End of Course Memo
CS 101 – Intro to Computing
Aaron Bloomfield (Spring 2005)

Course Objectives:

1. Understand fundamentals of programming such as variables, conditional and iterative execution, methods, etc.
2. Understand fundamentals of object-oriented programming in Java, including defining classes, invoking methods, using class libraries, etc.
3. Be aware of the important topics and principles of software development.
4. Have the ability to write a computer program to solve specified problems.
5. Be able to use the Java SDK environment to create, debug and run simple Java programs.

Assessment of Learning by Course-Objective:

For the assessment of these objectives, I analyze the scores of the various course assignments (homeworks, exams, and programming quizzes). The first two midterms were very easy (averages of 86.8% and 85.4%, respectively), the third was fair (average of 75.2%). The homeworks were also very easy, and those grades were also inflated (the average on the homeworks was 94.6%).

Objective 1: Understand fundamentals of programming such as variables, conditional and iterative execution, methods, etc.
Evidence that this objective was met can be seen through the lab programming quizzes and the homeworks. The last lab quiz was the most comprehensive, as it included the concepts taught throughout the entire course (iteration, conditional statements, OOP, defining classes, writing a computer program to solve a program, using the JDK, etc.). The average on the last lab quiz, which was the most comprehensive, was 89.1%. The homework average was 94.6%, but as mentioned above, that number is inflated.

Objective 2: Understand fundamentals of object-oriented programming in Java, including defining classes, invoking methods, using class libraries, etc.
This objective was met, and the evidence is the same as that for objective 1.

Objective 3: Be aware of the important topics and principles of software development.
I do not feel we met this objective very well, and this is something I plan to change in the future. The only software development the students saw was for small Java programs, and that does not constitute this objective. In particular, I plan on bringing in faculty to discuss their research and software development. Quantitatively measuring these additions will be difficult, however.

Objective 4: Have the ability to write a computer program to solve specified problems.

This objective was met, and the evidence is the same as that for objective 1.

Objective 5: Be able to use the Java SDK environment to create, debug and run simple Java programs.
This objective was met, and the evidence is the same as that for objective 1.

Assessment of Changes Made in the Course:

One of the biggest changes made to the course was in the grading. In the past, the grading was done on paper (which, for 520 students in 101 and 101-E, wasted a **lot** of paper). The current grading system I developed allows all of this to be done online. The students submit their assignments online, the grading is done (by the TAs) online, and the results are e-mailed back to the students. This allowed for a much more efficient use of the teaching assistants time, as well as saving a few forests.

Although lab attendance was required in the past, it was first enforced this semester. The students received a lab grade (within a week of their lab), which was a zero if they were not present. In the past, they were told at the very end of the semester their lab grade.

The students were given easy access to their grades and their graded assignments. Indeed, we went a bit overboard on this – the students had access to their weighted average, which caused them to focus on that average rather than learning the material. This is going to be changed in future semesters (they can compute their average themselves, but it will not be computed for them).

A number of the labs were redone and/or improved on. A few of the labs (in particular, the fourth lab) will need to be reworked for next semester.

Student evaluations were improved over last semester. The students submitted evaluation data for each homework assignment, for such questions as the difficulty of the assignment, how long they spent on it, etc. This allowed for analysis of the student opinions of the various assignments.

Other Issues:

1. Do you have concerns regarding the background of students coming into the course?

No. The students are not assumed to have any background in any computer field for this course.

2. Are there other issues affecting student learning beyond what has been discussed elsewhere in this report? Include any other concerns you have about what students have or have not learned when they have completed the course.

