

# Algorithms and Data Structures

Spring 2005

Instructors: Joe Fuller and Venkat Gudivada

Office: 205B Gullickson Hall

Phone: 696-6204

Email: [fullerj@marshall.edu](mailto:fullerj@marshall.edu)

[Gudivada@marshall.edu](mailto:Gudivada@marshall.edu)

Webpage: <http://users.marshall.edu/~fullerj>

<b>Material for Midterm Exam 1 .....</b>	<b>2</b>
I. Software Engineering .....	2
II. Program Correctness and Efficiency.....	2
III. Efficiency of Algorithms .....	3
IV. Inheritance .....	3
<b>Midterm One.....</b>	<b>4</b>
<b>Material for Midterm Exam 2 .....</b>	<b>5</b>
I. Lists and Collections .....	5
II. Stacks and Queues.....	5
III. Recursion .....	6
<b>Midterm Two .....</b>	<b>6</b>
<b>Material for Midterm Exam Three.....</b>	<b>7</b>
I. Trees .....	7
II. Hash Tables.....	7
III. Sorting.....	8
IV. Self Balancing Trees.....	8
<b>Midterm Three .....</b>	<b>9</b>
<b>Final Exam .....</b>	<b>10</b>
<b>Final Grades.....</b>	<b>10</b>

## **Material for Midterm Exam 1**

### **I. Software Engineering**

**Reading:** The student should read the following sections of the text: 1.1, 1.2, 1.3, 1.4

**Objectives:** In this section, the student will learn to answer the following questions:

- a. What is software engineering?
- b. Why is software engineering needed?
- c. What is a software lifecycle?
- d. What are the steps in some software lifecycle models?
- e. What is procedural abstraction?
- f. What is data abstraction?
- g. What is a Java Interface?
- h. How are interfaces useful in Software engineering?
- i. What is the `implements` clause?

The student will be able to discuss the field of software with potential employers or other software professionals.

**Evaluations:** In order to evaluate the student's success in mastering the concepts of software engineering

1. Any or all of the following will be part of the first midterm exam:

- a. The student will be asked to write a complete definition of software engineering.
- b. The student will be asked to write a description of one or more software lifecycles
- c. The student will be asked to define procedural and data abstraction
- d. The student will be asked to explain the importance of data and procedural abstraction

1. The student will write Java program(s) that show they have mastered the use of an interface. The students will write their own Java interface(s)

### **II. Program Correctness and Efficiency**

**Reading:** The student should read the following sections of the text: 2.1 –2.7

**Objectives:** In this section, the student will learn to answer the following questions:

- a. What are bugs and defects?
- b. What is the difference between a run time error and a syntax error?
- c. Describe the Java Exception class
- d. What is meant by throw, catch, try, and finally?
- e. Why is testing a program so important?
- f. What are some of the commonly used methods of testing a program?
- g. What are stubs? What are drivers?
- h. Who should test a program? When should a program be tested? How should a program be tested?

**Evaluations:** In order to evaluate the student's success in mastering the topics listed above:

1. Any or all of the following will be part of the first midterm exam
  - a. Define bugs and defects.
  - b. Define run time errors and syntax errors and explain the difference.
  - c. Describe the use of try-catch-finally blocks.
  - d. Describe stubs and drivers. Describe how a software system should be tested.
2. The student will write Java programs that use try-catch-finally blocks.
3. The student will be asked to demonstrate that he/she can use debuggers in his/her Java IDE

### III. Efficiency of Algorithms

**Reading:** The student should read section 2.8 of the text

**Objectives:** In this section, the student will learn to:

- a. Define "Big -O" notation.
- b. The student will be able to define  $O(1)$ ,  $O(n)$ ,  $O(n^2)$ ,  $O(\log n)$ ,  $O(n \log n)$ , and  $O(n!)$
- c. The student will be able to determine the "big O" for selected algorithms
- d. The student will be able to explain why Big-O notation is important

**Evaluations:** In order to evaluate the student's success in mastering the material listed above:

- The student will demonstrate his/her mastery of the terms listed above on the first midterm exam
- The student will do lab exercises that demonstrate his/her understanding of the importance of Big O to software engineering

### IV. Inheritance

**Reading:** The student should read sections 3.1 –3.4, 3.6-3.8

**Objectives:** In this section the student will review the concepts of inheritance, derived classes, abstract classes, the cosmic class Object, and packages.

