Midterm Exam Study Guide

(If you know everything on this study guide, you should do very well on the midterm exam.)

Object-oriented programming:

We have studied the fundamental concepts of object-oriented programming, including topics such as:

- Abstraction
- Encapsulation
- Modularity
- Hierarchy
- Inheritance
- Polymorphism

You should understand these fundamental concepts, and be able to define them or recognize their use in a practical application.

C++ Constructs:

We have studied many C++ features, including the following topics:

- Classes (and objects), including data attributes and methods
- Memory management
- Constructor and destructor functions
- Inheritance
- Virtual methods
- "friend" and "this"
- Operator overloading
- Parameter passing
- Exception handling

You should understand how to implement and use these features in actual code, and how code using these features works.

STL Constructs:

We have studied several C++ Standard Library and Standard Template Library constructs, including:

- List
- Set
- Vector
- String
- Queue
- Priority Queue

You should understand how to implement and use these features (including the various functions defined for each of these container classes) in actual code, and how code using these features works.

General Data Structures:

We are studying many ***data structures*** this semester, which so far includes:

- Arrays
- Linked lists

- Red-Black trees (Know basic structure, properties, and runtime requirements of operations. You do not need to know how the operations are implemented).
- Stacks
- Queues
- Priority Queues (heaps)

In general, for each data structure, you should know the following:

- What an instance of the data structure looks like (e.g., how a heap is organized)
- What the rules/properties are for that data structure (e.g., the properties that describe a valid heap)
- The main operations that are applied to that data structure, as part of the interface (e.g., "insert" and "deleteMin" are the primary operations of a priority queue)
- A typical implementation of these operations.
- The runtime requirements of the operations applied to that data structure (e.g., "deleteMin" on a priority queue takes time O(log n))
- Given an example data structure (with actual data), show the result of applying a particular operation on that data (e.g., show the result of inserting a new element in a heap).

Algorithms:

We are studying many ***algorithms*** this semester. So far, this includes:

- Insertion sort
- Mergesort

In general, for each algorithm we study this semester, you should know the following:

- How the algorithm works.
- Given an example, show how the algorithm would process that example (e.g., show the result of applying MergeSort on a particular set of input numbers).
- Know the data structures that would be used to produce the most efficient implementation of the algorithm (for example, when a priority queue should be used versus when a red-black tree should be used)
- Know the runtime of the algorithm.

Algorithm Analysis:

- You should understand the basic theory of algorithmic analysis (O, o, Θ, ω, Ω).
- You should be able to rank functions according to their asymptotic growth rate.
- You should be able to write the recurrence corresponding to an arbitrary algorithm provided to you.
- You should be able to convert common recurrences to their equivalent asymptotic notation (e.g., $T(n) = T(n/2) + c \rightarrow O(\log n)$).
- You should understand how to use a recursion tree to help analyze the runtime of an algorithm.
- Given an algorithm (or pseudocode), you should be able to analyze the computational complexity of that algorithm (using asymptotic notation).
- You should be able to generate efficient algorithms to provide a desired functionality.

Simulations:

- You should know the general purpose of discrete event simulators and how they are structured at the high level.
- You should know generally how the probability distribution functions are used in discrete event simulators and the role of random number generators in this context.
- You should know how priority queues are relevant to discrete event simulators.

Sample Problems

Here are some practice problems to help you prepare for the midterm. These problems are representative of the general types of questions that you may see on the midterm. However, these problems do not cover the entire scope of topics you are expected to know; pages 1 and 2 outline the scope of material you are expected to know.

