CS200 Spring 2004
Final Exam
May 10, 2004

		Value	Points
Lab	1.	15	
Multiple Choice	2.	40	
Short Answer	3a.	7	
	3b.	18	
	3c.	15	
	3d.	15	
Coding	4a.	15	
	4b.	25	
	Total	150	

Name _____

StudentID# _____

1. **[15 pts] Short Answer from recitations:**

a) [3 pts] As presented in lab:

```
A3 extends Assoc
Assoc implements Comparable
```

Given the following code:

```
Object str = new A3(0.0, 1.3, "tmp");
System.out.println(str);
```

State the class order in which Java will look for the `toString()` method:

_____

b) [3 pts] What is the difference between an interface and an abstract class?

_____ _____

c) [3 pts] What command(s) would you use to extract the files from "lionelle.assign6.tgz", using verbose output?

_____

d) [3 pts] What would the command be if you wanted to list the contents of a directory in Linux,sorted by modification tie with the most current date listed last?

_____

e) [3 pts] You have a Java graph program called "GraphSolve" that takes a week to run. For input, it reads a graph description from standard input. For output, it prints to standard output. What would the command line look like if you want to run it as a batch job and then logoff? The graph description is in the file "graph_description.txt". (Hint: use redirects and don't forget to save the output.)

_____

2. **[40 pts] Multiple Choice.** Circle the letter of each statement that is true in each of the statements that follow the beginning statement. Note: there can be more than one or no true answers to each statement.

a) Binary search…

i) requires a binary search tree.

ii) requires O(c) time when the element to be found is in the first position.

iii) requires O(log n) time when the element is not there.

iv) is a divide and conquer algorithm.

b) Indicate which operations produce the listed results given the following heap,

13	26	19	49	46	21

i) Adding 1 changes it to: [1, 13, 26, 19, 49, 21]

ii) Doing a remove produces: [19, 26, 21, 49, 46]

iii) Removing 46 produces: [13, 26, 19, 49, , 21]

iv) Doing an add of 15 changes it to: [13, 26, 15, 49, 46, 21, 19]

c) A Binary Search Tree …

i) Adds all new values to a leaf.

ii) is balanced.

iii) Can be made more efficient for the average case only by a skew tree.

iv) Typically replaces the root with the rightmost left descendant when the root is removed.

d) Linear structures…

i) use time (order of adding and removing) to determine the contents.

ii) can be implemented using a priority queue with input order used to derive the key.

iii) are linked lists and Vectors.

iv) are used because they provide linear time access to data.

e) To sort an array of numbers…

i) A simple sort such as Insertion sort might be used if the numbers were already nearly sorted.

ii) Using heap sort, requires simply adding all of the numbers to a heap.

iii) That is too large to all be fit in memory at once is best done using merge sort.

iv) As quickly as possible, then use quicksort.

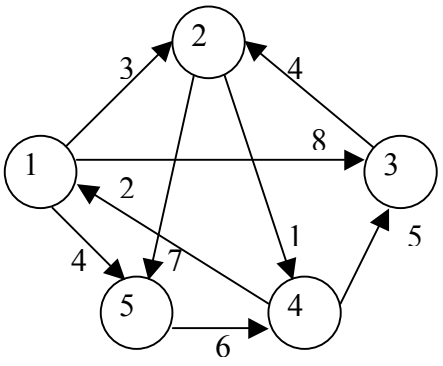
3. Short Answer

a) **[7 pts]** In your programming assignments, you implemented three different data structures for the spam and ok words. Which did you think was the best for this application? Give **two** reasons for your choice.

b) **[18 pts]** Show the first three *swaps* (not passes) that would be done if the following array were sorted using QuickSort. Assume the left position provides the pivot.

15	51	13	96	99	19

c) **[15 pts]** Given the following graph, compute **one** of the following by filling in the matrix or vertex for them. Costs have been put close to the arrow head on the edge. Use 999 to indicate no path. Also, give the Big-O value for the one you have chosen. For Dijkstra's SSSP, list the path to the node as well.



i) Floyd's All Pairs Shortest Path: **Big-O =**

ii) Dijkstra's Single Source Shortest Path starting from 5: **Big-O =**

d) **[15 pts]** Computing Transitive Closure on a graph.

i) Write down Warshall's algorithm (either the text version or the one in the code provided on the class web site) in pseudo-code or Java for computing transitive closure on a graph implemented using an adjacency matrix. For this purpose you can ignore the issue of vertex labels.

ii) Given the following adjacency matrix, fill in the empty matrix to show how it should look after Warshall's algorithm is run on it. 0 means no edge; 1 means an edge.

0	1	0	0	1
0	0	0	1	0
0	1	0	0	0
0	0	1	0	0
0	0	0	0	1

Name: **Student ID#:**

4. **Coding.**

   a) **[15 points]** Given the following skeletal class definition, write the remove method for an ordered doubly linked list. Remove should delete the linked list element containing an object that equals the parameter value. If you assume any helper methods or classes, write them as well.

```
public class DLLE{
     public DLLE(Object v, DLLE next, DLLE prev)
     public DLLE next()
     public DLLE previous()
     public Object value()
     public void setNext(DLLE next)
     public void setPrevious(DLLE prev)
}
public class DoublyLinkedList{
   protected int count;
   protected DLLE head;
   protected DLLE tail;
   public DoublyLinkedList() {
     head = tail = null;
     count = 0; }

   public Object remove(Object value){









}
```

   b) **[25 pts]** You must write two methods for an **adjacency list** implementation of a graph. The core data structure is a hash table (dict), where the label for the vertex is treated as the key and each

entry is an Association of key and value (which in a full implementation would be a linked list of edges). Use linear probing as the method for handling collisions in the hash table. The two methods are: `locate`, which finds the appropriate index in the hash table given a string label, and `add`, which adds a new vertex. You can assume you have access to an Association class as described below. Use the page back for more space.

```
public class Association {
     public Association(String label, LinkedList value)
     public Boolean equals(String other)
     public String getKey()
     public LinkedList getValue()
     public void setValue(LinkedList edges)  }

public class GraphList {
  protected int size;  // size of hash table
  protected Association dict[];  // hash table to translate labels->vertices
  protected boolean directed;  // graph is directed
  protected static Association reserved = new Association("reserved",null);
  protected GraphList(boolean dir) {
        size = 37;
        dict = new Association[size];
        directed = dir; }
  protected int locate(String label) {









    }
    public void add(String label) {
```