely must *not* change about the course, nearly 20% of the respondents named the CoWeb, while over 20% referred to collaboration in general

On the midterm survey, over 90% of CS1315 students reported that they had collaborated on assignments. On the same survey, students were specifically asked to comment on collaboration. Students reported that collaboration plays an important role in learning (37% of CS1315 respondents), that it serves as a resource for getting questions answered (7%), and that it reduces anxiety about the course (5%).

The following quotes illustrate the role that collaboration and the CoWeb played in some students' learning experiences:

Q. What about collaboration? Did you collaborate? Do you have any thoughts on collaboration?
Student 4: "Actually, I think that is one of the best things about this class. My roommate and I abided by all the rules, I mean, I know there is a big deal about collaboration in computer science right now and we didn¹t collaborate on anything we weren¹t supposed to even though we live like 5 feet away from each other. I mean, we didn¹t ever copy each other's code or anything, but we took full advantage of the collaboration. It was more just the ideas bouncing off each other. I don¹t think this class would have been as much fun if I wasn't able to collaborate."

Q: Do you think the coweb is beneficial? Why?
Student 1: "Very beneficial, it keeps you on track, keeps you organized. You get to see what people are having problems with and maybe see... I always start off looking at what people have had problems with."

Q. How do you plan to study for the final exam?
Student 2: "Hard. I have actually read the book. I read it before the first test. I'll go through all the lecture slides and look at old tests. And the online review [on the CoWeb] is awesome and I'm very grateful he did that. I like that you can make comments, I can see stuff where other people had questions cuz they ask stuff that I might not have thought of. And I'll study with a group."

4.4. What role did the technology we built play?

Two applications were built to support student programming and comprehension of media encoding. The Jython Environment for Students (JES) is a simple editor and interpreter that allows students to write and run Jython code (Figure 9). MediaTools is a suite of applications that allow students to explore media through a variety of representations (Figure 10).

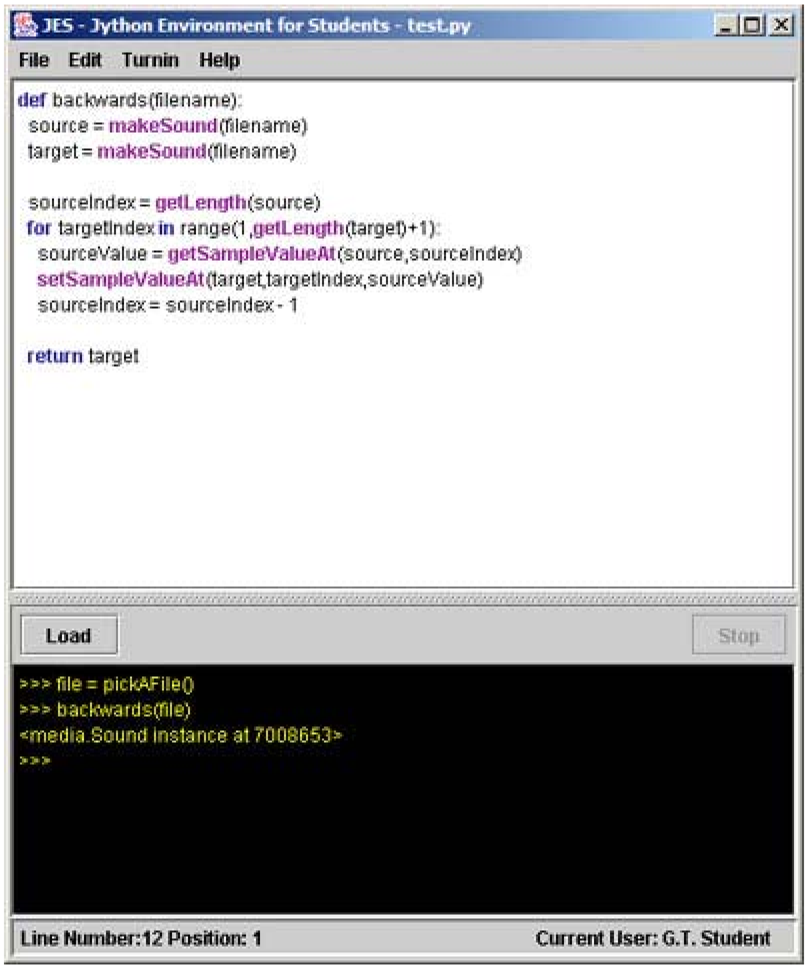
Students seemed quite comfortable using JES and reported few problems with it. Students remarked that they liked its simplicity. They did report some difficulties, including difficulties installing the software and some usability issues. They also reported several frustrations about the Jython language (including the annoyance of arranging spacing and indentation in their programs). We are currently updating JES and working to address these issues this summer.

During observations, students were able to use JES effectively when composing programs, but had difficulty debugging. This was partly due to lack of understanding of the debugging process (e.g., students who write an entire program without trying to compile or run it), and partly due to problems with JES. Students frequently remarked that error messages provided by JES were not very informative, and sometimes misleading, for trying to resolve the error. Also, some debugging issues are inherent in the domain of media manipulation – print statements or stepping through the program are impractical when dealing with sounds or pictures with thousands of elements to manipulate. The JES development team is currently working to improve error messages and develop a debugger appropriate for this domain.

Little use of MediaTools was observed as students completed their homework assignments. Students seem more likely to use tools they are more familiar with, such as Internet Explorer, for viewing pictures. It is unclear how much students use MediaTools outside of observations and what the obstacles to use are. This summer, developers are integrating MediaTools functionality into JES. We expect integration to encourage more use of MediaTools since students will only have to load one program instead of two and might be more likely to use MediaTools if it is omnipresent and easily accessible from JES.

Students felt strongly that JES was useful and something to play with, as suggested by this quote from the interviews:

> **Q: What do you think about the homework galleries on the coweb?**
> "I don't ever look at it until after I'm done, I have a thing about not wanting to copy someone else's ideas. I just wish I had more time to play around with that and make neat effects. But JES will be on my computer forever, so that's the nice thing about this class is that you could go as deep into the homework as you wanted. So, I'd turn it in and then me and my roommate would do more after to see what we could do with it."


