9-2002

CS 131.01: Fundamentals of Computer Science

Michael O'Conner
University of Montana - Missoula, michael.oconner@umontana.edu

Registration Information:
Any student who takes CS 131 or CS 132 without having the corresponding prerequisite and/or co-requisite does so at his/her own risk. It is not the responsibility of the instructor to help such a student gain the knowledge that he/she should have obtained from prerequisite/co-requisite courses.

CS 131
72748 131_01 MWF CP 109 10:10 – 11:00
72749 131_02 MWF SS 344 4:10 – 5:00
Prerequisite: Computer programming experience in a language such as BASIC, Pascal, or C
Co-requisites: Math 100 or consent of instructor

CS 132
72750 132_01 MWF SS 362 1:10 – 2:00
Prerequisite: CS 131
Co-requisites: Math 121 or consent of instructor

Lab Information:
"hands-on" help sessions:
schedule to be announced in class: times depend on lab openings and TA/instructor availability

recommended computer labs on campus:
Fine Arts 210
Liberal Arts 242
University Center 225
Liberal Arts 206
(see handouts in the labs for schedules and policies regarding lab use)

Instructor/TA Information:
Instructor: Mike O'Conner
Office: Social Science 413
Office Hours: MWF 11:00 – 12:00
MWF 2:00 – 4:00
Phone: 243-2217 (with voice-message capabilities)
e-mail: o_conner@selway.umt.edu

Teaching Assistant: TBA
TA Office: TBA
TA Office Hours: TBA

Course Information:
Required Text:
Java Software Solutions, 3rd Edition, Lewis and Loftus, Addison Wesley, 2003

Helpful Additional Resources:
- to acquire a good working knowledge of Java in a short time (good for beginners):
murach's beginning Java 2, Andrea Steelman, Murach, 2001 (or a more recent edition)
- a good introduction to using Java (good for near-beginners):
The Java Tutorial, Mary Campione and Kathy Walrath, SunSoft Press (Prentice Hall)
or
java.sun.com/docs/books/tutorial/
- for someone who has programmed before
On To Java, Patrick Henry Winston, Addison Wesley, 1998 (2nd Edition)
or an online version at
www.ai.mit.edu/people/phw/OnToJava (3rd Edition)

- for a balanced understanding of Java and Object-Oriented Programming – NOT FOR BEGINNERS! –
  Thinking in Java, Bruce Eckel
  or the freely-downloadable version (using sftp) found on a CS Department Computer, in the directory
  /class/CS132-o_conner/ThinkingInJava.pdf (approximately 3.5 MB, I think)

Thinking in Java is well worth buying. Bruce Eckel has also written Thinking in C++, and other "Thinking in" books

Course Synopsis:

Fundamentals of Computer Science is intended to be an introduction to a diversity of topics in the field of Computer Science: a "breadth-first" overview. In CS 131, much of the "exploration" will be accomplished by learning elements of the Java programming language. CS 132 will finish the overview begun in CS 131, then present basic introductions to various major topics in computer science.

The Java language was chosen because:
- it can be used to illustrate most of the elements we wish to study in computer science
- it insulates the learner from many of the troublesome aspects found in other languages
- it incorporates the syntax used in C and C++ (two widely-used programming languages)
- it is freely downloadable
- applets - java programs that can be downloaded from the web and executed by a web browser - are popular

Our breadth-first exploration of computer science involves 4 views:
(1) One focus of computer science is software: the way humans control machines by means of commands expressed in a computer programming language. In learning and using such a language one also becomes familiar with paradigms, concepts, principles, models, patterns, conventions, tools, and skills employed to create software.

(2) Another focus of computer science is the machine itself, including topics such as:
- electronics and the "digital" concept(transistor, switch, gate, latch, flip-flop, ...)
- components (memory, ALU, ...)
- architecture (the integration of components to form a machine)
- networking (interconnecting machines for better functionality, economy, and efficiency)

(3) A third focus of computer science is the application of the computer as a tool. Topics discussed under this view include: recursion and dynamic programming, simulation, Data Base Management Systems, and various artificial intelligence topics (robotics, machine learning, genetic algorithms and other problem-solving methodologies, speech recognition, and others). We will write programs that illustrate the principles we wish to study in those topics.

