End of Course Memo
CS 101 – Intro to Computing
Aaron Bloomfield (Spring 2006)

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 last three midterms this semester were unintentionally a bit on the easy side (averages of 88%, 83%, and 85%, respectively). The homeworks required a lot of work, but generally had high scores (average of 92%), due to the fact that the students could receive as much TA help as they wanted. Note that these averages are for all students in either 101 or 101-E.

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 and the most difficult, was 78%. The homework average was 92%, but as mentioned above, that number is somewhat 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.
This objective was met much better this semester than in the past. In particular, the big game at the end (see below) showed the a lot of the concepts related to programming large programs with interacting objects. I plan to do more on this next semester by using case studies, presentations by other faculty, etc.

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:

This course is a difficult one to make significant changes during the spring semester, due to the sheer volume of the number of students in the course (447 in 101 and 101-E combined). A number of changes were made last fall, and these changes were kept this semester – including using the game as the final project (see below), increasing the work load, etc.

The role of the undergraduate teaching assistants was improved this semester. There were 15 undergraduate TAs, the largest number ever in 101. There were also 5 graduate TAs. All of the TAs held office hours, which, when combined with the faculty members' office hours, provided the students with 50 hours during the week that there was a course staff member available for office hours. This was very well received by the students. In addition, the undergraduate TAs also had a bigger role in the course in terms of commenting on the various assignments, etc. I plan on increasing their role in future semesters.

I have been working hard at creating a set of experienced undergraduate teaching assistants, and this really paid off this semester. Of the 12 undergraduates who have TA'ed for me for CS 101 in the past, 10 of them returned this semester (the other 5 undergraduate TAs were new this semester). This experience was invaluable for allowing the course to run smoothly, and was a great asset. As an example, I was able to hold the first lab 2 days after I was alloted the teaching assistants – the fact that many were experienced allowed for this to occur without any hitches. The grading also went more smoothly, as there was no learning curve for the experienced TAs to learn how things are graded in this course.

The last 3 homeworks and the last 3 labs were all components of a larger computer program (in this case, a economic trading game). This idea was started last fall, and we were able to improve on it this semester. The game involved the students moving from one location to another, buying and selling various goods. We improved upon the issues we encountered last semester, and see further ways to improve it for next semester. The students got a chance to see how objects in a big system interact, as well as seeing how large programs are developed. They also enjoyed a number of aspects of this: both developing a big program (over 1,000 lines), and writing a computer game. I intend to keep this for next semester, although the resulting large program will have a different focus.

A number of labs were improved upon, mostly labs that were very poorly reviewed by students from last semester. In addition, 3 labs (of the 12 given) were created for this semester. These were components of the game, and thus probably can't be used next semester (we do not re-use assignments).

We required the students to include test code in their programs. The intent was to make them think about how to test their code. While a good idea, we see ways it can be better implemented next semester.

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 356 students in the CS 101 lecture is a terrible way to teach any subject, much less computer programming. More resources need to be devoted to this course (and not just by adding more course sections to already overworked faculty). Adding a recitation section would help improve student learning, although how to do this in a class that is only 3 credits and already has a lab is unknown. The course needs to focus more on problem solving, and less on coding – this is something that has changed some for this semester, and I plan to work on this more for next semester. **Java as a introductory programming language is a terrible choice, and should be replaced.** 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 was changed for this semester, and needs to be changed more 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 want 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 tries 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 (which may not be allowed by UVa).

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 this many students!

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