JES - Jython Environment for Students - test.py
File Edit Turnin Help
Load
Stop
Line Number:12 Position: 1
Current User: G.T. Student

```
def backwards(filename):
 source = makeSound(filename)
 target = makeSound(filename)

 sourceIndex = getLength(source)
 for targetIndex in range(1,getLength(target)+1):
  sourceValue = getSampleValueAt(source,sourceIndex)
  setSampleValueAt(target,targetIndex,sourceValue)
  sourceIndex = sourceIndex - 1

 return target
```

```
>>> file = pickAFile()
>>> backwards(file)
<media.Sound instance at 7008653>
>>>
```

Figure 9: Jython Environment for Students (JES)

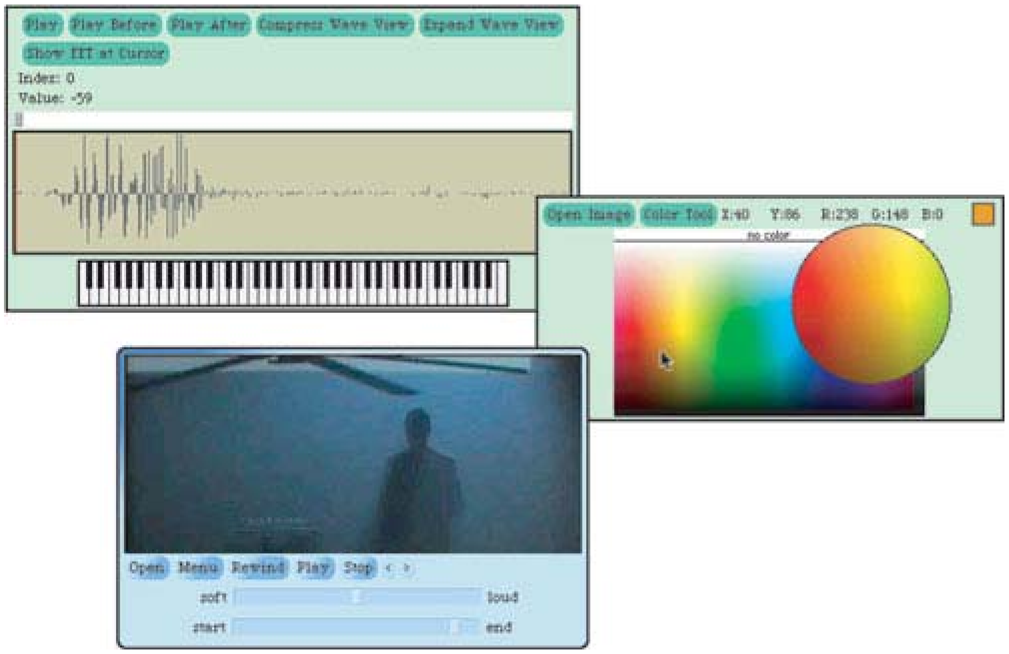


Figure 10: Shots of the MediaTools Application Suite

4.5. *How did these attitudes differ by College?*

While the number of survey responses collected from some Colleges and ethnicities was extremely low, we have made some preliminary observations based on the available data. In general, architecture students seem to have the most negative attitudes toward introductory computer science. Students of the Ivan Allen College of Liberal Arts who took CS1315 had particularly positive attitudes toward computer science by the end of the semester. Over 40% reported that they believe they will use programming again in the future, which is higher than the class average of 30%. Nearly 20% reported that they plan to take more computer science, compared to a class average of less than 10%. These numbers indicate that liberal arts students are especially likely to have positive attitudes toward computer science after taking CS1315. We attribute this trend to the fact that the media computation course was designed to appeal to students of design and communication.

	Number of respondents	**Will use programming again in the future**	Self-Reported Programming Expertise *1 = no skills 5 = expert*		**Feel they learned to program**	**Plan to take more CS.**	**Would recommend 1315 to friends who didn't have to take it**
			Before CS 1315	**After CS 1315**			
Architecture	8	12.5%	1.1	2.5	50.0%	0.0%	37.5%
Ivan Allen	24	41.7%	1.3	3.0	91.7%	17.4%	82.6%
Management	19	21.1%	1.3	2.9	68.4%	5.3%	68.4%
Science	3	33.0%	1.0	2.5	100%	0.0%	50.0%

Table 9: CS1315 students' final survey answers by College

5. Did the class appeal to women?

The scarcity of women who choose to pursue careers in computing has generated interest in better understanding the field and its perception among females. Literature on women and computing suggests that reasons for the marked disinterest of women include the emphasis on technical detail rather than application, the perception of computing as an uncreative field, and a frequently uncongenial culture.[4] One of our hopes is that media computation will provide a CS1 experience that engages women and overcomes these particular reasons for female disinterest in computing.

One student summed up her CS1315 experience as follows:

Q. What is the most surprising or interesting thing that you learned?

[4] Margolis, Jane and Fisher, Allan. 2001. *Unlocking the Clubhouse: Women in Computing*. Boston: MIT Press.
AAUW. 2000. *Tech Savvy: Educating Girls in the New Computer Age.* New York: American Association of University Women Education Foundation.

"… that programming is not scary, it is actually pretty cool and when you make a program that actually runs it's a really good feeling. I didn't expect to enjoy it at all because all I'd heard were just the bad stories."

In the midterm survey, we asked students what they liked best about the course they were taking. The following table includes some of the most common answers from female students and the percentage of responses that fell into each category:

What do you like best about this class so far?					
Course	**Don't Like It/Nothing**	**Programming**	**Enjoy Content**	**Content is useful**	**Feeling More Knowledge-able About Computers**
1321	23%	0%	8%	0%	8%
1315	0%	9%	28%	12%	4%
1361b*	0%	67%	33%	0%	0%
1361a*	0%	40%	20%	40%	0%
* number of female respondents extremely low.					

Table 10: Female Responses on Midterm Survey: What do you like best about this class so far?

5.1. Did female students find it relevant?

In section 4.2, we included quotes from female CS1315 students who described the relevance of the course and the real life applicability of what they learned. This is particularly important because research suggests that women who do choose to pursue computer-related careers often cite application in other fields as a reason for their interest in computing. About 12% of the women in CS1315 named the usefulness of the course content as the best feature of the class (Table 10).

5.2. Did female students find it creative?

Creativity was identified on surveys by a number of female students as a compelling aspect of CS1315. This is important because a perceived *lack* of creativity has been cited as one of the reasons that female students are not generally attracted to CS.

5.3. Were female students motivated to want to continue in CS?

Perhaps the best indicator of attitudes are behaviors. Students who have positive attitudes about CS and who enjoy computing are likely to continue in computer science. While it is too early to tell what CS courses CS1315 students will take in the future, we asked on the final survey whether or not they would be interested in taking a more advanced media computation course. 60% of the female respondents answered that they would take Media Computation II, which is very close to the overall course average of 63%.

Surprisingly, when asked *on the same survey* whether or not they plan to take more CS courses, only 6% of those same female respondents responded affirmatively. Why would 60% be willing to take media computation, while only 6% planned to take more CS courses? The obvious answer is that there is currently no advanced media computation course, and, given the current selection of CS courses, CS1315 students don't see a compelling reason to take more.

A striking example of change in attitudes toward CS can be found in the following statement made by a female CS1315 student when asked whether or not the course had changed her perception of computer science:

> "YES, I'm not intimidated by it anymore. My mom was SO surprised when I told her that I want to be a TA she almost fell on the floor, cuz she's heard me complain for 3 years about taking this class and now I want to go do it to myself again!"

