Data Structures and Algorithms

1. Aims and Objectives

- To distinguish between and be able to relate the high level (mathematical) world of data structures and the low level (engineering) world of storage structures.
- To develop a vocabulary for algebraic manipulation of data structures and a calculus of systematic refinement to algorithms and storage structures in the low level world of C and machines.
- To round off the foundations laid in IP and MF by engineering slightly bigger software on realistic computer systems.

2. Course Overview

	Algebraic View	Algorithmic View
Data	Data Structures Mathematical Definitions, laws, manipulations MF relations	Storage Structures Engineering Considerations Related to CO, LLP
Code	Recursive and closed form program specification May be implementable in a high level language like gofer or may not be implementable directly The intrinsic value of specification apart from programs	Explicit control through built-in control structures like sequencing, if, while Engineering efficient implementation of correct specifications

3. Course Contents

The course is organized according to the philosophy in the table below. The case studies/examples include but need not be limited to

Lists: Various types of representations.
Applications: symbol tables, polynomials, OS task queues etc

Trees: Search, Balanced, Red Black, Expression. Hash Tables
Applications: Parsers and Parser generators, interpreters, syntax extenders

Disciplines: Stack, queue etc and uses

Sorting and Searching: Specification and multiple refinements to alternative algorithms

Polymorpic structures: Implementations (links with PP course)

Complexity: Space-time complexity corresponds to element-reduction counts. Solving simple recurrences

4. Course Organization

	Algebraic world	Algorithmic world
Correctness	Bird Laws, Category Theory	Refinement, Predicates
Transformation	via Morgan Refinement	
ADTs and Views	— Formulation as recursive datatypes — Data structure invariants — Principles of interface design — Algebraic Laws	— C-storage: — Representation Invariants — Addressing Semantics — Use of struct, union and other assorted C stuff — Maximising abstraction by macros, enums etc
Mapping	via transforms and coupling invariants	
Code	— Pattern Matching based recursive definitions — Exhaustive set of disjoint patterns correspond to total functions — Correspond to runtime bug-free programs — Recursive Code structures follow from recursive data structures	— Refinement of recursive definitions into iterative algorithms — Techniques (Bentley) for improving algorithms eg sentinel, double pointers, loop condition reduction, strength reduction etc
Continuations	— Control as Data — Coroutines vs subroutines — General framework for escape procedures, error handling	— Loops — functions@ — Stack based software architecture
Error Policy	— Types — Patterns — Laws — Deliberate Partiality	Predicate Transformer Semantics for control
Modules	Category Theory	Files, make

5. Bibliography

1. Data Structures and Algorithms; Aho, Hopcroft and Ullman, Addison Wesley Inc.
2. Data Structures; Kruse; Prentice Hall
3. Programming from Specifications; Carroll Morgan; Prentice Hall
4. Algebra of Programs; Bird; Prentice Hall
5. Programming Perls, Writing Efficient Programs; John Bentley; Prentice Hall
6. Structure and Interpretation of Computer Programs; Abelson Sussmann; MIT Press
7. Functional Programming; Henderson; Prentice Hall
8. The Art of Programming Vol 1. & Vol 3; D. E. Knuth, Addison Wesley Inc.