

**GEORGIA PERIMETER COLLEGE
COMPUTER SCIENCE
TEACHING GUIDE -- CSCI 1302**

I. Principles of Computer Science II

II. Prerequisite: CSCI 1301 (Principles of Computer Science I) with a C or better.

III. Text: Data Structures Using Java, S. Malik, S. Nair, Thompson, ISBN: 0-619-15950-2

IV. Catalog Description:

The course continues the introduction of the fundamental principles of computer science from CSCI 1301. It extends algorithm development to large programs and introduces additional data structures, dynamic storage allocation, recursion, abstract data types, object-oriented design and programming, algorithm analysis, and file processing techniques, while continuing to emphasize structured programming techniques.

V. Course Objective:

This course continues the development of fundamental problem-solving, algorithm development, and programming skills in preparation for further study of computer science.

VI. General Notes:

The topics in this outline should be introduced as needed in the context of one or more projects involving building larger programs. The instructor may choose to begin with the statement of a sizeable project, then use structured programming techniques to develop a number of small subprojects covering the various topics of the course. The emphasis on good programming style, expression and documentation begun in CSCI 1301 should be continued. Analysis of algorithms should be introduced, but at this level such analysis should be given by the instructor to the student.

Consideration should be given to the implementation of programming projects by organizing students into programming teams. Team programming techniques are essential in advanced level programming courses and will be used in this course.

VII. Course Outline:

Week	Chapter Readings	Homework & Lab Assignments
1	Chap. 1 – Software Engineering and Algorithm Analysis	<ul style="list-style-type: none"> • Hmwk. p. 65 # 6-9 • Java Programming Review Assignment
2	Chap. 1 – User-defined Classes Chap. 2 -- Inheritance	<ul style="list-style-type: none"> • Hmwk. p. 64-70 # 1c,3,4,5,10,13,14,15,16,19,20, 22; • Ch.2 Hmwk p. 157 # 1(a,b,c,d), 4-14 • Programming Problem p. 72 # 4
3	Chap. 2 – Exception Handling	<ul style="list-style-type: none"> • Hmwk: p. 164 # 16 • Carefully study and understand the Programming Example: Grade Report on p. 133
4	Chap. 3 – Array Based Lists	<ul style="list-style-type: none"> • Programming Problem p. 166 ex. 6
5	Chap. 3 -- Vectors	<ul style="list-style-type: none"> • Hmwk. p. 217 Ex. #1-9; • Programing Problems p. 218 # 1-4
6	Chap. 4 -- Linked List Basics	<ul style="list-style-type: none"> • Hmwk. p.308 #1-13
7	Chap. 4 -- Linked Lists as an ADT and Variations	<ul style="list-style-type: none"> • Programming Problem p. 313 #1
8	Chap. 5 -- Recursion	<ul style="list-style-type: none"> • Hmwk. p. 355 # 1-10,12 • Implement hmwk. prob. # 12 on p. 357
9	Chap. 6 -- Stacks	<ul style="list-style-type: none"> • Hmwk. p. 427 # 1-7 • Programming Problem # 1 p. 430 (be sure to include the author's Stack exception classes)
10	Chap. 7 -- Queues	<ul style="list-style-type: none"> • Hmwk: p. 479# 1-11 (odd) • Programming Problem p. 483 #4
11	Chap. 8 -- Search Algorithms and Hashing	<ul style="list-style-type: none"> • Hmwk. p.517# 1,2,3 4,9,10,15,16 • Programming Prob. # 8 on p. 519-520
12	Chap. 9 -- Sorting	<ul style="list-style-type: none"> • Hmwk. p. 588 # 1-5
13	Chap. 9 -- Sorting	<ul style="list-style-type: none"> • Hmwk. p. 588 # 6-9
14	Chap. 10 -- Binary Trees	<ul style="list-style-type: none"> • Hmwk: p.589 # 9 & p. 665 # 1, 3-10, 12 • Programming Prob. # 1,2,3 on p. 669
15	Review & Wrap Up	