5.4. Did it matter that the course was non-CS and non-Engineering?

Making sure that students feel comfortable in class and asking questions is not just conventional wisdom: a study on factors contributing to success in CS1 found that the best predictor of success in introductory computer science courses is students' comfort level.[5] When asked about her use of the CoWeb, one female student made it clear that she felt not having CS majors in CS1315 made it easier for the non-CS majors to ask questions and get help:

> **Q. Have you ever posted to the CoWeb?**
> "I think I've posted to everything. Sometimes I'll just make random comments. Sometimes I ask a specific question and he [the professor] asks for clarification. I would feel different in a class with a bunch of CS majors. But since we are there with a bunch of management—other students—it's kind of more comfortable."

Another student suggested that the anonymous nature of the CoWeb made it easier to ask questions, particularly early in the semester when concepts are still new and students may still feel that their questions are "dumb":

> **Q. Have you consistently felt comfortable asking questions?**
> Not at the beginning. One on one, yes. In lecture, not at the beginning because I felt that I was so far behind other people and the ones who were putting things on the web were the ones who really know stuff. But now I have no problem.

[5] Wilson, Brenda Cantwell, and Shrock, Sharon. 2002. Contributing to success in an introductory computer science course: A study of twelve factors. Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education. (pp. 184-188). New York: ACM.

Q. Do you think the CoWeb is beneficial?
Yes. And there's no reason to feel uncomfortable because if you feel dumb, just don't put your name at the end! I did that a few times.

This kind of freedom to ask questions benefits all the students, not just females; however, the work of Margolis and Fisher indicates that female students in particular tend to suffer because of both real and perceived inhospitable features of CS culture. Even during the first round of interviews female CS 1315 students reported feelings of competence and empowerment:

Q. Has the class provided additional insight into CS?
"[It has] changed my opinion for the better, not so intimidated anymore. I could figure out something if I had to. Like if my parents came to me I could help them. I'm still kind of intimidated by the CS stereotype, I guess, but I could figure it out if I had to."

5.5. Future work on studying female attitudes toward CS

During Summer 2003, Heather Perry will be using the CS1315 interview guide with women in CS1321 so that we can compare female attitudes between the two classes.

6. How do CS1315, CS1321, and COE1361 compare?

6.1. In students' impressions of computer science

The initial survey, which was distributed during the first week of the semester in each of the three classes, included two open-ended questions about students' perceptions of CS and their expectations for CS1: "What is CS to you?" and "What do you hope to get out of this course?"

Recurrent responses were used to develop codes for analyzing the short answers. For the question: "What is CS to you?" two coders independently tested the codes on a sample of answers and, after some minor changes to the code definitions, each coder applied the codes to the entire set of answers with about 75% intercoder reliability. At a second pass, reliability approached 90%. For the remainder of the short answer survey questions, a sample of answers were tested to reach a high level of intercoder reliability and a single coder applied the final codes to the entire set of answers.

Some common answers to the question "What is CS to you?" are summarized in Table 11. The codes were developed to reflect a range of increasingly sophisticated definitions of CS, from a self-reported lack of knowledge to defining CS as design or communication. The most sophisticated codes did not appear frequently enough in students' answers to warrant inclusion in the tables.

Initial Survey Responses: What is CS to you?							
Course	**Don't Know**	**How Compu-ters Work**	**Program-ming**	**Using Compu-ters for a Purpose**	**Scary/ Difficult**	**A Required Class**	**Web/ Interne t**
CS1321	4.9%	6.1%	51.2%	19.5%	8.5%	7.3%	0.0%
CS1315	4.7%	7.0%	48.8%	16.3%	10.5%	15.1%	8.1%
COE 1361b	11.4%	18.2%	45.5%	31.8%	2.3%	4.5%	0.0%

Final Survey Responses: What is CS to you?							
Course	**Don't Know**	**How Compu-ters Work**	**Program-ming**	**Using Compu-ters for a Purpose**	**Scary/ Difficult**	**A Required Class**	**Web/ Interne t**
CS1321	3.8%	3.8%	42.3%	26.9%	3.8%	11.5%	0.0%
CS1315	0.0%	7.4%	50.0%	20.4%	1.9%	20.4%	0.0%
COE 1361b	4.8%	14.3%	66.7%	38.1%	0.0%	4.8%	0.0%

Table 11: Definitions of CS at the beginning and the end of the semester.

While programming figured prominently in each classes' definitions of CS, some of the answers seem to indicate that the three courses had attracted different audiences for computer science. In particular, in the initial survey, a number of CS1315 students defined CS in terms of the Internet, while no students in either CS1321 or COE1361b mentioned the Internet at all. We believe this indicates an interest in the communication-related aspects of computing among students who are attracted to media computation. Not surprisingly, after a semester of computer science, none of the students defined CS in terms of the Internet on the final survey; still, we believe that interest in the Internet is likely to motivate many non-CS majors who take CS1.

Once the final surveys were collected, it became clear that many students had begun defining CS in terms of logic or problem solving, which had not been the case on the initial surveys. In addition, more students made positive or negative value judgments about CS. To investigate these trends, the initial and final surveys were recoded for references to problem solving and value judgments. The results can be seen in Table 12.

Initial Survey Responses: What is CS to you?				
	problem solving	**logic**	**positive**	**negative**
CS1321	7.32%	3.66%	6.10%	4.88%
CS1315	1.16%	0.00%	1.16%	4.65%
COE1361b	6.82%	6.82%	0.00%	2.27%

Final Survey Responses: What is CS to you?				
	problem solving	**logic**	**positive**	**negative**
CS1321	34.62%	7.69%	8.00%	11.54%
CS1315	5.56%	7.41%	11.11%	9.26%
COE1361b	14.29%	4.76%	0.00%	9.52%

Table 12: More definitions of CS at the beginning and the end of the semester.

Positive and negative definitions polarized slightly, with students seeming to become either more negative or more positive about CS by the end of each course, but without drastic differences between the three courses. The biggest increase in positive attitudes was seen in CS1315, while negative attitudes increased slightly across the board. Students were not asked to give an opinion of CS; these numbers represent students who decided to include a value judgment in their general definitions of CS.

Problem solving-related definitions of CS jumped considerably in CS1321, with less spectacular increases in CS1315 and COE 1361. All three courses showed a general increase in students defining CS as logic or problem solving skills.

6.2. *In how hard students perceive the courses*

Students' perception of course difficulty was determined via two questions on the midterm survey that asked the students to rate how hard they expected the class to be and how well the class matched these expectations. Overall, COE1361 students expected the least difficulty and understandably reported that the courses turned out to be more difficult than expected. CS1315 and CS1321 students expected moderately difficult courses, and generally reported that the courses met their expectations.

