CS 201 – Computer Programming II
Course Syllabus and Calendar – Fall 2002

Professor Don Colton

Brigham Young University Hawaii

PRELIMINARY DRAFT

1 Brief Overview

Simply stated, this class provides Intermediate-level Programming Proficiency.

By now you probably have a clue of what programming is about, what it looks like, how it acts, its joys and frustrations. You are no longer clueless. But you are not confident and capable yet. Alternately, you have taken some Calculus and developed skills in symbolic manipulation and problem solving that should put you on a level playing field with the other students in the class. *If you do not have prior Calculus or programming experience, you should confer with the instructor to make sure this is the right class for you.*

This course, Computer Programming II (CS 201), is designed to take you to the next level. By the time you finish this course, you will be programming web pages. Not just HTML, but Forms and server-side CGI, relational data base entry, maybe even games. You will have some marketable skills. If you are a Math major or an Information Systems major, completion of this course will constitute "programming proficiency" for your major.

In the case of **Information Systems** majors, IS is not all about programming, but those without programming skills have limited mobility in the after-graduation job market.

In the case of **Mathematics Education** majors, you may be the most computer-savvy person at your school, and may get the opportunity to introduce students to computer programming.

In the case of **Mathematics** majors, some real programming ability is assumed in today's job market.

In the case of **Computer Science** majors, or other students who want more than an intermedate level of programming proficiency, there is yet more to learn. This course is a stepping stone to Computer Programming III (CS 202), where you will apply your 201 skills to Object-Oriented programming in a language like Java or C++.

1.1 The Course

- **Course Number:** CS 201
- **Title:** Computer Programming II
- **Course Description:** Review of CS 101. Problem solving, stacks, queues, hash tables, mathematical analysis of algorithms, regular expressions. Web programming including CGI and database. (Prerequisites: CS 101 and Math 110; or Math 112; or Math 119.)
- **Textbook (recommended):** *Perl by Example*, by: Ellie Quigley. ISBN 0-13-028251. SRP $44.99.
- **Section 1:** Class Time: MWF 8:00–8:50 AM
  Final Exam: Fri 13 Dec, 7:00–10:00 AM
  Classroom: GCB 140
- **Section 2:** Class Time: MWF 9:00–9:50 AM
  Final Exam: Mon 9 Dec, 7:00–10:00 AM
  Classroom: GCB 101 (CS Lab)

1.2 The Instructor

- **Instructor (me):** Don Colton
- **My email:** don@colton.byuh.edu
- **My Office:** GCB 130 B
- **Office Hours:** TTh 9 AM to 3 PM

1.3 The Tutors

- **T.A. Hours:** Mon–Thu, 3 PM to 1 AM
- **T.A. Hours:** Fri, 3 PM to 8 PM
- **T.A. Hours:** Sat, 3 PM to 6 PM
- **T.A. Location:** GCB 101 (CS Lab)

1.4 Special Needs

Brigham Young University Hawaii is committed to providing a working and learning atmosphere, which reasonably accommodates qualified persons with disabilities. If you have any disability that may impair your ability to complete this course successfully, please contact the students with Special Need Coordinator, Leilani A'una at 293-3518. Reasonable academic accommodations are reviewed for all students who have qualified documented disabilities. If you need assistance or if you feel you have been unlawfully discriminated

against on the basis of disability, you may seek resolution through established grievance policy and procedures. You should contact the Human Resource Services at 780-8875.

1.5 Preventing Sexual Harassment

Title IX of the education amendments of 1972 prohibits sex discrimination against any participant in an educational program or activity that receives federal funds, including Federal loans and grants. Title IX also covers student-to-student sexual harassment. If you encounter unlawful sexual harassment or gender-based discrimination, please contact the Human Resource Services at 780-8875 (24 hours).

1.6 Under Construction

This course is under construction. We taught it under the number IS 231 for three years, and recently decided to remodel it to provide more usable skills in the context of the existing CS 101 and CS 202 courses. This paragraph is a warning that things may shift more than normal as we move through the semester. In particular, the Database portions of the class are being newly developed this term.