**Evaluations:** In order to evaluate the student's success in mastering the material in the above list

1. On midterm one, the student will provide definitions and discussion of the following terms:
  - a. The "is a" relationship
  - b. The "has" a relationship"
  - c. The Object class
  - d. Overriding and overloading methods
  - e. Casting and cloning
2. The student will write Java program(s) that demonstrate proficiency in using inheritance, overriding, overloading, casting and cloning.

## **Midterm One**

Midterm one will be given on the topics I, II, III, and IV. It will be scaled to be worth 100 points. It will count 20% of the final test average.

Programming and Lab assignments in this section will count 20% of the final programming average.

## **Material for Midterm Exam 2**

### **I. Lists and Collections**

**Reading:** The student should read sections 4.1, 4.3-4.8

**Objectives:** Students will learn

- a. The abstract list and abstract collection
- b. The List and Collection interfaces
- c. How to define and implement a linked list
- d. How to define and implement a doubly linked list
- e. The ArrayList class
- f. How to use the Java LinkedList class

**Evaluation:** In order to evaluate the student's success in mastering the material in the above list

1. On midterm two, students will be asked questions that demonstrate their knowledge of
  - a. Lists (ordered and unordered)
  - b. Collections
  - c. List and Collection interfaces
  - d. Linked and doubly linked lists
2. Students will write a Java program that implements a generic linked list
3. Students will be asked to demonstrate their proficiency with the Java LinkedList class

### **II. Stacks and Queues**

**Reading:** The student should read sections 5.1, 5.2, 6.1, and 6.2

**Objectives:** Students will learn

- a. The specifications for a queue and for a stack
- b. The students will learn how to use Java Library classes for queue and stack applications
- c. How to implement stacks and queues with a linked list class

**Evaluation:** In order to evaluate the student's success in mastering the concepts listed above

1. On midterm two, students will be asked to demonstrate their knowledge of
    - a. Queues – definition and specification
    - b. Stacks – definition and specification
    - c. Applications appropriate for queues
    - d. Applications appropriate for stacks
- Students will write at least one Java program to implement a stack or a queue
  - Students will be asked to demonstrate proficiency with the Java Library classes that implement stacks and queues

### III. Recursion

**Reading:** The student should read sections 7.1-7.3 carefully. Students should scan sections 7.4-7.6

Objectives: Students will learn the following:

- a. What is a recursive function in mathematics? What is a recursive function in Java (C, C++, etc)
- b. How to write recursive functions in Java
- c. When it is it appropriate to use recursion? When is it not appropriate?
- d. How to compare and contrast recursive methods with iterative methods
- e. Students will see examples of recursion including calculation of  $n!$ , array searching, and the Towers of Hanoi.

**Evaluation:** In order to evaluate the student's success in mastering the concepts listed above

1. On midterm 2, students will be asked to demonstrate their knowledge of
  - a. Recursive functions and recursive function definitions
  - b. Advantages and disadvantages of recursion
2. Students will be asked to write at least one Java program that uses recursion to solve a problem

### Midterm Two

Midterm two will be given on the topics I, II, and III above. It will be scaled to be worth 100 points. It will count 20% of the final test average.

Programming and Lab assignments in this section will count 40% of the final programming average.

## **Material for Midterm Exam Three**

### **I. Trees**

**Reading:** The student should read sections 8.1-8.4 carefully. Students should scan sections 8.5 and 8.6

**Objectives:** Students will learn the following:

- a. Basic definitions of trees and binary trees
- b. Tree traversal – in order, post order, and pre order
- c. How to specify a binary tree
- d. How to specify a binary search tree

**Evaluation:** In order to evaluate the student's success in mastering the concepts listed above

1. On midterm 3, students will be asked to demonstrate their knowledge of
  - a. Trees and associated definitions such as root, node, height, path, subtree, balanced tree, complete tree, etc
  - b. Methods of tree traversal
  - c. How to insert, delete, and retrieve nodes in a binary (search) tree
2. Students will be asked to write a Java program that implements a binary search tree. The program will be written so that the tree can store data from any class.