(4) Because of the potential for the abuse of the knowledge, skills, and privileges gained through the study of computer science, we will also focus on “Computer Ethics”. By examining ethical issues that arise from the use of computers, we hope to raise each student's awareness of the issues, as well as to encourage ethically sound decisions throughout that person's computer science career.

Course Objectives:

Ten major objectives of the Fundamentals of Computer Science course are listed below. They are abilities the successful student will acquire.

1) become proficient in the creation of software, based on:
   problem analysis
   solution design
   program implementation in various paradigms
   program verification and modification
2) become adept in the use of basic commands and the file system of the UNIX (and Unix-like) operating systems
3) create interactive programs in both command line and graphical interfaces
4) understand the specification of a programming language, and the steps involved in translating a program into machine-executable form
5) manage data via static and dynamic structures
6) understand the use of recursion, dynamic programming, and simulation as problem-solving tools

7) implement the basic operations of a relational database
8) understand the basics of combinational and sequential logic circuits
9) employ simple artificial intelligence models, such as
searching a solution space,
utilizing machine learning, or
creating model simulations,
to complement problem-solving efforts
10) develop an appreciation of the importance of professional ethics in the field of Computer Science

Proposed Course Agenda - CS 131: Examine fundamental computer science concepts by writing programs that use the high-level structured programming language Java.

Software engineering is the art/science of controlling a computer (a symbol-manipulating machine) by means of commands expressed in a language devised for that purpose. In Fundamentals of Computer Science I, we consider two programming paradigms (both of which can be demonstrated using java), software engineering, graphical user interface, and ethics.

Lecture Plan:
Introduction, Syllabus, New Accounts
Ch 1 – Computer Systems: Hardware, Network, Software and Programming
Ch 2 – Objects and Primitive Data: Type, Assignment, Input, Output
Ch 3 – Program Statements: Sequence, Selection, Repetition
Ch 4 – Writing Classes: Methods (p223 to p233)
EBNF Specification of Language Syntax
Ethics #1
Ch 4 – Writing Classes: Encapsulation, Overloading
Ch 5 – Enhancing Classes: References, Wrappers, Interfaces, Events
Ch 6 – Arrays: Sorting, Multi-dimensional Arrays
Ch 7 – Inheritance: Overriding, Polymorphism
Ch 8 – Exceptions and I/O Streams: Keyboard and File Input/Output
Ethics #2
Ch 9 – Graphical User Interfaces: User-Application Interaction (via events and event handling)

To successfully complete CS 131, the student will:
- demonstrate a basic understanding of the "von Neumann" architecture, including the components that comprise a computer, and their roles in the fetch-decode-execute cycle
- acquire familiarity with the programming environment: basic operating system commands related to programming, the file system, text editors, compiler/interpreter, remote login, ftp
- develop facility in software engineering skills (application of the waterfall and spiral models of program lifecycle), including: problem analysis, program design, prototyping, incremental implementation, debugging and testing strategies
- demonstrate programming ability in a variety of forms: batch and interactive, menu-driven and event-driven, application and applet, command line interface and graphical user interface, (pseudo-)imperative paradigm and object-oriented paradigm
- begin formation of personal ethics through the resolution of ethical questions/dilemmas by applying "analogy" or an ethical norm

Proposed Course Agenda - CS 132: A survey of computer science topics including recursion, algorithms, basic data structures, operating systems, artificial intelligence, graphics, user interfaces, and social and ethical implications of computing.

In Fundamentals of Computer Science II, we consider ways to make our programming more efficient, more powerful, and more elegant. Topics include:
- understanding the compiler (SLR Parser)
- tools/techniques for managing data (data structures, searching and sorting algorithms, databases)
- problem-solving techniques (Recursion, Dynamic Programming, Artificial Intelligence topics)
- Logic Design and "Programming in Hardware"