Topics to be Covered	Suggested Chapters
1. Review (5%)	
A. Problem Solving and Algorithm Development	Chap. 1
B. Programming Language Concepts*	
1. Data Types	Chap.1
2. Elementary Data Structures	Chap.1
3. Input - Output	Chap.1
4. Control Structures	Chap.1
5. User-defined Classes	Chap.1
2. Advanced Programming Techniques (20%)	
A. Implementation and Modification of Classes	Chap.1, 2
B. User-defined Classes*	
1. Data Members	Chap.1, 2
2. Methods	
a. Constructors	Chap.1, 2
b. Accessors	Chap.1, 2
c. Mutators	Chap.1, 2
d. Utilities	Chap.1, 2
3. Overloading	Chap.1, 2
C. Using an Existing Class Given Its Interface	Chap.1, 2
D. Implementing and Modifying a Class	Chap.1, 2
E. Inheritance and Polymorphism*	Chap. 2
3. Recursion	Chap. 5
4. Search Algorithms	
Advanced searching techniques and their Efficiency	
1. Searching	
a. Linear Search for First Occurrence	Chap 8.
1) Ordered List	Chap 8.
2) Unordered List	Chap 8.
b. Binary Search	Chap 8.
c. Hashing	
1) Hash Functions	
a. Prime-Number Division Remainder	Chap 8.
b. Mid-Square	Chap 8.
c. Fold-and-Add	Chap 8.
2) Techniques for Handling Collisions	
i. Open Addressing	
a. Linear Probing	Chap 8.
b. Random Probing	Chap 8.
c. Quadratic Probing	Chap 8.
ii. Chaining	Chap 8.
5. Sorting	
a. Selection	Chap. 9
b. Insertion	Chap. 9
c. Quicksort	Chap. 9
d. Merge Sort	Chap. 9
e. Heapsort	Chap. 9
6. Program Development	
1. Large Scale Program Strategies	
a. Decomposition	Chap. 1

	b.	Successive Refinement	Chap. 1
2.		Introduction to Program Verification	
	a.	Loop Invariants - Basic Definition	Chap.1,2
		Additional Testing, Debugging and Maintenance Techniques	
	i.	Black-box testing	Chap. 1
	ii.	White-box testing	Chap. 1
3.		Introduction to Software Engineering Concepts	Chap. 1
		Software Development Phases	Chap. 1
7.		Data Abstraction (35%)	
	A)	Implementing and Modifying a Class	Chap.1
	B)	Inheritance and Composition	Chap. 2
	C)	Abstract data types	
	1.	Static Data Structures	
		a) Array-based Lists	Chap. 3
		b) Stacks	Chap. 6
		c) Queues	Chap. 7
		a) Heaps	Chap. 9
		b) Hash Tables	Chap. 8
	2.	Dynamic Data Structures	
		a) Vectors	Chap. 3
		b) Linked Lists	Chap. 4
		c) Stacks	Chap. 6
		d) Queues	Chap. 7
		e) Binary Trees	Chap. 10
	D)	Object-oriented design and implementation	Chap.1,2
8.		File processing techniques* (30%)	Chap. 2
9.		Analysis of Algorithms (10%)	
	A.	Characteristics of analysis	Chap. 1
	B.	Big-O Notation	Chap. 1
	C.	Advanced Searching and Sorting and the Efficiency of Algorithms	Chap.8,9

* Introduced in CSCI 1301

VIII. VII. Evaluation Methods

Details of grade determination are left to the instructor with the approval of the Department Head. Exams, assignments, and a final exam prepared by individual instructors will be used to determine the course grade. The course grade must weigh examinations for at least 50% of the grade and programming assignments for not more than 50% of the grade. Five to seven student programming projects must be assigned. Testing must consist of at least two one-hour examinations and a comprehensive final examination. The final examination must be weighted at not less than 25% nor more than 35%.

IX. **Effective Date:** May, 2005

Approved Date: May, 2005