SAULT COLLEGE OF APPLIED ARTS & TECHNOLOGY
SAULT STE. MARIE, ONTARIO

COURSE OUTLINE

COURSE TITLE:	COMPUTER PROGRAMMING I		
CODE NO.:	CSD100	SEMESTER:	FALL 97
PROGRAM:	COMPUTER ENGINEERING TECHNOLOGY/COMPUTER PROGRAMMER		
AUTHOR:	DENNIS OCHOSKI		
DATE:	JUNE 1997	PREVIOUSLY DATED:	JUNE 1996

APPROVED: [illegible]
DEAN

May 22/97
DATE

COMPUTER PROGRAMMING I — CSD100

COURSE NAME — COURSE CODE

TOTAL CREDITS: 4

PREREQUISITE(S): NONE

I. COURSE DESCRIPTION: This course is intended to provide a firm foundation of computer programming skills needed in the computer studies area. It is the first of two courses that use the C programming language to develop the students computer programming and problem solving skills.

II. TOPICS TO BE COVERED:

1. Introduction to computer programming concepts.
2. Basic C program structure.
3. Input/output in C.
4. Making decisions in C.
5. Repetition in C.

III. LEARNING OUTCOMES AND ELEMENTS OF THE PERFORMANCE:

A. Learning Outcomes:

	Approx. % of Course Grade
1. Discuss and apply the concepts involved in the development of software to solve problems using the computer.	15%
2. Write C programs applying the concepts of structure, arithmetic, and assignment.	10%
3. Develop algorithms and write C programs to solve problems involving the standard computer operations of input and output.	15%
4. Develop algorithms and write C programs to solve problems involving the standard computer operations of decisions/conditions and selection.	25%
5. Develop algorithms and write C programs to solve problems involving the standard computer operations of looping and repetition.	35%
	100%

B. Learning Outcomes and Elements of the Performance:

Upon successful completion of this course the student will demonstrate the ability to:

1. Discuss and apply the concepts involved in the development of software to solve problems using the computer. (Unit 1 - Perry and lecture notes)

Elements of the performance:

- define the concept of a "computer program/software"
- differentiate between prewritten software and custom-designed software
- differentiate between high level languages and machine language
- describe the purpose of a compiler/interpreter
- describe the top-down process of developing a program
- apply the "golden rule" for writing computer programs
- describe the process of transforming a source program to an executable module
- differentiate between batch processing and online processing
- write algorithms and describe them using pseudocode and flowcharts

2. Write a simple C program applying the concepts of program structure, arithmetic, and assignment. (Units 2, 3, 5, 6, 7 and 9: pgs. 182-190)

Elements of the performance:

- explain the main components of a C program
- name and distinguish C's basic data types
- explain and properly use the naming conventions for C identifiers
- differentiate between character and numeric constants
- differentiate between character and numeric variables
- declare and initialize variables correctly
- explain computer memory concepts and how they relate to processing data
- use assignment operators (=, +=, -=, *=, /=, ++, --)
- use arithmetic operators and apply their precedence (+, -, *, /, %)
- evaluate integer and mixed-mode arithmetic correctly
- explain automatic promotion and apply typecasting to define data types
- use the *sizeof* operator to determine how much memory is needed to hold a value
- differentiate between syntax and logic errors
- write and compile a simple program in C incorporating the concepts above

3. Develop algorithms and write C programs to solve problems involving the standard computer operations of input and output.
(Unit 3 pg. 54, Unit 4, Unit 6 pgs. 120-123, Unit 15 pgs. 303-317)

Elements of the performance:

- apply the scanf function to perform input of data
- apply the printf function to perform output of data
- apply simple character-based functions to perform input/output of data (getchar(), putchar(), putc(), getch(), putch(), getche())
- apply simple string-based functions to perform input/output of data (gets(), puts())
- apply proper variable format codes to the input and output of data
- explain the purpose of "include" files for the scanf and printf functions
- write, test, and debug programs using the scanf and printf functions

4. Develop algorithms and write C programs to solve problems involving the standard computer operations of decisions/conditions and selection.
(Units 8 and 14)

Elements of the performance:

- describe the use of the relational and logical operators, and use them to write complex logical expressions (==, !=, <, <=, >, >=, !, &&, ||)
- describe the operation of the following C decision-making structures and use them in C programs:

  a. if...else
  b. nested ifs
  c. if...else if...else
  d. the switch statement

- write algorithms to solve problems containing decision-making structures, and describe them using pseudocode and flowcharts
- write, test, and debug programs containing selection structures

5. Develop algorithms and write C programs to solve problems involving the standard computer operations of looping and repetition.
   (Units 11, 12, and 13)

Elements of the performance:

- discuss the concept of repetition/looping in computer programs
- describe the operation of the following C repetition structures and use them in C programs:
  a. while
  b. do...while
  c. for
  d. nested loops
  e. break and continue statements

- describe and correct an "infinite loop" problem
- write algorithms to solve problems containing repetition structures, and describe them using pseudocode and flowcharts
- write, test, and debug programs containing repetition structures

IV. EVALUATION METHODS:

The mark for this course will be arrived at as follows:

Quizzes:	
outcome #1	10%
outcomes #2 & #3	20%
outcome #4	20%
outcome #5	25%
Assignments:	
outcome #1	5%
outcomes #2 & #3	5%
outcome #4	5%
outcome #5	10%
Total	100%

The grading scheme used will be as follows:

A+	90 - 100%	Outstanding achievement
A	80 - 89%	Excellent achievement
B	70 - 79%	Average achievement
C	55 - 69%	Satisfactory achievement
R	Repeat	
X	Incomplete.	A temporary grade limited to special circumstances have prevented the student from completing objectives by the end of the semester. An X grade reverts to an R grade if not upgraded within a specified time.

V. SPECIAL NOTES

1. In order to pass this course the student must obtain an overall **quiz** average of 55% or better, as well as, an overall **assignment** average of 55% or better. A student who is not present to write a particular quiz, and does not notify the instructor beforehand of their intended absence, may be subject to a zero grade on that quiz.

2. Assignments must be submitted by the due date according to the specifications of the instructor. Late assignments will normally be given a mark of zero. Late assignments will only be marked at the discretion of the instructor in cases where there were extenuating circumstances.

3. The instructor reserves the right to modify the assessment process to meet any changing needs of the class. Consultation with the class will be done prior to any changes.

4. The method of upgrading an incomplete grade is at the discretion of the instructor, and may consist of such things as make-up work, rewriting tests, and comprehensive examinations.

5. Students with special needs (eg. physical limitations, visual impairments, hearing impairments, learning disabilities) are encouraged to discuss required accommodations confidentially with the instructor.

6. Your instructor reserves the right to modify the course as he/she deems necessary to meet the needs of students.

VI. PRIOR LEARNING ASSESSMENT:

Students who wish to apply for advanced credit in the course should consult the instructor.

VII. REQUIRED STUDENT RESOURCES

Text:	Programming C in 12 Easy Lessons by Greg Perry
Diskettes:	minimum of 3, 3 1/2"