Lecture Plan

Greetings and Syllabus, changes in lab environment, review of the final exam from CS 131
Ch 8 – Exceptions and I/O Streams: Handle exceptional conditions ("exceptions") if they occur, and examine techniques for managing input and output – with specific discussion of input and output involving files
Ch 9 – Graphical User Interfaces: a more complete discussion than what was presented in CS 131
Ch 10 - Software Engineering: A review of programming models and paradigms
Ch 11 - Recursion: The concept itself, and the use of the Activation Record Stack, call-trees, and some useful "patterns"
Ch 12 - Data Structures: Abstract Data Types, Dynamic Data Structures, and a discussion of (linear) dynamic data structures: List, Stack, Queue, and Doubly-Linked List
Searching and Sorting: The use of Lists and Trees, and Big "O" notation
SLR Parser: A discussion of compiler theory – especially the parsing phase – and the use of finite automata to represent a process (such as parsing)
Logical Design and Circuits
Relational Databases: the 5 Orthogonal Operators, and a comparison of the use of pointers in C and C++ vs. references in Java
Dynamic Programming: the CYK Parsing Algorithm
Artificial Intelligence Topics: Simple Genetic Algorithm, Machine-Learning, or some other topic of interest

To successfully complete CS 132, the student will:
- obtain .and demonstrate a rudimentary skill in using recursion as a programming/problem-solving tool
- acquire facility programming with static and dynamic data structures (array, table, list, stack, queue, deck, tree), including the design and "Big O" analysis of several searching and sorting algorithms
- implement the parsing phase of a compiler, based on both the SLR algorithm (which is used to introduce finite automata and finite state machines) and the CYK algorithm (which is used to illustrate dynamic programming)
- design optimized special-purpose combinational logic circuits using the Karnaugh Map technique, and examine sequential logic circuits based on latches and flip-flops
- implement the 5 fundamental operations of a Relational Database Management System (Union, Set Difference, Cartesian Product, Projection, and Selection) and understand how they interact to provide more complex operations (such as Join, Intersection, Theta-join, and Quotient)
- demonstrate the ability to relate the ACM or SWE Code of Ethics to issues arising in the workplace
- modify/complete code written by a third party
- work on a software project as part of a team

Student Evaluation:

CS131 and CS132 will each have:
2 Ethics Essays
5+ Programming Assignments
Probably 1 or 2 Non-Programming Assignments
midterm exam 1: at the end of week 5 (Fri 4 Oct)
midterm exam 2: at the end of week 10 (Fri 8 Nov)
final exam (CS131):

CS 131_01	8:00 – 10:00 AM	Mon 16 Dec
CS 131_02	1:10 – 3:10 PM	Mon 16 Dec

final exam (CS132):

CS 132_01	1:10 – 3:10 PM	Thu 19 Dec

Most students submit homework assignments on or before the due date, often making sacrifices to do so. As one recognition of their efforts: work not submitted on the due date may be submitted before the next class but will receive a 25% reduction in points. For example, a grade of 88 becomes a 66 if submitted late.
Students who enroll in the class after any due dates have passed will be given two weeks to catch up with the rest of the class.

The computer automatically places a time stamp on work submitted electronically. For other work, make sure the submission date is on the homework when I accept it.

To count as "submitted homework", your work must be a reasonable attempt at doing the assignment.
The lowest "submitted homework" grade will be replaced by the average of the other "submitted homework" grades if you submit at least 8 homework assignments (**not including Ethics essays**). Then an average of all homework grades (including zeros) will be calculated.

Attempt to do every assignment: a missing assignment receives a grade of zero, and is not eligible to be dropped.

Accommodations are routinely made for disability-related issues, medical reasons, participation in university-related activities (such as performing arts productions, athletics, computer-programming contests, and the like), and work-related or family-related conflicts. In some cases, supporting documentation may be requested. Students utilizing the services of DSS should notify the instructor of that early in the course, and mention any desired accommodations.

If you know you will miss an exam, notify the instructor beforehand, to make arrangements for taking the exam at some other time. (See the second item in the "Rules" section of the "Miscellaneous Notes" section below.)
If you miss an exam for a reason beyond your control, bring supporting documentation when you request an opportunity to take a "make-up" exam.

Exam scores will use proportional re-scaling: Divide every grade by the highest grade and multiply the result by 100, to set the highest grade for the exam at 100, and set the other grades to their percentage of the highest grade.

A student's grade for the course will be based on total points calculated, and a "90-80-70-60" grading scheme will be used to obtain a letter grade. Total points are calculated from average points for assignments, for ethics essays, and for exams – each weighted by a contribution factor: 60/99 for assignments, 12/99 for ethics essays, and 27/99 for exams.
For the purposes of a "pass" or "not pass" grade, a ratio of 67.5% or higher is considered passing.