1. C++ Constructors, etc.

Behold the following C++ program (which continues onto the next page), which correctly compiles and executes:

```
#include <stdio.h>
#include <string>
using namespace std;

class myClass1 {
  public:
    myClass1();
    myClass1(string initialValue);
    myClass1(const myClass1 &rhs);
    myClass1 &operator= (const myClass1 &rhs);
    ~myClass1();
    string getString();
    void setString(string value);
  protected:
    string myData;
};

class myClass2 {
  protected:
    myClass1 t;
};

myClass1::myClass1() {
  myData = "Go Vols";
  printf("Create myData: %s\n", myData.c_str());
}

myClass1::myClass1(string initialValue) {
  myData = initialValue;
  printf("Init myData: %s\n", myData.c_str());
}

myClass1::myClass1(const myClass1 & rhs)  {
  myData = rhs.myData;
  printf("Assign myData: %s\n",myData.c_str());
}

myClass1 & myClass1::operator=(const myClass1 & rhs)  {
  myData = rhs.myData;
  printf("Operator= myData: %s\n",myData.c_str());
}

myClass1::~myClass1()  {
  printf("Destroying object: %s\n", myData.c_str());
}

string myClass1::getString() {
  printf("Getting string: %s\n", myData.c_str());
  return(myData);
}
```

```
      void myClass1::setString(string value) {
        myData = value;
        printf("Setting string: %s\n", myData.c_str());
      }


      main()
1     { string s;
2       myClass1 var1;
3       myClass1 var2("Beat Bama");
4       myClass1 var3 = var1;
5       myClass2 var4;
6       myClass1 *var5;

7       var1 = var2;
8       s = var1.getString();
9       printf("Var1 = %s\n", s.c_str());

10      var5 = new myClass1;
11      var5->setString("RockyTop");
12      s = var5->getString();
13      printf("Var5 = %s\n", s.c_str());

14      delete var5;
      }
15
```

Instructions: In the following table, give the output that occurs **upon completion** of the indicated line of code. If there is no output after a particular line, then leave the answer for that line blank.

1	
2	
3	
4	
5	
6	
7	
8	
9	

10	
11	
12	
13	
14	
15 (i.e., when `main` exits)	

2. Heaps #1

Suppose you are given the following array representing a binary min heap:

28 29 32 54 41 38 33 66 72 73 51

a. Show the array that results when you insert 14 into the heap.

b. Show the array that results when you delete 28 from the original heap.

3. Heaps #2

a. Show the final result of inserting 17, 19, 3, 21, 4, 6, 9, 22, 5, 10, 8, 7, 14, 15, 2, one at a time, into an initially empty binary heap.

b. Show the result of using the linear-time algorithm to build a binary heap using the same input.

4. Algorithm Analysis

What is the running time of the following program, expressed in Big-O notation? For full credit, you must give a tight upper bound.

```
sum = 0;
for (i = 1; i < n; i++)
  for(j = 1; j < i * i; j++)
  if (j % i == 0)
     for (k = 0; k < j; k++)
       sum++;
```

5. Object-oriented Design

Awesome! You've almost landed a sweet job offer from Google, complete with stock options. However, you have to pass the object-oriented design test before they will finalize your job offer. They want you to design a document organization tool for calculating the impact of printed articles. To start with, they have defined two types of documents – news articles and scholarly journal articles.

The fields for a news article are:

- Title: A string containing the title of the article (e.g., "UT Unveils Power T Cookie")
- Date: An integer representing the date the article appeared
- Impact factor: A floating point number representing the number of people who read this article

The fields for a scholarly journal article are:

- Title: A string containing the title of the article (e.g., "Distributed Algorithms for Multi-Robot Observation of Multiple Moving Targets").
- Date: An integer representing the date the article appeared
- Journal: A string representing the journal where the article appeared (e.g. "Autonomous Robots")
- Citations: An integer representing the number of citations of this article

You also need to create the following methods:

- string getTitle(): Returns the article title
- int getDate(): Returns the date the article appeared
- float getImpactFactor(): Returns the impact factor of this article. For a news article, the impact factor is just the number of people who read this article. For a scholarly journal, the impact factor is a function of the journal where the article appeared and the number of citations, evaluating to a floating point number. (You don't need to know exactly how it is calculated.)

Your job: Give the .h file that you would prepare for this application. (Room to write this .h file is given on the next page.) You must use inheritance to receive full credit.

Your answer should include:
1. Declarations for the newsArticle and scholarlyJournal classes.
2. Declarations for any additional classes you need in order to use inheritance.
3. Declarations for each of the variables and methods shown above. A crucial part of your answer is how you decide to distribute the variables and methods among the classes.