### **II. Hash Tables**

**Reading:** Students should read sections 9.1-9.3

**Objectives:** Students will learn the following

- a. The Java Set Interface and the Java Map Interface
- b. Definition of a Hash Table
- c. Issues associated with a hash table including collisions, chaining, probing, and hashing functions
- d. Students will be able to compare and contrast hash tables with linked lists, arrays, and binary search trees

**Evaluation:** In order to evaluate the student's success in mastering the concepts listed above

1. On midterm 3, students will be asked to demonstrate
  - a. Their knowledge of the definitions of Maps, Sets, and Hash tables
  - b. How to implement a hash table
  - c. Their knowledge of the characteristics of a good hash table
  - d. Their knowledge of collisions in a hash table and how to deal with them
2. Students will be given several lab and homework exercises to demonstrate their grasp of the concepts from this section.

### III. Sorting

**Reading:** Students should read sections 10.1-10.9.

**Objectives:** Students will learn to

- a. Describe and implement selection sort, bubble sort, insertion sort, Shell sort, Merge sort, and Quick Sort
- b. Students will learn the Big Oh() associated with each sort
- c. Students will use the sort method of the Java Arrays class
- d. Students will learn to compare and contrast any two of the sorts listed in item a

**Evaluation:** In order to evaluate the student's success in mastering the concepts listed above

1. On midterm 3, students will be asked to demonstrate
  - a. Their knowledge of each of the sorts covered in item a in the objectives. Students may be expected to use a "by hand" walk through for this purpose
  - b. Their ability to compare and contrast the sorts
  - c. Their knowledge of the efficiency of each sorting method
2. Students will be given several lab and homework exercises to demonstrate their grasp of the concepts from this section. At least one program will require the student to write a Java program to implement 2-5 of the sorts covered in the lectures.

### IV. Self Balancing Trees

**Reading:** The student should read sections 11.1, 11.2, and 11.3

**Objectives:** In this section the student will learn

- a. Why a balanced tree is desirable
- b. What it means to do a left rotation about a node in a tree and what it means to do a right rotation about a node
- c. What an AVL tree is
- d. How to specify an AVL tree
- e. What a Red-Black tree is
- f. How to specify a Red-Black tree

**Evaluation:** In order to evaluate the student's success in mastering the concepts listed above

1. On midterm 3, students will be asked to demonstrate
  - a. Their knowledge of why balanced trees are important
  - b. Their knowledge of an AVL tree
  - c. Their knowledge of algorithms to balance a tree using AVL algorithms
2. Students will be given several lab and homework exercises to demonstrate their grasp of the concepts from this section.

### **Midterm Three**

Midterm two will be given on the topics I, II, III, and IV above. It will be scaled to be worth 100 points. It will count 20% of the final test average.

Programming and Lab assignments in this section will count 40% of the final programming average.

### **Final Exam**

The final exam will be worth 40% of the final test average. It will be multiple choice and all questions will be taken from midterm exams I, II, and III.

### **Final Grades**

Final grades will be determined by weighting the test average by 65%, the lab-homework average by 35%. The 10% scale will be applied to the result.

**Example:** A student earned 60%, 70%, 90%, and 78% on the three midterms and the final. The student earned 90% of the lab points for exam 1, 80% of the lab points for exam 2, and 75% of the lab points for exam 3.

His test average is  $0.2 \cdot 60 + 0.2 \cdot 70 + 0.2 \cdot 90 + 0.4 \cdot 78 = 75.2$  His lab

average is  $0.2 \cdot 90 + 0.4 \cdot 80 + 0.4 \cdot 75 = 80$  His final grade would be

$0.65 \cdot 75.2 + 0.35 \cdot 80 = 76.88$  Using the 10% scale, the student would earn a

C.