1.7 Grading

It is my intent that grading be rather flexible: there are three grading methods. Your final grade will be computed using each of the methods, and the highest grade will be yours.

All three methods are based on points, and all three methods require you to pass the final. The methods differ in the mix of points from various class activities.

Your total-points grade is based on 1000 points of assigned work. Some extra credit points are also available. The total-points grading will be as follows:

930+	A	900–929	A-	870–899	B+
830–869	B	800–829	B-	770–799	C+
730–769	C	700–729	C-	670–699	D+
630–669	D	600–629	D-	0–599	F

Method 1: The Typical This method is based to a larger extent on your activity in the class. Major points are earned by reading and attendance.

att	attendance	150 pts
read	readings	150 pts
pgm	programming labs	400 pts
qtc	testing center quizes	100 pts
qic	in-class quizes	100 pts
final	final exam (in class)	100 pts
tot	total points assigned	1000 pts

Method 2: The Natural This method is more suited to those that find reading a burden because after the a brief glance, or just from listening in class, they know what to do and can do it.

att	attendance	50 pts
pgm	programming labs	500 pts
qtc	testing center quizes	150 pts
qic	in-class quizes	150 pts
final	final exam (in class)	150 pts
tot	total points assigned	1000 pts

Method 3: The Genius This method is suited to those geniuses that do really well on the final, but cannot be bothered to do the day-to-day work because it is so easy that it is boring.

pgm	programming labs	500 pts
qtc	testing center quizes	100 pts
qic	in-class quizes	100 pts
final	final exam (in class)	300 pts
tot	total points assigned	1000 pts

In All Cases: In all cases, you must take and pass the final exam. Your final grade will be the **lower** of your total-points grade and your final-exam grade.
To get an overall final grade above a C, you must also pass the final exam (five to ten programming problems, ten to twenty points each) with a sufficient score. The final-exam grading will be as follows:

93+	A	90–92	A-	87–89	B+
83–86	B	80–82	B-	77–79	C+

Grading is discussed further below. Some of the discussion is based on old course outlines and is under revision. In case of a conflict, you should trust the numbers above.

1.8 Office Hours

My office hours for Fall 2002 are TTh 9–3. Updated office hours (when necessary) are posted outside my office door. Students for whom the posted hours are not convenient can contact me by email to make an appointment.

I also have an open-door policy, posted on my office door as follows: "If my door is open (even just a bit) feel free to knock and come in. – Bro. Colton"

1.9 Subject to Change

It is possible that I will revise some aspects of the course as we go along. Any changes I make are likely to be to your advantage. If any of my changes seems unfair to you, let me know. I will try to correct it.

2 Now, About the Course

I assume that you want the ability to use programming in your future career. At the successful conclusion of this course, you will have programming proficiency in Perl. In today's job market (2002) and economy, this proficiency is not enough. There are many people qualified and experienced and out of work, and this raises the bar for finding a job. But if you want to do your own programming and not be at the mercy of others, this course will prepare you.

We will develop your programming skills by completing projects in areas that support electronic commerce on the web. We will develop your knowledge of a number of additional topics that you are likely to encounter in programming.

Knowledge of operating systems is also very important. Today's client-side world seems dominated by Microsoft Windows, but there is a strong server-side presence from Unix. UNIX and Windows are the two operating environments that I believe will dominate the IS computing world in the next decade and beyond. Therefore, this class utilizes UNIX to a modest degree. You will know the most commonly used commands, including those for file system maintenance (how to move from directory to directory, make new directories, move, rename, and delete files, etc.). You will know how to operate the most prominent free-software text editor, EMACS.

At the end of this course, you should feel comfortable listing Perl, UNIX, and EMACS among your skills on you résumé.

2.1 What is the Course Like?

Much like CS 101, you will write programs that are graded by my robotic grader, GradeBot. Class time will be devoted to understanding basic concepts of computer programming as applied to the World Wide Web.

