CS200 Final Exam
Fall 2007

Name _____

Topic	Possible	Received
Programming	53	
Complexity, Recurrence	18	
Divide & Conquer	16	
Priority Queues, Heaps	18	
Graphs	38	
Hash Tables	12	
Misc	45	
TOTAL	200	

1. Programming [53 points]

a. [30 points] Given the following interfaces for BinarySearchTree and KeyedItem (based on the textbook's code) and code for a HashTable, fill in the method stubs. Note: This hash table uses "Separate Chaining" where the auxiliary storage is a Binary Search Tree.

```
public class BinarySearchTree extends BinaryTreeBasis {
  // inherits isEmpty(), makeEmpty(), getRootItem(), and
  // the use of the constructors from BinaryTreeBasis

  public BinarySearchTree() { …}
  public BinarySearchTree(KeyedItem rootItem) { … }
  public void insert(KeyedItem newItem) { … }
  public KeyedItem retrieve(Comparable searchKey) {…}
  public void delete(Comparable searchKey) throws TreeException {… }
  public String toString() { … }
}

public class KeyedItem {
   private Comparable searchKey;
   private Object item;

   public KeyedItem(Comparable key, Object value) { … }
   public Comparable getKey() { … }
   public Object getValue() { … }
   public String toString() { … }
}

public class HashTable {
    public final int HASH_TABLE_SIZE=101;
    private BinarySearchTree[] table;

    public HashTable() {
        table = new BinarySearchTree[HASH_TABLE_SIZE];
    }

    public int hashIndex(Comparable key) {
        return Math.abs(key.hashCode() % HASH_TABLE_SIZE);
    }

    public String toString() {
        String output = "";
        for (int i=0; i< HASH_TABLE_SIZE; i++) {
            if (table[i] != null) {
                output = output + "\n" + i + ". " + table[i].toString(); }
        }
        return output;
    }
```

```
  public void tableInsert(KeyedItem data) {

  }
public boolean tableDelete(Comparable searchKey) {

  }
```

b. [10 points] The code in the last question did not use generics because a generified Array is not allowed in Java.

1. In addition to the hassle of my having to create non-generic versions of perfectly good classes, what hassle might programmers using my HashTable class incur?

2. What would need to be done to HashTable to make it generic?

c. [13 points] Match the lines in the Java code below (method is in Graph class) to lines in Dijkstra's algorithm (from the Rosen text). Note: the code uses the Graph implementation from Carrano and Prichard.

```
private int Dijkstras(Integer source, Integer dest) {
 1.  int[] distances = new int[adjList.size()];
 2.  Arrays.fill(distances, 1000000);
 3.  distances[source] = 0;
 4.  boolean[] visited = new boolean[adjList.size()];
 5.  Arrays.fill(visited, false);
 6.  int numVisited = 0;
 7.  while (numVisited < visited.length) {
 8.     int minDist = 100000;
 9.     int minIndex = -1;
10.     for (int i=0; i < distances.length; i++) {
11.        if ((!visited[i]) && (distances[i] < minDist)) {
12.           minIndex = i;
13.           minDist = distances[i];  } }
14.     if (minIndex > -1) {
15.        Set<Integer> edges = adjList.get(minIndex).keySet();
16.        Iterator<Integer> overEdges = edges.iterator();
17.        while (overEdges.hasNext()) {
18.           Integer neighbor = overEdges.next();
19.           Integer edgeWeight = adjList.get(minIndex).get(neighbor);
20.           int alt = distances[minIndex] + edgeWeight;
21.           if (alt < distances[neighbor]) {distances[neighbor] = alt;} }
22.        visited[minIndex] = true;
23.        numVisited++; }
24.     else {     numVisited = visited.length; } }
25.  return distances[dest]; }
```

Code Lines	Algorithm
0	Procedure Dijkstra(G: weighted connected simple graph, with all weights positive, z: vertex in G)
	for i := 1 to n { L(v_i) := ∞ }
	L(a) := 0
	S := ∅
	while z ∉ S {
	u := a vertex not in S with L(u) minimal
	S := S ∪ {u}
	for all vertices v not in S {
	if L(u) + w(u,v) < L(v) then L(v) := L(u) + w(u,v) }
	} return L(z)

2. Complexity/Recurrence [18 points]

a. [12 points] The complexity of MergeSort can be described by the recurrence relation $M(n) = 2M(n/2) + n$

1. What does the first number "2" correspond to in the algorithm?

2. Solve the recurrence relation for *M(8)*.

3. Which case for the Master Theorem applies for MergeSort: $a < b^d$, $a = b^d$ or $a > b^d$

b. [6 points] Describe how the number of comparisons in the worst case increases when these algorithms are used to search for an item in an array when the size of the array doubles from n to 2n, where n is a positive integer.

1. binary search

2. linear search

3. Divide & Conquer [16 points]

a. [6 points] Given a choice between MergeSort and QuickSort to implement for a given application, which would you pick? Justify your choice.

b. [10 points] For the array [1,3,5,7,9,2,4,6,8,10]

1. how many comparisons are needed for MergeSort to merge the two partitions?

2. how large are the two partitions created by Quicksort if it chooses the median of the values at *positions* {0, 4, 9} as the pivot.