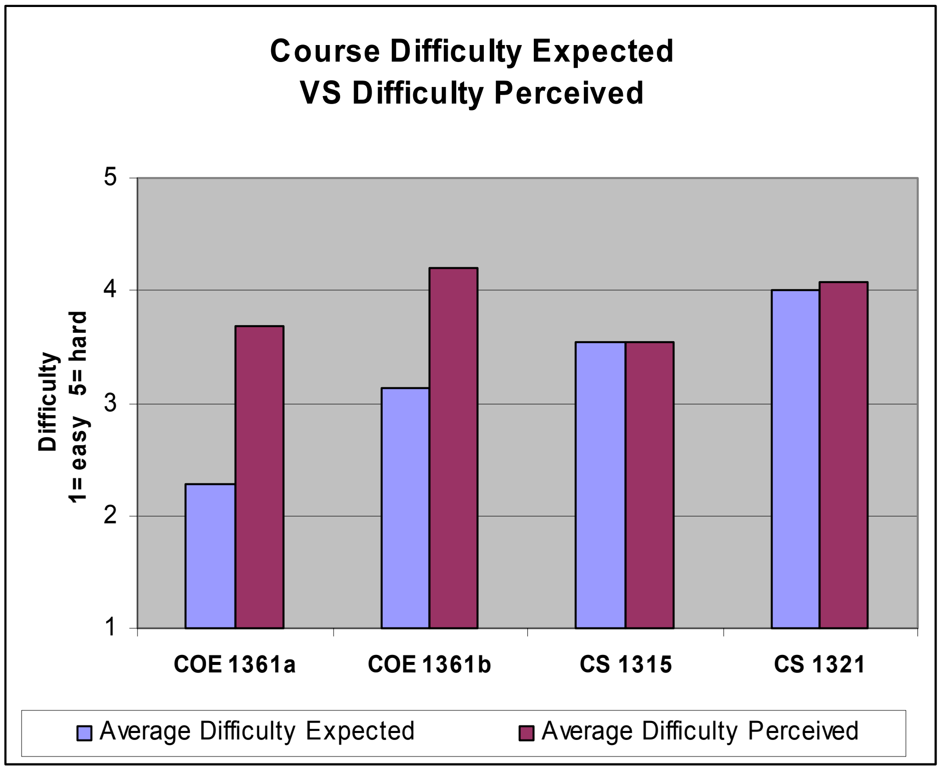


Figure 11: Students' Perceptions of Course Difficulty

6.3. *In how motivated students are to continue in CS*

Students were asked at the beginning of the semester and at the end of the semester whether or not they intend to take CS courses in the future. Both CS1321 and CS1315 saw increases in the number of students who reported plans to take more computer science in the future.

Do you plan to take any other CS classes after this class?		
Course	**Initial Survey**	**Final Survey**
CS1321	34.6%	46.2%
CS1315	3.5%	9.3%
COE1361b	15.9%	14.3%

Table 13: Answers on initial and final surveys.

	Raw Count	**Percentage**
Would take it	33	63.5%
Motivation:		
to fulfill requirement	69.2%	
for fun	20.5%	
to learn useful things	10.3%	
Wouldn't take it	15	28.8%
Not sure	4	7.7%
Total	52	100.0%

Table 14: Student interest in advanced media computation course

CS1315 were also asked whether they would likely take advanced media computation and to indicate their motivation (Table 14). The trends described in section 5.3 for female CS1315 students also apply to the class as a whole: while fewer than 10% of CS1315 students reported a desire to continue in CS, the number of students who reported that they would take an advanced media computation course exceeded 63%. This indicates that media computation has captured the interest of many students who otherwise would not choose to pursue computer science learning.

6.4. Future work on comparing the CS courses

These courses are changing considerably over the next year. COE1361 is becoming a CS course. CS1315 moves from being a pilot course, to being a traditional course taught by professors other than the original developer. It will be important to continue our evaluation and comparison efforts to see how things change. For example, the "hardware-first" approach to CS1 was invented by Yale Patt at the University of Michigan, but is no longer taught at the University of Michigan. Once Yale Patt left, they found that nobody else could teach the same class and get the same kinds of results, so they eliminated the course. We hope that CS1315 won't succumb to a similar fate.

Appendix A: Homework Assignments

(Appears at http://coweb.cc.gatech.edu/cs1315/46)
On homeworks, you are encouraged to collaborate with your fellow students. However, if you do work with others, you must list the people you worked with in the "Notes to TA" area of the JES turnin window.

Homework 1:

Write a program named **hw1** to accept a picture as input, and change its pixels as follows:

- Set the green component to 125% of its current value
- Decrease the blue by 25%
- Decrease the red by 75%

Turn in your program via JES as "hw1".

Ask questions on Spring 2003 Homework 1 Questions

Homework 2:

Replicate the splicing example to create the sound "We the UNITED people of the United States," but *normalize* the word "UNITED." Look for the maximum sample *just in that word*, compute the multiplier, and apply it *just to that word*. Call your function **louderSplice**.

Turn in your program via JES as "hw2".

Ask questions on Spring 2003 Homework 2

Homework 3:

Create a collage of the same image at least three times fit onto the 7x9.5in blank JPEG. (You are welcome to add additional images, too.) You can do any of

- scaling the image,
- cropping the image,
- creating a negative of the image,
- shifting or altering colors on the image,
- and making it darker or lighter.

Turn in your code, your original image (or images), and your resultant image as "hw3".

Upload the image if you wish to the Sp2003 HW3 Collages gallery.

Ask questions on Spring 2003 Homework 3

Homework 4:

Given a folder with images in it, create an index HTML page with links to each image.

You will write a function called **linksPage** that takes a string which is the path to a directory. You will create a page in that folder named `index.html`. Index.html should be an HTML page containing a link to every JPEG file in the directory. The **anchor** in the link should be a small version of the image, 100 pixels high.

Turn in your program via JES as "hw4". The function name should be "linksPage" and the program file name should be "hw4.py".

Ask questions on Spring 2003 Homework 4

Homework 5:

Build an animation of at least three seconds in duration (30 frames at 10 fps, or 75 frames at 25 frame per second). You must have at least two things in motion during this sequence. You must use at least one composited image (a JPEG image that you scale (if necesary) and copy into the image) and one drawn image (a rectangle or line or text or oval or arc -- anything that you draw).

Turn in your program and the composited image(s) via JES as "hw5". Your file must be named "hw5.py" and your function must be named "hw5" and accept a directory as input where the frames should be stored.

You must use more than one function in your solution. The main function will be "hw5" that takes input of a string that is a directory path. You must use at least one sub-function, so your file "hw5.py" will contain the function hw5() and at least one another.

Feel free to post your movies to Sp2003 HW5 Movies for sharing with others.