Performance Assessment:
The instructor (and TA's, if appropriate) may examine a student's overall performance during the course and decide whether or not the grade – as produced by the grading scheme above – accurately reflects that student's performance. If it is determined that the grade is not a fair assessment, then an increase of up to 2 percentage points may be made.

The intent of this provision is to be fair to students whose total points are very close to the dividing line between grades. For example, if the 12 grades that fall in the range 80 to 90 lie between 80 and 83 – with the exception of two students whose grades lie between 88.5 and 89.5 – then this criteria makes it possible to consider the work of those students more closely, and award the higher grade if that appears to be justified.

Receiving a grade of Incomplete is subject to the following criteria:
(See the University of Montana Catalog for 2002-2003, page 21 – "Policy on Incompletes" – for university requirements.)
- you must find out the financial impact that an incomplete will have on you (the effect on financial aid, probation, ...), and inform the instructor
- special circumstances (beyond the student's control) are involved
- coursework was at an acceptable (passing) level before the special circumstances occurred
- a reasonable expectation exists that the student can complete the course work independently
- the CS Department paperwork (yellow form) has been completed, signed by student and teacher, and submitted to the CS Department office (or to the instructor) before the end of finals week (20 Dec).

Support Services:
- various handouts or downloads will be available during the semester for assignments, lecture topics, and exam reviews
- we hope to have a weekly help session for homework and reviews for exams (depending on student interest)
- several of the CS Department floor monitors (who make it possible to provide after-hours access to the department) may be upper division students who can answer questions you may have about CS 131 and CS 132. The schedule should be out by the second week of the semester.

Grading Criteria:

"I do not give grades: you earn them."

Ethics Essays:

55 content is relevant to the assignment topic
10 responds to specific questions from the assignment
10 has a heading (upper left of page):

name
Ethics #(1, 2, 3, or 4)
CS 131/132 – Spr 2002
Date

10 acceptable spelling and grammar
10 general appearance (margins, indentation, ...)
5 appropriate length

Each Ethics assignment will be in the directory (or a sub-directory of) /class/CS131-o_conner or /class/CS132-o_conner. The directory will contain the assignment description, plus any other files needed (or useful) for the assignment.

Programs:

5 basic information about the program is in comments at the top left of the text file:

```
// file name
// author's name
// CS 131 or CS 132
// date
// brief description of program, including special details regarding execution
```

40 student-written code is present, appropriate for the assignment, and represents a reasonable attempt toward doing the assignment
10 proper indentation of code
10 output values labeled appropriately
5 a prompting statement to the user is used for input values

70 – maximum grade if program doesn't compile
80 – maximum grade if it compiles but crashes during execution
90 – maximum grade if it runs to normal termination, but gives wrong answers or results
100 – maximum grade if it works as described in the assignment

Each programming assignment will be on a CS Department computer, in directory (or a sub-directory of) either

/class/CS131-o_conner

or

/class/CS132-o_conner

The directory will contain the assignment and possibly several exercises. The exercises are not handed in, but they deal with aspects of the assignment. The experience gained by doing the exercises will greatly reduce the effort needed to do the assignment.

When possible, test your program execution on a CS Department computer before submitting it for grading. (You won't be able to test applets or GUI programs on CS Department computers over the internet.)

Other Assignments:

Grading criteria will be specified in the assignment.

Important Dates:
(see the back of the "schedule of classes" booklet for a more thorough listing)

Thu	5	Sep	late registration fee begins
Mon	23	Sep	last day to add/drop by DIALBEAR or Cyberbear, and pay fees/receive full refund
Mon	14	Oct	last day to drop by signatures on drop/add form, or change sections or grading option
Mon	28	Oct	advising for Spring 2003 begins
Mon	4	Nov	begin Cyberbear registration for Spring 2003 (ends Tue 19 Nov)
Tue	5	Nov	Election Day Holiday – no classes
Tue	11	Nov	Veteran's Day Holiday – no classes
Wed	27	Nov	Thanksgiving Vacation (ends Mon 2 Dec)
Fri	6	Dec	last day to withdraw from Autumn 2002 semester
Fri	13	Dec	last day for drop/grade-option-change petitions
Mon	16	Dec	final exam week begins (ends Fri 20 Dec)

Miscellaneous Notes:

Preparation:
- You will get more from the lectures if you have looked over the material beforehand.

