CS 231 **INTRODUCTION TO SYSTEMS PROGRAMMING AND COMPUTER ORGANIZATION** **FALL 2002**

Syllabus & Course Policies

INSTRUCTOR: Robert England EMAIL: englandr@rhodes.edu
OFFICE: OH 419; Hrs: 2pm-3pm M-F PHONE: 843-3725

TEXT: *Introduction to Computing Systems: From Bits & Gates to C and Beyond*, by Patt and Patel. It would also be a good idea to track down a decent C (not C++) language reference. Supplemental material will be distributed in class as needed.

COURSE DESCRIPTION:

This course takes a bottom-up view of the computer as computational machine, beginning with bits and bytes and moving up to higher-level languages. The process of computation is presented through a hierarchy of virtual machine abstractions, beginning with the hardware and moving upward through various levels of increasingly sophisticated software, culminating in an introduction to the C language on the UNIX operating system. We will examine the facilities available at each layer of abstraction, along with the mechanisms and software tools that lead to the realization of the hierarchy.

PREREQUISITES:

CS 142 and Discrete Math.

TOPICS: [not necessarily in this order]
[and perhaps a bit ambitious]

Introduction
- Brief history of computing machines
- The components of a computing system

Machine level representation of data
- Bits, bytes, and words
- Numeric data representation and number bases
- Signed and twos-complement representations
- Fundamental operations on bits
- Representation of nonnumeric data

Digital logic
- Logic gates
- Combinational logic circuits
- Memory and simple storage elements

TOPICS, cont.:

Assembly level machine organization
- Basic organization of the von Neumann machine
- Control unit
- Instruction fetch, decode, and execution
- Instruction sets and types
- Assembly/machine language programming
  - Variables, types, expressions, and assignment
  - Simple I/O
  - Conditional and iterative control structures
  - Functions and parameter passing
  - Structured decomposition
- Instruction formats

Input and output
- Simple text I/O and text files
- Binary files

Fundamental data structures for low-level programming
- Primitive types
- Arrays, records, strings and string processing
- Data representation in memory
- Static, stack, and heap allocation
  - Implementation of recursion
- Runtime storage management
- Pointers and references

Introduction to programming in C
- Mapping high-level constructs to low-level implementations
- Special C operators for low-level programming
- C I/O facilities

Introduction to processes and threads

Introduction to programming in a UNIX environment
- C/C++ development environment
- The UNIX command line and simple scripts

GRADING:

Program assignments and homework:	40%
2 in-class tests:	35%
Final exam:	20% [Friday, 13 Dec 2002, 1:00pm-3:30pm]
Attendance:	5% Free! (...sort of. See next page.)

ATTENDANCE:

Attendance will be checked each class lecture period. As a bonus to those who attend regularly, each student who misses no more than 2.0 class meetings will be allowed to omit selected questions on the final exam worth a total value of approximately 15 points out of 100. These 2.0 allowed absences are to cover emergencies and do not have to be explicitly excused --- no excuses for other absences will be accepted with regard to this bonus. Each tardy (arrival after the start of class) counts as 0.5 absence. After class, you should alert me to your presence if you were tardy; otherwise, you will probably be counted absent.

In addition, regular attendance is worth a Free 5% of your overall course grade. Students who miss more than 6.0 class meetings forfeit this Free 5%.

Students are responsible for all material assigned or covered in class. The instructor's own lecture notes will not be made available for copying or review, so be sure to get notes from a colleague in the class if you should (unavoidably) miss a class meeting.

TESTS:

Both of the regular tests and the exam will be closed book, closed notes. Typical test format is a list of multiple choice questions, one code writing problem, and one code trace problem, though there may be slight variations to this format.

PROGRAM ASSIGNMENTS:

Each program assignment will each be awarded a letter grade A through X:

A: (100 pts)
'A' programs are carefully designed, efficiently implemented, well documented, and produce clearly formatted, correct output.

A- : (94 pts)
This is an 'A' program with one or two of the minor (?) problems described for grade 'B'.

B: (88 pts)
A 'B' program typically could easily have been an 'A' program, but it may have minor/careless problems such as poor, inadequate, or incomplete documentation; several literal values where symbolic constants would have been appropriate; wrong file names (these will be specified per program assignment); incomplete hard copies; sloppy source code format; minor efficiency problems; minor (?) memory leaks; etc. (This is not an exhaustive list.) You would be wise to consider 'B' the default grade for a working program --- this might encourage you to review and polish your first working draft of an assignment to produce a more professional quality final version of your program.

C: (75 pts)
A 'C' program has more serious problems: incorrect output for important special cases (the "empty" case, the "maxed-out" case, etc.), failure to carefully follow design and implementation requirements spelled out in the assignment description, very poor or inefficient design or implementation, near complete absence of documentation, etc.

PROGRAM ASSIGNMENTS, cont.:

D: (60 pts)
A 'D' program compiles, links, and runs, but it produces clearly incorrect output for typical cases. Or, it may deviate greatly from the design or implementation requirements stated in the assignment description.

F: (35 pts)
Typically, an 'F' program produces no correct output, or it may not even compile. It may "look like a program" when printed as a hard copy, but there remains much work to be done for it to be a correct, working program. Still, as a last resort, an 'F' program is better than no program turned in at all.

X: (0 pts)
A grade of 'X' will be recorded for each program not turned in. Each 'X' has the additional side effect of lowering the final grade for the course by one letter.

A fully documented sample program that you can use as a model for source code format will be distributed with or before the first programming assignment. The first line of each program source code file submitted for credit must be a comment that states the name of the source code file. Each student is responsible for keeping a back-up copy on disk of all source code turned in for an assignment. Failure to do so could result in loss of credit for an assignment.

Programming assignments must be turned in on the due date to receive full credit. Programs will be accepted late, but with a penalty of one letter grade per day.

Recommendation: *keep this list* of policies handy as a partial checklist of requirements to review before turning in each completed assignment!

ACADEMIC INTEGRITY:

All programs and tests must be the student's *own* work. Copying all or part of an assignment, or downloading code from the Internet and submitting it as your own, or having someone else write code for your assignment, or having someone else debug your assignment, or *allowing* someone else to copy from you, or coding or debugging someone else's assignment --- these are all included in the definition of reportable Honor Code violations for this course. If you have any doubts about whether or not a program development practice on an assignment is acceptable, clear it with the instructor before proceeding.

The instructor reserves the right to alter this syllabus as necessary.