Some of the programs will be graded by demonstration in class. In such cases you will have more freedom in the user interface. Programs graded by GradeBot allow freedom in the methods used to construct the answer, but they do not allow freedom in how the answer is presented: the user interface is specified for you.

2.2 Prerequisites

The prerequisite is CS 101 (Computer Programming I). I assume that you have written some programs. You know how to do formatted printing, and use **if**, **else**, **while**, **do while**, **for**, and subroutines. I assume that you have some skill, but you are probably not ready to sit in an interview and claim that you are a programmer.

Alternately, you can take the class if you have already completed some Calculus. This is because your level of mathematical maturity will allow you to learn the CS 101 material during our review (the first three weeks of class).

3 Course Content

The CS 201 course covers the following CC2001 Knowledge Units. These are defined in Computing Curricula 2001, a joint project of IEEE-CS and ACM. The IEEE Computer Society and the Association for Computing Machinery are the two major professional societies in computer science.

Much of this material is covered in more than one course.

PF1. Fundamental programming constructs

We review material that was introduced in CS 101.
- Basic syntax, semantics of a higher-level language
- Variables, types, expressions, and assignment
- Simple I/O
- Conditional and iterative control structures
- Functions and parameter passing
- Structured decomposition

PF2. Algorithms and problem-solving

- Problem-solving strategies
- The role of algorithms in problem-solving process
- Implementation strategies for algorithms
- Debugging strategies
- The concept and properties of algorithms

PF4. Recursion

We introduce material covered more fully in CS 202.
- The concept of recursion
- Recursive mathematical functions
- Simple recursive procedures
- Divide-and-conquer strategies

AL1. Basic algorithmic analysis

We introduce material covered more fully in CS 301.
- Asymptotic analysis of upper complexity bounds
- Differences among best, average, and worst case
- Big O, little o, Ω (omega), and Θ (theta) notation
- Standard complexity classes (linear, log, exponential)

NC4. The web as an example of client-server computing

- Web technologies
  - Server-side programs
  - Common gateway interface (CGI) programs
  - Client-side scripts
- Characteristics of web servers
  - Handling permissions

  - File management
  - Capabilities of common server architectures
- Role of client computers
- Nature of the client-server relationship
- Web protocols
- Tools for web site creation and management
- Developing Internet information servers
- Publishing information and applications

IM5. Database query languages

We introduct material covered more fully in IS 351.
- Overview of database languages
- SQL (query formulation, update sublanguage)
- Embedding queries in a procedural language

4 Grading

Your grade is earned by getting points for completing labs, readings, and tests. Once your computer account is set up, progress reports are available to you by computer at any time.

Deadlines: Each assignment has a deadline. You can see these deadlines by sending email to GradeBot (see below) asking for a *status* report. Most deadlines are "soft." Before the deadline an item is worth a certain number of points (100%). After the deadline, it is worth somewhat less (usually one point) each day until it reaches maybe 60% of its original value. It then remains near the 60% level until the last day of class. All work must be completed by the end of the last day of class. The final exam has a separate deadline.

The in-class quizes and the final exam are programming tests. You will be asked to write several fairly simple programs. The tests must be taken in class at the designated time. This is because anyone with advance knowledge of the test could memorize answers easily, so I must make sure no one has advance knowledge. If you must take the test at some other time, I must create a totally separate test for you.

Incomplete and UW: If you quit working in the class before achieving a passing grade, I will probably give you a "UW" grade instead of an "F."

I do not give "I" grades (incompletes) except in unusual circumstances. In my experience only a small fraction of incompletes are ever completed. I will consider giving you an incomplete if you request it, seem to have a good reason, have a pretty solid time line for completion, and you get the necessary paperwork filled out.

5 Work (No Pain, No Gain)

Most of your time will be spent writing programs. I am not sure how much time it would take a good student programmer to complete all of these assignments. A professional could probably do all of them in a week. Maybe less. But you are not a professional yet. The work is difficult mainly because it is unfamiliar. Our task is to make it familiar, and therefore easier. You will find that assignments you did in three or four hours early in the semester can be done much more quickly later in the semester. You should feel a great sense of achievement.