Attendance:
I assume students will attend all classes. If you miss a lecture, you alone are responsible for the course material you missed – including information not in the text – as well as any changes regarding assignments and exams. The instructor is not responsible for helping the student acquire material missed due to "cutting class".

Courtesy:
- During lectures, please don't distract other students by carrying on 'side conversations.
- Be polite to CS Department staff in SS 401.
- If you arrive or leave during class, please try to do so in a way that will minimize any disruptive effects on other students.
- Profanity, obscenity, racial slurs, and hate-speech have no place in a scholarly community, and will not be tolerated, neither in class nor in assignments.

Rules:
- When entering the CS Department (Social Sciences, floor 4N) after 5:00 PM, you must print your name, CS class, and destination room number in the security log book.
- You need to show your Student ID to office staff in order to take an exam at other than the normal time and place, if you are taking the exam under their supervision. You will be provided with the best accommodations available at that time, but we can't guarantee normal exam conditions: you may experience interruptions and/or noise or distractions. Also, there may be no one present who is able to clarify a question you may have about the exam.

Collaboration and Cheating:
- The CS Department encourages students to share ideas and discuss assignments and projects (collaborate with others).
-The CS Department expects the work you submit to reflect your efforts (be responsible for your performance).

So, "learn" by discussion, but "do" by yourself.

Presenting the work of someone else as if it were your own is "cheating" (or "plagiarism", if the work is taken from a published source). The current policy regarding cheating is to consider the person who provides the work and the person who accepts it to be equally accountable. The penalties for cheating are severe; don't get involved in it.

Note: leaving your work on a CIS lab computer creates a possibility for someone else to copy it, in which case you will find yourself in the position of being considered to be involved in cheating – and with no way to prove otherwise.

So, I recommend saving all work on the computer's hard-drive, then transferring it to a floppy or zip disk when finished – and then deleting whatever you left on the hard-drive.

Another View of Collaboration and Cheating:
- One of the purposes for seeking a university education is to receive a fair evaluation of your performance in scholarly endeavors. (Others often use those evaluations to make judgments regarding your suitability for employment or your opportunities for further education.) If you present someone else's work as your own, that evaluation is worthless.

- Another purpose of a university education is to enhance your ability to think clearly – abstractly, analytically, critically, productively, and so on – which requires an effort on your part. If you do not make that effort, you do not get the experience from which you can learn more efficient and effective modes of thought.

- Yet another benefit of a university education is the opportunity to interact with highly intelligent people and observe how they have organized the knowledge of their field – acquiring an organized body of knowledge through such an interaction will save you years of experience and discovery.

- However, if you involve yourself in cheating, you are depriving yourself of the benefits of being in a university community; you are subverting your potential for future successes in life and/or academia; and in general you are indicating you are not interested in doing what is done at a university.

If you are not interested in doing what university students do here, then there is no point in your remaining here.

Besides, you are paying for a university education: get your money's worth!

Late Drops and Incompletes:
Be aware that late drops and incompletes often have serious financial aid implications. Be sure you understand them before requesting a late drop or incomplete.

Information from CIS:

Lorrie DeYott of CIS asked me to convey the following comments to CS students who use the CIS Labs:

1) Sometimes a bit on a floppy disk is not set correctly for Windows NT, so students saving to disk should open the command prompt window (click the Start button, then select the command prompt menu choice), type "**A:**" (without the quotation marks) and press the "Enter" key to change to the floppy drive, then type "**chkdsk /f**" (without the quotation marks) and press the "Enter" key. (This especially applies to the "green" disks sold at the UC.) You only need to do this once for each floppy disk. (My understanding: NT has an image of the disk it expects to be in the floppy drive. If you switch disks in the middle of something, and try to save to a different disk, NT will write the image it has onto the (incorrect) disk, and make a mess of whatever is there. Setting the bit on the floppy to the proper value will prevent that.)

2) Wait for the green light above the disk drive to go off before removing a floppy disk from the drive.

3) It is generally faster (especially if files will be saved to more than one disk) to work on the hard drive, then copy the file(s) to disk(s) once, just before leaving the machine. It is a good idea to then delete those file(s) on the hard drive (to leave less clutter on the lab machines – **and to prevent someone else from copying your work without your knowledge** – which can make you susceptible to the penalties for cheating, even though you did not intentionally participate in it).