Ask questions on Spring 2003 Homework 5

Homework 6:

http://www.cnn.com is a popular news site. You are to write a function (named "hw6") that will input a directory as a string then:

- Visit http://www.cnn.com and pick out the *top three* news stories headlines in the **MORE NEWS** section. (Hint: The anchors for the news story headlines all have <div class="cnnMainNewT2"> before them. Find that tag, then search for the beginning of the anchor <a, then you can find the anchor text, which is the headline.)
- Create a ticker tape movie on the 640x480 canvas of all three news stories. Have one come across at y=100, another at y=200, and the third at y=300. Generate 100 frames, and don't have the ticker tapes move more than 5 pixels per frame. (In 100 frames, you won't make it all the way across the screen -- that's fine.) Store the frames to files in the input directory.

You must use more than one function in your solution. The main function will be "hw6" that takes input of a string that is a directory path. You must use at least one sub-function.

Turn in your code as "hw6". Your filename should be "hw6.py"

Ask questions on Spring 2003 Homework 6

Appendix B: Take-Home Exams

(Appears at http://coweb.cc.gatech.edu/cs1315/48)

Take Home Exam #1:

At the Feb. 7 class, we recorded this sound thisisatest2.wav ("This is a test."). Using MediaTools, we found the end points for each of the words in the sound:

Recorded word	**Index where it stops in the sound**
This	7865
is	27170
a	40326
test.	55770

Write a function **backSpliced** that splices the last word ("test") into the front of the word, but *backwards*, so that the result is: tsetisatest.wav ("Tset is a test.") It's okay if you have a variable that directly refers to the file on your disk: Your TA will fix it to point to his or her file when testing the program.

Some suggestions:

- Understand Recipe 15 (page 85). See the lecture notes for Feb. 10 on what it means to understand a program.
- Understand Recipe 14 (page 83), but you may want to build your program by editing the simpler splice on page 84. (That's how I wrote my solution to generate the above sound.) Be sure to use the slides for a more accurate version of Recipe 14.
- To do this right, you will write and run more lines of code than you put in your final solution. Do work at *understanding* the problem, then *writing* your solution.

This is a **NON-COLLABORATIVE ACTIVITY**! You may not talk to anyone about their code, nor look at anyone else's code, nor allow anyone to see your code. This is a **TAKE HOME *EXAM***. It is an "open book" exam. You may use your book, any slides, any material in the CoWeb, and any programs you've written (even with others) that you already have direct access to.

Ask questions on Spring 2003 Take Home Exam 1 but be sure to ask questions only about objectives and process. You are welcome to ask questions about any of the programs in the book or in the slides, but you cannot ask anyone (even the TA's or teacher) about your own solution.

Turn in your program via JES as "exam1". When you turn in your exam, you are to enter into the Comment area the statement: "I did not provide nor receive any aid on this exam." **IF YOU CANNOT MAKE THAT STATEMENT TRUTHFULLY, DO NOT SUBMIT YOUR EXAM!**

Take Home Exam #2:

Given a folder with images and sounds in it, create an index HTML page with links to each image and sound.

You will write a function called **indexPage** that takes a string which is the path to a directory. You will create a page in that directory named `index.html`. Index.html should be an HTML page containing a link to every JPEG file and every WAV in the directory. At the top of the page, put a heading (level 1) with the phrase "Directory listing of sounds and images in " and then the directory, e.g., "Directory listing of sounds and images in C:\Documents and Settings\Mark Guzdial\mediasources"

The links should be each an item in an unordered (<ul>) list. The **anchor text** in each link should be the filename of the image or sound. The **destination** (href) should be the same filename.

- *For each image, on the same line as the filename, list the **horizontal and vertical size (in pixels)** of the image.*
- *For each sound, list the length of the sound in **seconds**.*

For example, your listing might say:

- Filename: barbara.jpg, height 240 pixels, width 120 pixels
- Filename: hello.wav, length 4.2 seconds

Clicking on "barbara.jpg" above should display that file. Clicking on "hello.wav" should go to that sound (as the destination of the link)--and if your browser and sound card as set up appropriately, you should hear the sound.

Turn in your program via JES as "exam2". Your filename should be "exam2.py" Your function name should be **indexPage**

This is a **NON-COLLABORATIVE ACTIVITY**! You may not talk to anyone about their code, nor look at anyone else's code, nor allow anyone to see your code. This is a **TAKE HOME *EXAM***. It is an "open book" exam. You may use your book, any slides, any material in the CoWeb, and any programs you've written (even with others) that you already have direct access to.

Ask questions on Spring 2003 Take Home Exam 2 but be sure to ask questions only about objectives and process. You are welcome to ask questions about any of the programs in the book or in the slides, but you cannot ask anyone (even the TA's or teacher) about your own solution.

When you turn in your exam, you are to enter into the Comment area the statement: "I did not provide nor receive any aid on this exam." **IF YOU CANNOT MAKE THAT STATEMENT TRUTHFULLY, DO NOT SUBMIT YOUR EXAM! ANY EXAM WITHOUT THAT STATEMENT WILL NOT BE GRADED.**

Appendix C: Midterm Exam #1

(14) 1. Write a program with IFs.

Students in a class have taken three exams. Each exam is worth 1/3 of their final grades. Write a function **finalGrade** that takes in the three exam scores as inputs (e.g., **def finalGrade(exam1, exam2, exam3):)**and computes the final score. You should print the final score, and print the grade as follows:

If the final grade is:	the computer should print:
>=90	Congratulations, you got an A!
>=80 and <90	You got a B.
>=70 and <80	You got a C.
>=60 and <70	You got a D.
<60	Better luck next time.

For reference, you might consider these programs, copied and slightly modified from the midterm review:

```
def pay(hours, payrate):
  gross = hours * payrate
  if gross <= 100:
    tax = .25
  if 101 < gross < 300:
    tax = .35
  if 299 < gross < 400:
    tax = .45
  if gross > 399:
    tax = .50
  taxAmount = gross * tax
  print "net pay = ", gross - taxAmount
  print "gross pay = ", gross
```

```
def checkLuminance(r, g, b):
  newRed = int(r*.299)
  newGreen = int(g*.587)
  newBlue = int(b*.114)
  luminance = newRed + newBlue + newGreen
  if luminance < 10:
    print "That's going to be awfully dark"
  if luminance>50 and luminance < 200:
    print "Looks like a good range"
  if luminance > 250:
    print "That's going to be nearly white"
```

(20) 2. Tracing a program

```
def newFunction(a, b, c):
   print a
   list1 = range(1,5)
   value = 0
   for x in list1:
      print b
      value = value +1
   print c
   print value
```

If you call the function above by typing:

`newFunction("I", "you", "walrus")`

what will the computer print?