Reading: This is the first semester that I am using *Elements of Programming with Perl* by Johnson. I have read portions of the book and it seems quite good. I also received a very strong positive recommendation from someone I know. For these reasons I have adopted the book as the text for this class. I am especially interested in your feedback as students. I want to know whether you like this book.

Programming Perl by Wall, Christiansen, and Schwartz is a much more detailed treatment of the Perl Language. Larry Wall is the primary creator of the language, so this book is very authoritative and much more complete. It is also quite well written. If your programming skills are more advanced, you may want to look at this book. I believe it is available in the BYUH bookstore.

I do give credit for reading in the books. To honestly get reading credit, you must (at a minimum) let your sight rest on each of the words in the assignment, let the word pass through your mind, and try to understand what is being said. If you can speed-read some or all of it with reasonable comprehension, that is acceptable too.

As you complete each assigned reading mentioned on the course calendar, notify me by submitting a program that tells me "I read chapter 1, Introduction, of Elements of Programming with Perl by Johnson." Or something like that. Email GradeBot (see below) for authoritative details. When you submit such a program, you are asserting that you have in fact done what the program says about you.

Labs: The key to a programming course is programming. (Duh.) You will complete a number of programs. Programs are graded by GradeBot (see below). Each must run perfectly before it will be accepted. Half of the points for your grade come from these labs.

Tests: There are six tests (maybe) given in the testing center using bubble sheets, including one that is comprehensive and is due the last day of class. You can complete the tests as soon as you want. I allow unlimited time and scratch paper, but no books, no notes, and no calculators. For each test, a sample test is available through GradeBot for you to use as a study guide. You only get one chance to take each test. (If you feel there

is some special reason you should get another chance, such as illness, discuss it with me.)

There are three tests given in class. These are programming tests where you will be asked to write some particular program without the aid of notes of a computer: written by hand, graded by hand. Time is limited. Scratch paper is provided. No books, no notes, and no calculators. No sample tests are provided.

6 Lectures

You do your part by reading, attempting the labs, deciding what questions to ask in class, and bravely asking them. I prepare the overall calendar, the syllabus, the list of assignments, the GradeBot routines to grade them, and the schedule of readings. I also prepare myself to answer whatever questions you can think of.

I follow a general "got questions?" teaching philosophy. It leaves the responsibility for learning with the people that are supposed to learn: the students. (I cannot learn for you.) Canned lectures can be fun and exciting, but frequently the relevant material is in the assigned reading. Our class time will be focused on things you need to do the nearby assignments, or on explaining things that were not sufficiently clear from the reading.

Attendance: I take roll in this class. Attendance counts for 10% of your final grade. Typically attendance is worth 3 points per day. I take 3-point roll at the start of class. I take 2-point roll about 10 minutes into class. If you come later than that, you can get one point by making sure I notice you in class (maybe right after class). Missing and unnoticed persons get zeros.

Due to INS (immigration) and VA (veterans) requirements the Vice President for Student Life is supposed to be notified whenever a student misses four consecutive class days. I try to do this.

7 GradeBot

GradeBot is my robotic program grader. It is generally available 24 hours a day, seven days a week, to grade and return your lab assignments. This is currently done via email.

I enable you to have a computer account on the (cs.byuh.edu) computer network. This account gives you access to a UNIX system, software (including compilers and assemblers), email, and some storage (type "quota" to see how much). You can also put up your own web page and CGI scripts. Most of you will use this account to do all the lab work in this class. See me if you need any help getting set up.

For grading, GradeBot is correct and authoritative. It is your boss. It is your client. It is your Drill Sergeant. There is always a particular correct behavior that it demands. You must make your program behave in exactly the way that GradeBot is requiring (including spelling errors, if any). Be sure to look at a sample "conversation" with GradeBot before you start writing your program.

