Douglas College   Course Information

A: *Division*	Instructional	*Date*	June 18, 1997
B: *Department*	Pure and Applied Science and Technology	*New Course*	
		Revision of Course	X
		Dated	Jan 7, 1991

C: CMPT-110 *Course Number*   D: Introduction to Computing Science Using C++ *Descriptive Title*   E: 4 *Credits*

F: *Calendar Description*
This course introduces the science of computing. Emphasis is placed on the analysis of problems, the design of algorithms, and the abstraction of control and data in computer implementations of the design. Initially structured top-down design and procedural programming is used followed by an introduction to recursive functional programming then an introduction to Object Oriented Design (OOD) and Object Oriented Programming (OOP). C++ is used as the implementation language.

Summary of Revisions

Sections revised:
K

G: *Type of Instruction*

Lecture	2 x 2 hrs/week
Lab.	1 x 2 hrs/biweekly
Seminar	
Clinical Experience	
Practicum	
Shop	
Studio	
Student Directed Learning	5 hrs/week (approx.)
Other	
Total	10 hrs/week

H: Course Prerequisites:
CMPT-100 and
MATH-110

I: *Course Corequisites*
None

J: *Course for which this course is a prerequisite*
CMPT-210, CMPT-220

K: *Maximum Class Size*
34

L: *College Credit*
Transfer X

RECEIVED
JUL - 3 1997
Registrar's Office - N.W.
By: ....

M: *Transfer Credit*
Requested ____
Granted X

Course Equivalents

U.B.C.	CPSC(3)
S.F.U.	CMPT101
U. of Vic.	CSC110
Others in transfer guide	

[illegible]
Course Designer

[illegible]
Vice-President - Instruction

Desmond Wilson
Dean

P. H. [illegible]
Registrar

N: *Textbook and Materials to be Purchased by Students*

- Dale N., Weems C., Headington M., Programming and Problem Solving with C++, D. C. Heath and Company
- Portfolio for Programming Assignments
- Two 31/2" high density diskettes

O: *Course Objectives*

The student should be able to:
- analyze problem specifications
- form data and control abstractions
- design computer algorithms using either a structured top down design methodology or Object Oriented Design
- implement, in a widely acceptable style, algorithms in C++ using standard programming methodologies
- document a project

The student should understand the concepts of:
- generality through abstractions
- maintainability, reusability and extensibility through modularity

P: *Course Content*

1 Introduction and Review (syntax of C++)
  - 1.1 Program Structure
  - 1.2 Primitive data types and expressions
  - 1.3 Control Structures
  - 1.4 Functions and parameter passing
  - 1.5 Arrays
  - 1.6 Top-down design review and specs. for assignment #1: procedural programming with emphasis on control structures, procedures, and arrays

2 Abstractions (implementations are not considered)
  - 2.1 Strings
  - 2.2 Collections
    - 2.2.1 Lists
    - 2.2.2 Sets
    - 2.2.3 Stacks

3 Implementing Abstractions
  - 3.1 C++ Strings
  - 3.2 Introduction to pointers (domain of arrays and parameter passing)
  - 3.3 C++ records (struct)
  - 3.4 Structured design issues and specs. for assignment #2: procedural programming with emphasis on: cohesion and coupling and using more complicated static data structures
  - 3.5 Design of set primitives
  - 3.6 Recursion
    - 3.6.1 Numerical examples: factorial, Fibonacci, ...
    - 3.6.2 Examples from symbolic (LISP-like) expressions (SExpressions)
  - 3.7 Discussion and specs for assignment #3: functional programming using an existing module for SExpressions with emphasis on recursion and list processing

4 Encapsulation, Instantiation, and OOP
  - 4.1 Structure (syntax and semantics)
  - 4.2 Examples
    - 4.2.1 Sets implementation and use
    - 4.2.2 Stacks implementation and use
  - 4.3 Specs for assignment #4: OOP

5 OOD and Separate Compilations
  - 5.1 OOD
  - 5.2 Examples
  - 5.3 Specs for assignment #5: OOP
  - 5.4 Introduction to inheritance
    - 5.4.1 Examples

Q: *Method of Instruction*

There are three components to the course: lectures, labs., and assignments.

The lecture is used to introduce new material; usually via a sequence of theoretical concepts, practical considerations (usually language dependent), and one or more example case studies. The book is to be used as an additional source of problems and examples.

The two hour biweekly lab. is exclusively used to evaluate the student's practical programming ability. They are marked mostly on results; i.e. correctness of the algorithm.

Assignments are the most important learning vehicle and are done on the student's own time. They are marked according to program design, correctness and efficiency of the algorithms, coding style, and completeness of the documentation.

R: *Evaluation*

The final grade will be calculated from a particular distribution from the range below. The exact distribution will be given to the student on the first day of classes along with the course outline and necessary policies.

Distribution Range:

6 labs.	=	15% - 25%
2 tests @ 15% - 20% each	=	30% - 40%
1 exam	=	20% - 30%
5 assignments	=	20% - 35%

Example Distribution:

6 labs.	=	15%
test #1	=	15%
test #2	=	20%
assignments	=	25%
exam	=	25%
Total	=	100%