Your answer should not include:
1. Code to implement the methods
2. Declarations for constructors or destructors
3. Declarations for any methods not shown above

6. Miscellaneous True or False

a. True or False: If $f(n) = O(g(n))$, then $g(n) = o(f(n))$.

b. True or False: If $f(n) = \Theta(g(n))$, then $f(n) = O(g(n))$.

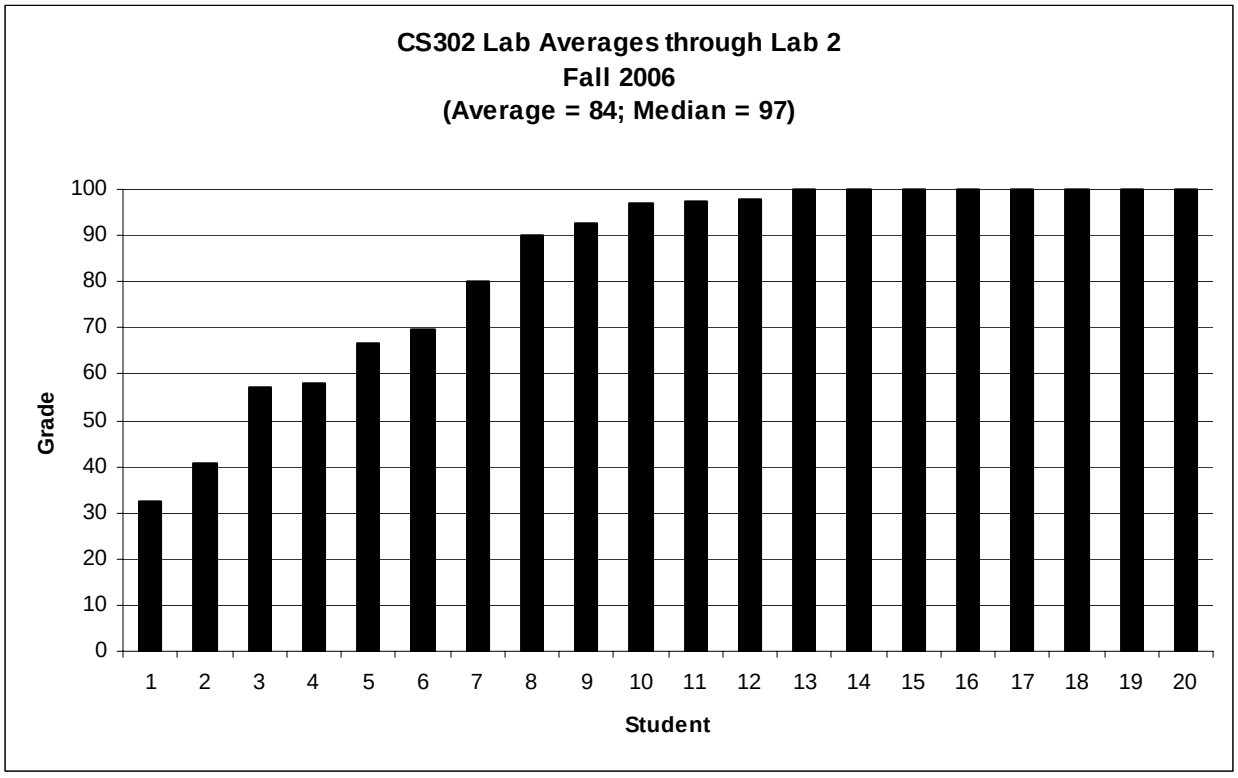
c. True or False: The following algorithm is $O(n^3)$:

```
total = 0;
for (i=1; i<n; i++)
   for (j=1; j<n; j++)
      total = total + a[i,j];
```

d. True or False: On average, using a red-black tree to perform insertions will be more efficient than using a heap for insertions.

e. True or False: C++ templates are evaluated when the code is executed.

f. True or False: In cases of class inheritance, data members that are private in the base class are private in the derived class.


CS302 Lab Averages through Lab 2
Fall 2006
(Average = 84; Median = 97)
Grade
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Student