To submit a program to GradeBot, (1) send it by WebBot, or (2) use the CS "turnin" command, or (3) send it by email to `<gradebot@gradebot.byuh.edu>`. You can do this from almost anywhere on the Internet. The basic subject line for this class is "Subject: cs201". With "Subject: cs201 status" you get a *status* report telling you everything you have completed, everything that is still due (and when), and what grade you have earned or are likely to get. To submit an assignment "xxx" to GradeBot, the subject line is "Subject: cs201 xxx". If you are having problems with extra stuff appearing after your program (such as an advertisement for juno or hotmail), you can put a "BEGIN" line before your program and an "END" line after it. Currently GradeBot does not understand attachments; your program must be in the body of your message. Do not use any special encoding, such as HTML or MIME or Rich Text.

If you discover a case where you believe that GradeBot is wrong, tell me about it. If you found an error in GradeBot, I generally reward you with some extra credit. Otherwise, you must assume GradeBot is right.

8 Lab Submission Rules

Cheating has never been a problem in this class, but there are rules. I am unhappy when I see cheating in any class. These may be cases where one student gives a copy of their completed program to another student, and the second student keys it in, possibly with minor changes, such as changing the names of variables. In worse cases, the second student uses cut-and-paste to copy the program, or sections of it. In almost every case, the second student *does not understand how the program works,* or why the program says what it says. I consider any such behavior to be plagiarism and an honor code violation. I want you to learn, but not do things that might let you complete the assignments without learning.

Open-Neighbor versus Copying: All labs are "open-neighbor" in the sense that you can *confer* with other people. You can read their code (if they let you). You can show your code to them. You can talk about your code, your approach, your difficulties, and your ideas. You can draw pictures and make analogies and ask questions. You can use their ideas. However, *you cannot make a copy of their code or submit their code to GradeBot, even if you first modify it.*

Never let another student take, borrow, or keep a copy of any program you wrote for this class. You can look at it *together*. If it is printed, please look at it away from any computers. If it is online, look at it on the author's own screen. Never bring up a window on the second student's screen so they can look at the first student's program. You can talk about what the program does, and why it is that way. Do NOT leave them with a copy of your program.

If you receive a copy of a program from someone, and use it as the basis for the program you are submitting, you are cheating.

9 Programming Labs

The purpose of lab work is to experience programming and grow thereby. Programming can be an extreme joy, where time ceases to exist (e.g., hours pass quickly but you don't notice). It can be a great pleasure to cause a machine to produce reports and process data at your will. Or it can be a nightmare, where nothing seems to work right, and the most insignificant things turn out to have far too much significance, and you pull out great clumps of your hair and hit you head against the wall and you are glad that not every IS professional needs to be an accomplished programmer. Labs reflect the true reality of a programmer's life. You should experience labs.

10 Course Calendar

Generally the lectures and discussion in class will follow the due dates for the various assignments (shown below). In-class tests will generally occur at the start of class.

11 Due Dates and Points

The names, dates, and points on this list are not guaranteed, but they are approximately correct. You should run a GradeBot status report to find the authoritative, correct due dates for you. The wording in this list is condensed to make it fit the space available.

```
  1: hello     thru Aug 30 (Thu)  10 pts
  2: aj1       thru Aug 31 (Fri)   4 pts
  3: aj2       thru Aug 31 (Fri)   4 pts
  4: aj3       thru Sep 04 (Tue)   6 pts
  5: a03       thru Sep 05 (Wed)   3 pts
... to be added (and corrected) ...
 84: comp      thru Dec 07 (Fri)  60 pts
 85: final     thru Dec 10 (Mon) 100 pts
```

Note x: I will be off-island for ISECON'2001, the Information Systems Educators Conference, in San Antonio, Texas. I am a member of the board of directors for EDSIG, the sponsoring organization, and am also the Proceedings editor. I plan to fly out Tuesday, Oct 30 and return Sunday, Nov 4. The conference is Thursday through Sunday. **There will be no class on Wednesday. `qic2` will be given as scheduled on Friday, in class.**