(21) **3. Which does which?**

```
def funcA():
  af=getMediaPath("barbara.jpg")
  a = makePicture(af)
  bf = getMediaPath("7inX95in.jpg")
  b = makePicture(bf)
  # Now, do the actual copying
  tx = 100
  for sx in range(45,200):
    ty = 100
    for sy in range(25,200):
      c = getColor(getPixel(a,sx,sy))
      setColor(getPixel(b,tx,ty), color)
      ty = ty + 1
    tx = tx + 1
  show(a)
  show(b)
  return b
```

```
def funcB(s):
  m = int(getWidth(s)/2)
  for y in range(1,getHeight(s)):
    for x in range(1,m):
      p = getPixel(s, x+m,y)
      p2 = getPixel(source, m-x,y)
      c = getColor(p2)
      setColor(p,c)
```

```
def funcC(s):
  for a in getSamples(s):
    v = getSample(a)
    setSample(a,v * 2)
```

```
def funcD(r):
  for p in getPixels(r):
    x=getRed(p)
    setRed(p,x*0.5)
```

```
def funcE(q):
  t = 0
 for s in getSamples(q):
    t = max(t,getSample(s) )
    b = 32767.0 / largest
  for s in getSamples(q):
        t =  b * getSample(s)
        setSample(s,t)
```

```
def funcF():
  af=getMediaPath("barbara.jpg")
  a = makePicture(af)
  bf = getMediaPath("7inX95in.jpg")
  b = makePicture(bf)
  x = 45
  for x2 in range(100,100+((200-45)/2)):
    y = 25
    for y2 in range(100,100+((200-25)/2)):
      color = getColor(getPixel(a,x,y))
      setColor(getPixel(b,x2,y2), color)
      y = y + 2
    x = x + 2
  show(barb)
  show(canvas)
  return canvas
```

For each of the below descriptions, write funcA, funcB, funcC, funcD, funcE, funcF, or "none" to indicate which function implements the give description:

_______ Takes a sound as input and doubles the amplitude (and thus, increases the volume)

_______ Takes a picture as input and decreases the red in each pixel by 50%

_______ Takes a sound as input and normalizes the volume by making the amplitudes as large as possible without clipping.

_______ Takes a picture as input and mirrors the image vertically: The left half is copied onto the right half.

_______ Copies (crops) just Barbara's face onto the canvas.

_______ Copies (crops) just Barbara's face onto the canvas, but makes it larger (scales up) in the copying.

_______ Copies (crops) just Barbara's face onto the canvas, but shrinks it by half (scales down) in the copying.

(20) **4. Concept Questions**

a. Why can't you hear CD-quality music well over a telephone?

b. What's the difference between an array and a matrix? Which do we use for representing a sound and which for a picture?

c. How does the frequency of the sound impact our sensation of the sound?

d. What is Python?

e. What is Moore’s Law?

(25) 5. **Generalized changeVolume**

Write a new version of the function **increaseVolume** and **decreaseVolume** (see below) called **changeVolume** that takes as input a sound *AND* an amount to increase or decrease the volume by. That amount will be a number between -.99 and .99.

- changeVolume(sound,-.10) should decrease the amplitude by 10%.
- changeVolume (sound,.30) should increase the amplitude in the sound by 30%
- changeVolume (sound,0) should do nothing at all to the amplitude.

Use as examples:

```
def increaseVolume(sound):
  for sample in getSamples(sound):
    value = getSample(sample)
    setSample(sample,value * 1.5)


def decreaseVolume(sound):
  for sample in getSamples(sound):
    value = getSample(sample)
    setSample(sample,value * 0.5)
```

Appendix D: Midterm Exam #2

(20 points) 1. What do these programs do?

```
def foo(a):
  x = 0
  y = 0
  for i in a:
    if i == "a":
      x = x + 1
    if i == "b":
      y = y + 1
  print y-x
```

a. What will this program print when you execute **foo("abracadabra")**

b. Will this program work for lists with single character elements, e.g., **foo(['a','b','c'])?** Why or why not?

```
def bar(sound):
  s = ""
  for i in range(1,100):
    v = getSampleValueAt(sound,i)
    if v > 0:
      s = s + "+"
    if v <= 0:
      s = s + "-"
  print s
```

c. This program **bar** prints out a string, based on the samples in the sound. What does the string represent?

d. How long (how many characters) will the output of this program be?

(25 points) 2. Write a function **rainfall** that will input a list of numbers, some positive and some negative, e.g., [12, 0, 41, -3, 5, -1, 999, 17]. These are amounts of rainfall. Negative numbers are clearly a mistake. Print the average of the positive numbers in the list. (Hint: The average is the total of the positive numbers divided by the average of just the positive numbers.)

You may want to recall these examples from lecture on how to manipulate lists:

```
>>> mylist = ["This","is","a", 12]
>>> print mylist
['This', 'is', 'a', 12]
>>> print mylist[0]
This
>>> for i in mylist:
...     print i
...
This
is
a
12
>>> print mylist + ["Really!"]
['This', 'is', 'a', 12, 'Really!']
```

Extra credit (5 points): Write the function so that, if the number 999 appears in the list, *add in no later numbers in the list*. So, if the above example were input, the 17 would not be added into the average.

(28 points) 3. Short Essay

A. What are general rules for when you should write a program to create a graphic or a sound, and when you should just use application software like Photoshop to create a graphic or a sound?

B. What's the difference between a function and a method?

C. Why is a tree a better representation for files on a disk than an array? Why do you have many directories on your disk, and not just one gigantic one?

D. Internet addresses are four numbers, each between 0 and 255. How many bits are in an Internet address?

(27 points) 4. Let's imagine that you are keeping track of all of your contacts in a file named **contacts.txt** in your JES directory. The format of the file is name, colon, company, colon, phone number, colon, and key phrases that you want to remember about this contact. (You may assume that names, companies, phone numbers, and key phrases won't have colons in them.) The file might look like this:

Calvin:Calvin & Hobbes:990-8979:Cartooning, tiger tamer, philosophy
Luke:Skywalker Ranch:897-7765:Lightsaber battles, martial arts, philosopher
Arthur:Camelot Inc.:453-8976:Holy grail seeker, sword puller, king

Write a function called **contact** that will take a keyword in a string as input. If that keyword is found in the contacts file, print the name, company, and phone number of the contact. If it is not found, print "Not Found". So, if you executed **contact("king")** for the above file, you would get printed out "Arthur:Camelot Inc.:453-8976"

