Douglas College Course Information

A: *Division* Instructional

B: *Department* Pure and Applied Science and Technology

Date June 18, 1997

New Course

Revision of Course X

Dated June 12, 1992

C: CMPT-210
Course Number

D: Data and Control Structures
Descriptive Title

E: 4
Credits

F: *Calendar Description*

This course continues the study of Object Oriented Design (OOD) and Programming (OOP) with a study of inheritance and polymorphism. Other topics include an introduction to the analysis of algorithms, techniques for searching state spaces, and dynamic data structures including lists, stacks, queues, and trees. Programs are written in C++.

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-110

I: *Course Corequisites*
None

J: *Course for which this course is a prerequisite*
None

K: *Maximum Class Size*
34

L: *College Credit*
Transfer X

M: *Transfer Credit*
Requested
Granted X

Course Equivalents

U.B.C.	CPSC216
S.F.U.	CMPT201
U. of Vic.	CSC115
Others in transfer guide	

Course Designer

Vice-President - Instruction

Desmond Wilson
Dean

Registrar

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

- Headington M., Riley D., Data Abstraction and Structures Using 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 the time complexity of iterative algorithms such as searching and sorting
- use OOD on problems where inheritance is advantageous
- take advantage of polymorphism
- choose the most appropriate abstract data structure and be able to implement it efficiently

The student should understand the concepts of:

- asymptotic behaviour of algorithms
- inheritance
- dynamic versus static data structures
- late/dynamic binding and polymorphism
- the need for techniques when searching an exponential space

P: *Course Content*

1 OOD and OOP

1.1 Specs. for assignment #1: consists of one program encompassing most topics from the prerequisite course (does not include inheritance)

1.2 Modules, information hiding, and inheritance

1.3 Specs. for assignment #2: inheritance

2 Analysis of algorithms

2.1 Worst and average case complexity of sequential and binary search

2.2 Worst case complexity of sorting algorithms: bubble, selection, linear insertion, binary insertion, mergesort, and quicksort

2.3 Film: Sorting out Sorting

2.4 State spaces, backtracking, and exponential growth

2.5 Specs for assignment #3: recursive backtracking in a puzzle state space

3 Dynamic Data Structures

3.1 Linear structures: lists, stacks, queues

3.2 Assignment #4

3.3 Trees

3.3.1 Binary trees

3.3.2 Recursive algorithms for traversals

3.3.3 Iterative algorithm for searching a tree: depth-first using a stack, breadth-first using a queue, and heuristic using a priority queue (implementation of the priority queue using a heap is considered later)

3.3.4 Binary Search Trees

3.3.5 Heaps

3.3.5.1 Heapsort

3.3.5.2 Priority queue

3.3.5.3 Assignment #5

3.3.6 Trie

3.3.7 Huffman codes

Q: *Method of Instruction*

There are three components to the course: lectures, labs., and self-directed learning (i.e. 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%