4. Priority Queues, Heaps [18 points]

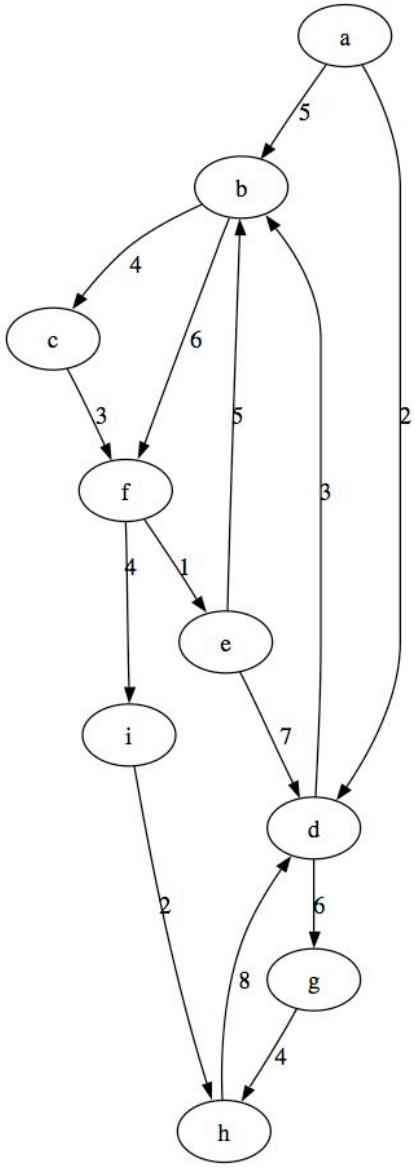
a. [12 points] Circle T or F for True or False for the following statements.

T	F	Mergesort is more efficient than HeapSort in the worst case.
T	F	An inorder traversal of a heap produces an ordered set.
T	F	In an array-based implementation of a heap, the HeapInsert operation is O(n) worst case.
T	F	When 40 is added to the heap [13, 100, 32, 101, 125, 160, 34, 102, 112, 130, 313, 175, 161, 200], the array becomes [13, 100, 32, 101, 125, 160, 34, 102, 112, 130, 313, 175, 161, 200, 40].
T	F	When a value is removed from the heap [13, 100, 32, 101, 125, 160, 34, 102, 112, 130, 313, 175, 161, 200], the array becomes [32, 100, 34, 101, 125, 161, 160, 102, 112, 130, 313, 175, 200].
T	F	A Binary Search Tree could be used to implement a Priority Queue.

b. [6 points] What makes a heap the best choice for implementing a priority queue?

5. Graphs [38 points]

a. [18 points] Answer the questions that follow for this graph. Whenever you have a choice in the algorithm being used, pick the vertex that is lower alphabetically.



1. Treating the graph as though it were undirected, draw a minimal spanning tree for it starting from vertex a.

2. Treating the graph as though it were undirected, draw a spanning tree that is *not* the minimal spanning tree for it starting from vertex a.

3. Treating the graph as directed, draw a topological sort for it.

b. **[20 points]** Fill in the blanks of the following definitions and theorems:

1. _____ in a graph G is a simple circuit containing every edge of G.

2. An undirected graph has an even number of vertices of _____

3. Let G=(V,e) be a graph with directed edges. Then $\sum_{v \in V} \deg^{-}(v) =$ _____

4. A simple graph is bipartite if and only if it is possible to assign

   _____

5. A directed path is _____ if there is a path between every two vertices in the underlying undirected graph.

6. Hash Tables [12 points]

Using the Java built in hash code as described in class and a hash table of size 7, the following words hash to the codes that follow them:

```
(bad:1)(bear:2)(echidna:3)(slo:3)(rat:4)(xhn:1)
```

Assuming they are added to a hashtable implemented using linear probing in the order: bear, echidna, slo, rat, bad, xhn, which pairs will experience primary clustering?

Which pairs will experience secondary clustering?

7. Misc [45 points]

a. [16 points] Stacks and Queues are the basis for graph and tree traversals. Each of the following traversals use one or the other:

inorder binary tree traversal **postorder binary tree traversal**
level order binary tree traversal
depth first graph traversal **breadth first graph traversal**

For each of Stack and Queue, pick a traversal that uses it and describe the algorithm. One traversal should be for a tree and the other for a graph.

STACK Traversal _____

QUEUE Traversal _____

b. [9 points] For the following grammar:

```
S->0S1 | 1S0 | 0 | 1
```

1. Describe the language produced by it.

2. Is 010101 a legal string for the language?

c. [20 points] Circle T or F for True or False for the following various statements:

T	F	An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.
T	F	There are at most h^m leaves in an m-ary tree of height h.
T	F	New items enter a queue at its front.
T	F	In the network routing simulation for your programming assignment, the event manager needed to maintain a queue of packets.
T	F	The keyword `extends` is used in the class definition of a subclass to indicate its superclass.
T	F	Stack can best be implemented with an ArrayList if the pushes and pops happen at position 0.
T	F	Adjacency list graph implementation is preferred for sparse arrays.
T	F	Hybrid data structures combine other different data structures to achieve their functionality.
T	F	An Euler circuit requires an odd number of edges.
T	F	Including a main method in every class expedites unit testing and debugging.