You may want to use these programs from lecture as reference.

```
def findSequence(seq):
  sequencesFile = getMediaPath("parasites.txt")
  file = open(sequencesFile,"rt")
  sequences = file.read()
  file.close()
  # Find the sequence
  seqloc = sequences.find(seq)
  #print "Found at:",seqloc
  if seqloc != -1:
    # Now, find the ">" with the name of the sequence
    nameloc = sequences.rfind(">",0,seqloc)
    #print "Name at:",nameloc
    endline = sequences.find("\n",nameloc)
    print "Found in ",sequences[nameloc:endline]
  if seqloc == -1:
    print "Not found"
```

```
def findTemperature():
  weatherFile = getMediaPath("AtlantaWeather1.html")
  file = open(weatherFile,"rt")
  weather = file.read()
  file.close()
  # Find the Temperature
  humloc = weather.find("Humidity")
  if humloc != -1:
    # Now, find the "," where the temp starts
    temploc = weather.rfind(",",0,humloc)
    endline = weather.find("<",temploc)
    print "Current temperature:",weather[temploc+1:endline]
  if humloc == -1:
    print "They must have changed the page format -- can't find the temp"
```

Appendix E: Final Exam

1. Let's imagine that you are keeping track of all of your contacts in a file named **contacts.txt** in your JES directory. The format of the file is pound sign, name, colon, company, colon, phone number, colon, and key phrases that you want to remember about this contact. (You may assume that names, companies, phone numbers, and key phrases won't have colons in them.) The file might look like this:

#Calvin:Calvin & Hobbes:990-8979:Cartooning, tiger tamer, philosophy
#Luke:Skywalker Ranch:897-7765:Lightsaber battles, martial arts, philosopher
#Arthur:Camelot Inc.:453-8976:Holy grail seeker, sword puller, king

Write a function called **contact** that will take a keyword in a string as input. If that keyword is found in the contacts file, print the name, company, and phone number of the contact. If it is not found, print "Not Found". So, if you executed **contact("king")** for the above file, you would get printed out "Arthur:Camelot Inc.:453-8976"

You may want to use these programs from lecture as reference.

```
def findSequence(seq):
  sequencesFile = getMediaPath("parasites.txt")
  file = open(sequencesFile,"rt")
  sequences = file.read()
  file.close()
  # Find the sequence
  seqloc = sequences.find(seq)
  #print "Found at:",seqloc
  if seqloc != -1:
    # Now, find the ">" with the name of the sequence
    nameloc = sequences.rfind(">",0,seqloc)
    #print "Name at:",nameloc
    endline = sequences.find("\n",nameloc)
    print "Found in ",sequences[nameloc:endline]
  if seqloc == -1:
    print "Not found"
```

```
def findTemperature():
  weatherFile = getMediaPath("AtlantaWeather1.html")
  file = open(weatherFile,"rt")
  weather = file.read()
  file.close()
  # Find the Temperature
  humloc = weather.find("Humidity")
  if humloc != -1:
    # Now, find the "," where the temp starts
    temploc = weather.rfind(",",0,humloc)
    endline = weather.find("<",temploc)
    print "Current temperature:",weather[temploc+1:endline]
  if humloc == -1:
    print "They must have changed the page format -- can't find the temp"
```

2. You're thinking about adding something to your HTML home page generator that will make random, relevant comments about the weather depending on the temperature.

- If it's going to be less than 32, you want to insert either "Watch out for ice!" or "Did I move North?!?"
- If it's going to be between 32 and 50, you want to insert either "I can't wait for winter to be over!" or "Come on, Spring!"
- If it's over 50 but less than 80, you want to insert either "It's getting warmer!" or "Light jacket or less weather."
- If it's over 80, you want to insert either "FINALLY! Summer!" or "Time to go swimming!"

Write a function named **weathercomment** that will take a temperature as input and *return* one of the phrases *randomly*. In other words, you want to use the temperature to decide which are the relevant phrases, then pick one randomly from there.

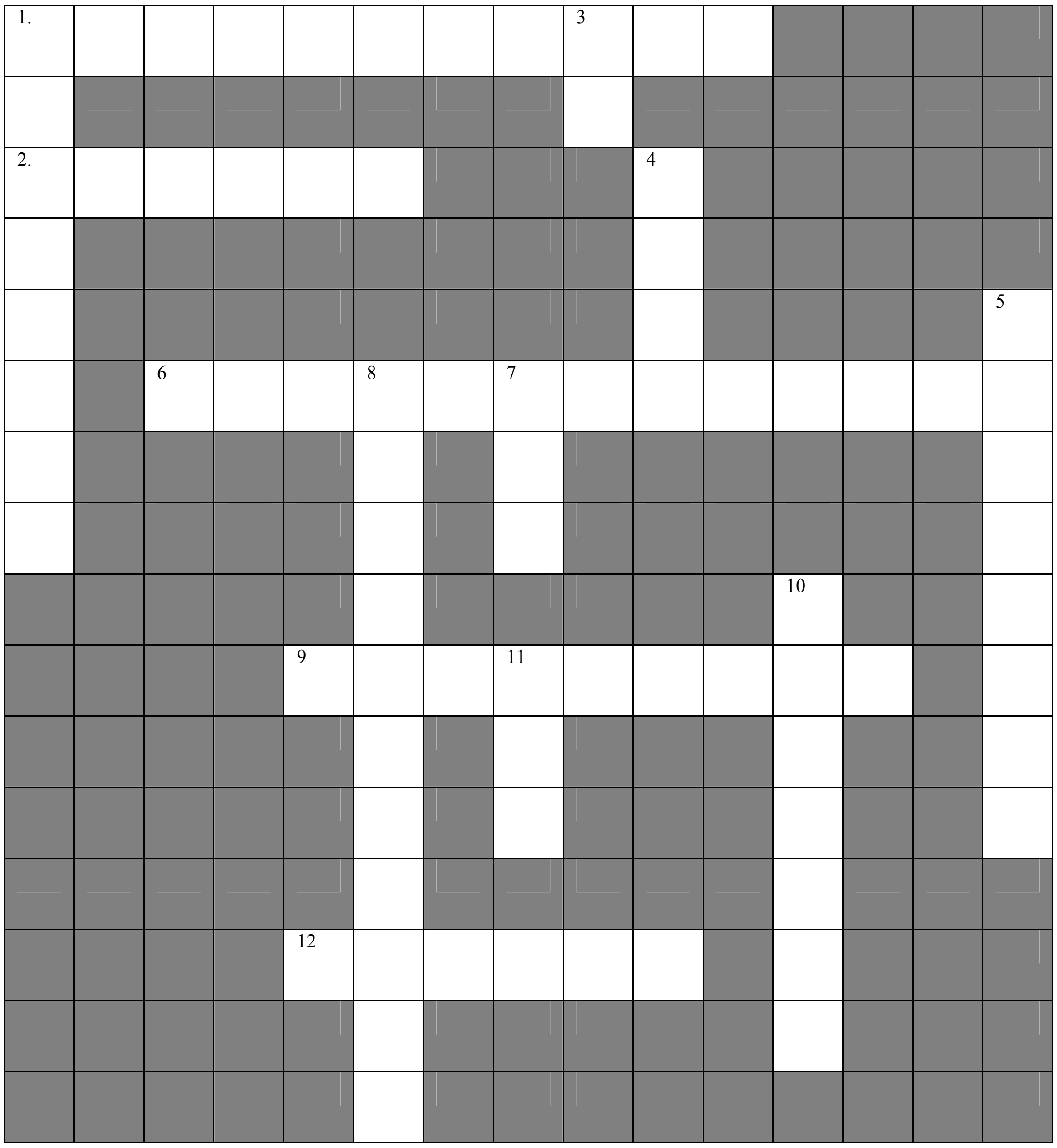
You might recall these programs from class and exam review as examples. The top one generates gendered sentences, and the bottom one uses *return* to return a sentence to be inserted into

```
import random
def sentence(x):
  nounsM = ["Mark", "Adam", "Larry", "Jose", "Matt", "Jim"]
  nounsF = ["Angela", "Laura", "Meghan", "Mary"]
  verbs = ["runs", "skips", "sings", "leaps", "jumps", "climbs",
"argues", "giggles"]
  phrases = ["in a tree", "over a log", "very loudly", "around the
bush", "while reading the Technique"]
  phrases = phrases + ["very badly", "while skipping","instead of
grading", "while typing on the CoWeb."]
  if x =="M":
    print random.choice(nounsM), random.choice(verbs),
random.choice(phrases)
  if x=="F":
    print random.choice(nounsF), random.choice(verbs),
random.choice(phrases)
```

```
import random

def sentence():
  nouns = ["Mark", "Adam", "Angela", "Larry", "Jose", "Matt", "Jim"]
  verbs = ["runs", "skips", "sings", "leaps", "jumps", "climbs",
"argues", "giggles"]
  phrases = ["in a tree", "over a log", "very loudly", "around the
bush", "while reading the Technique"]
  phrases = phrases + ["very badly", "while skipping","instead of
grading", "while typing on the CoWeb."]
  #print random.choice(nouns), random.choice(verbs),
random.choice(phrases),".‟
  return random.choice(nouns)+" "+random.choice(verbs)+"
"+random.choice(phrases)+"."
```