Lots. The course is broken in so many ways. Having a lecture of 420 students is a terrible way to teach any subject, much less computer programming. If the department and/or school were serious about improving the student experience in this course, this would be the aspect to tackle first. The school/dept needs to devote more resources to this course (not just add more course sections to already overworked faculty). The state of the lab room was absolutely terrible (I repeatedly felt it necessary to apologize for the state of the room) – the computers were repeatedly down, and the furniture was pathetic. Although this is being fixed (to some extent) at the end of this semester, the students should not have had to put up with that this semester at all. There is a general lack of teaching assistant support for this class (I had to fight very hard to get a decent amount of undergrad TA support). The amount of funding for undergraduate teaching assistants (who are generally **better** than the graduate teaching assistants) was much lower than what was needed to run a good course. The prevalent department attitude about graduate student teaching (specifically, that their teaching is not very important and their research is) – while it might be good for publications, is **not** good for the pedagogical experience of the students in the course. I had serious problems with one teaching assistant this semester, and was told there was little I could do about it. Adding a recitation section would help improve student learning. The course needs to focus more on problem solving, and less on coding – this is something that is going to be changed for next semester. The course also needs to show why computer science is interesting, and how there is more to it than just programming in Java (this is also something that is going to change for next semester).

3. If you know of changes being made or considered in the curriculum that might affect the course, briefly describe what these are and how the course might be affected.

Some other departments are trying to replace CS 101 with a different type of introductory course, such as one based on Matlab. This is not likely to affect the CS curriculum, as there will still be a Java section. But it will affect the core classes that all incoming Engineering students take. A potential problem is if a student takes the Matlab-based version of the introductory course, and then wants to switch into the computer science major – he or she will not have the Java background necessary to move into CS 201, and will have to repeat the introductory course.

4. List any other comments you think the Committee that monitors our degree programs should know about this course this semester.

Lower the class size! It's ridiculous to have a computer science lecture that has 420 students! Better labs; more TA funding; better departmental attitude about graduate student teaching; etc. See my discussion above.

Mapping of Course Objectives to BSCS Outcomes:

CS Degree Outcomes: Students who graduate with a BSCS will…	**Course Obj. 1**	**Course Obj. 2**	**Course Obj. 3**	**Course Obj. 4**	**Course Obj. 5**
(1: Math & DLD) Have demonstrated comprehension in relevant areas of mathematics (including calculus, discrete math, and probability), and in the area of logic design.					
(2: Fundamentals) Have demonstrated comprehension in fundamental topics of computing, including the intellectual core of computing, software design and development, algorithms, computer organization and architecture, and software systems.	D	D		D	D
(3: Analysis & Evaluation) Have applied knowledge of areas of computing to analyze and evaluate algorithms, designs, implementations, systems, or other computing artifacts or work-products. Application of this knowledge includes the ability to design, conduct and evaluate the results of experiments and testing activity.	D	D		D	
(4: Build Solutions) Have applied knowledge of areas of computing to create solutions to challenging problems, including specifying, designing, implementing and validating solutions for new problems.	D	D		D	
(5: Research Awareness) Be aware of current research activity in computing through activities including reading papers, hearing research presentations, and successfully planning and completing an individual research project in computing or its application.			F		
(6: Broadening) Have demonstrated comprehension of subjects in the humanities, social sciences, and the natural sciences in order to broaden a student's education beyond engineering and computing.					
(7: Social and Professional) Comprehend important social, ethical, and professional considerations related to computing practice and research, and be able to apply this knowledge when analyzing new situations.					
(8: Post-graduation) Be prepared to enter graduate programs in computing or related fields, and be prepared to begin a professional career in computing.					
(9: Life-long Learning) Have demonstrated a self-directed ability to acquire new knowledge in computing, including the ability to learn about new ideas and advances, techniques, tools, and languages, and to use them effectively; and to be motivated to engage in life-long learning.					
(10: Teamwork) Have demonstrated the ability to work effectively in a development team.					
(11: Communication) Have demonstrated the ability to communicate effectively (orally and in writing) about technical issues.					
(12: Professional development practices) Comprehend important issues related to the development of computer-based systems in a professional context using a well-defined process to guide development.	D	D		D	D