CS 141 **COMPUTER SCIENCE I: PROGRAMMING FUNDAMENTALS** **FALL 2004**

COMPUTER SCIENCE I: PROGRAMMING FUNDAMENTALS

Syllabus & Course Policies

INSTRUCTOR: Robert England
EMAIL: englandr@rhodes.edu
OFFICE: OH 419
PHONE: 843-3725

TEXT: *Computer Science Tapestry,* by Owen L. Astrachan, 2nd ed.
Supplemental material will be distributed in class as needed.

COURSE DESCRIPTION:

In this course, you will learn the fundamental concepts of procedural programming. Algorithmic design and problem solving strategies will be emphasized as well as basic syntax of the C++ programming language. Topics to be covered include data types, control structures, functions, arrays, files, and the mechanics of running, testing, and debugging programs. The course also includes an introduction to the historical and social context of computing and an overview of computer science as a discipline.

PREREQUISITES:

None. No programming or computer science experience is required. However, you should have sufficient high-school mathematics background to solve simple linear equations and to appreciate the use of mathematical notation and formalism.

TOPICS: [not necessarily in this order]

Fundamental programming constructs
- Syntax and 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

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

Fundamental data structures
- Primitive types
- Arrays
- Records
- Strings and string processing

Overview of operating systems
- The role and purpose of operating systems
- Simple file management

TOPICS, cont.:

Human-computer interaction
- Introduction to design issues

Software development methodology
- Fundamental design concepts and principles
- Structured design
- Testing and debugging strategies
- Test-case design
- Programming environments
- Testing and debugging tools

Social context of computing [as time permits]
- History of computing and computers
- Evolution of ideas and machines
- Social impact of computers and the Internet
- Professionalism, codes of ethics, and responsible conduct
- Copyrights, intellectual property, and software piracy

GRADING:

Major program assignments:	40%	
2 in-class tests:	35%	[Fri, 1 Oct; Mon, 1 Nov]
Final exam:	20%	[Fri, 10 Dec, 5:30pm]
Homework:	5%	

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 12 to 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 be careful to remember to sign the attendance sheet if you were tardy; otherwise, you will be counted absent.

If you can verify that you were unavoidably absent because of a school sponsored event, the absence will not count against your 2.0 allowed absences, but you will still be responsible for all material covered in class, of course --- be careful to get lecture notes from a colleague for the class meetings you miss. The instructor's own lecture notes will not be made available for copying or review.

Aside from the possible effect of the exam bonus on your grade for the course, experience has shown that there is an unmistakable relationship between class attendance and the degree of mastery of the material presented in this course. 'Nuff said.

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.

No makeup tests will be given unless arrangements are made in advance.

MAJOR PROGRAM ASSIGNMENTS:

All programs assigned in this course must be written to run on MicroSoft Visual C++ 6.0 (or later). 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. However, each student has an allowance of three late days for the semester to cover emergencies. When these free late days have been expended, programs will be accepted late, but with a penalty of one letter grade per day.

Each major program assignment will each be awarded a letter grade A through X according to the following general guidelines:

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); sloppy source code format; minor efficiency problems; 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.

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.

MAJOR PROGRAM ASSIGNMENTS, cont.:

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.

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

HOMEWORK:

Homework assignments will typically be assigned one class period and be due by midnight before the next class meeting. They will be very short programming assignments that focus on some new topic just covered in class. Each homework programming assignment submitted on time will be awarded a grade Pass (full credit) or Fail (half credit). No homework will be accepted late.

The shrewd student may observe that it is numerically possible to achieve a grade of ‘A’ in this course without submitting any homework assignments at all! However, a high grade resulting from such action (inaction?) is highly unlikely: each homework assignment will be specifically designed as a warm-up for the next major programming assignment. Homework assignments will allow you opportunities to practice new programming skills with little risk and little effort before you are required to use these skills in nontrivial ways in the larger programs. Also, the Honor Code does not apply to homework --- cheat at your leisure! (E.g., you may enlist a seemingly more enlightened classmate to help you complete your homework assignment, nag the student assistant into telling you how to do it, etc.) Though when carried to extreme such practice clearly defeats the purpose of the homework, it may help reduce stress and enhance learning to collaborate on homework. Note that even in team efforts, each student must individually submit a copy of a homework assignment to receive credit for it.

ACADEMIC INTEGRITY:

Unlike the homework, all major programs and tests must be the student's *own* work. Copying all or part of a major program 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 a major program assignment is acceptable, please clear it with the instructor before proceeding.

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