3. Solve the crossword puzzle.
(Clues on next page)



Across:	**Down**
1. The property of methods that they can apply to more than one object. 2. A computational structure that has both data and behavior. 6. A property of objects that each holds its own data and behavior, that we can't reach inside another object. 9. What we call a function calling itself. 12. A specific function that takes a function as input and a list; it applies the function to each item of the list and returns just those items for which the input function returns true.	1. An agreement for how two computers will interact; examples are HTTP and FTP. 3. __ address: The four numbers each between 0 and 255 that define an Internet computer. 4. A biological entity that scales well and is robust; a model for objects. 5. A network of networks based on a set of agreements. 7. A common language for database creation and manipulation. 8. A word that describes how objects can be combined (objects within objects) to create more complex structures. 10. A property of functions that are well-defined in such a way that the function does one and only one goal. 11. The address of something on the Web (abbreviation); Consists of a protocol, a server domain name, and the path to the page, image, or other object.

4. Applying your knowledge in new situations.

(a) Your mother shows you this cool new image processing program she bought. "I can finally get rid of red-eye!" When she clicks the red-eye removal button, she:

- Draws a circle with her mouse around the red area in the eye.
- Then clicks on a palette of colors of what the eye color *should* be.
- And all the red is replaced with the selected color.

She says, "*How did that work?* That's amazing. How can the computer figure out what part is red?" Answer your mother's questions, (i) How did the red get replaced by the selected color? and (b) how did the computer know what part was red?

(b) You have a new computer that seems to connect to the Internet, but when you try to go to http://www.cnn.com you get a "Not Found" error. You call tech support, and they tell you to try to go to http://64.236.24.20 That works.
Now both you and the Tech know what's wrong with your computer's settings. What isn't working properly since you can get to a site via the Internet but can't get the domain name www.cnn.com to be recognized?

(c) Your father calls you. “My tech support people are saying that the company website is down because the database program is broken. What does the *database* have to do with our company *website*?”
You explain to him how databases can be integral to running large websites. Explain both (a) how the website comes to be authored through the database and (b) how the HTML is actually created.

5. Short Essay

(a) What are three reasons why someone should use multiple functions in their programs, rather than one big function?

(b) What is a protocol and where is it specified in a URL?

(c) Is it possible to write a program to solve the *Traveling Salesman Problem*? Is it possible to write a program to solve the *Halting Problem*?

(d) What does a compiler most commonly produce? It takes a program written in a language like C or Java and it generates something new. What does it generate?

6. . **NAME THAT ALGORITHM!**

(A) Consider these two programs.

```
def halfFreq(filename):
  source = makeSound(filename)
  target = makeSound(filename)

  sourceIndex = 1
  for targetIndex in range(1, getLength( target)+1):
    setSampleValueAt(                target,                targetIndex,
        getSampleValueAt( source,
        int(sourceIndex)))
    sourceIndex = sourceIndex + 0.5

  play(target)
  return target
```

```
def copyBarbsFaceLarger():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  sourceX = 45
  for targetX in range(100,100+((200-45)*2)):
    sourceY = 25
    for targetY in range(100,100+((200-25)*2)):
      color = getColor(
        getPixel(barb,int(sourceX),int(sourceY)))
      setColor(getPixel(canvas, targetX, targetY), color)
      sourceY = sourceY + 0.5
    sourceX = sourceX + 0.5
  show(barb)
  show(canvas)
  return canvas
```

What algorithm is being implemented in these two examples? Explain the algorithm in English.

B. Consider these two programs:

```
def findInSortedList(something, alist):
  for item in alist:
    if item == something:
      return "Found it!"
  return "Not found"
```

```
def turnRed():
  brown = makeColor(57,16,8)
  file = r"C:\Documents and Settings\Mark Guzdial\My
Documents\\mediasources\barbara.jpg"
  picture=makePicture(file)
  for px in getPixels(picture):
    color = getColor(px)
    if distance(color,brown)<100.0:
      redness=getRed(px)*1.5
      setRed(px,redness)
  show(picture)
  return(picture)
```

What algorithm is being implemented in these two examples? Explain the algorithm in English.

C. Consider this program.

```
def findInSortedList(something, alist):
  start = -1
  end = len(alist)
  while (start + 1) < end:
    checkpoint = int((start+end)/2.0)
    if checkpoint < 0 or checkpoint > len(alist):
      break
    printNow("Checking at: "+str(checkpoint)+" start="+str(start)+"
end="+str(end))
    if alist[checkpoint]==something:
      return "Found it!"
    if alist[checkpoint]<something:
      start=checkpoint
    if alist[checkpoint]>something:
      end=checkpoint
  return "Not found"
```

What algorithm is being implemented in this example? Explain the algorithm in English.