COMPUTER SCIENCE 112 - Spring 2012
FINAL EXAM

Name: ____

CIRCLE your recitation section: 01 02 03 12 13 14

- Be sure your exam has 7 questions. DO NOT TEAR OFF THE SCRATCH PAGES OR REMOVE THE STAPLE.
- Be sure to fill your name and circle your recitation time above, and your name in all subsequent pages where indicated.
- This is a CLOSED TEXT and CLOSED NOTES exam. You MAY NOT use calculators, cellphones, or any other electronic device during the exam.
- In all questions involving analysis, think carefully and write your answer clearly and precisely, including all the steps that are needed to get the answer. Credit will not be given if your answer comes without a proper sequence of steps with clear explanation.

Do not write below this line

Problem	Value	Score
1 Medley	10	_____
2 Sorting	20	_____
3 Graph Connected Components	15	_____
4 Dijkstra’s Algorithm	20	_____
5 Huffman Coding	20	_____
6 Heap Update Priority	20	_____
7 AVL Tree	20	_____
TOTAL	125	_____

1. **Medley (10 pts, 3+3+4)**

For each of the following questions, CIRCLE the correct answer out of the list of choices, AND justify your answer as asked.

a) A sequential search is performed on a sorted linked list as follows:

```
public static boolean search(Node front, int target) {
    while (front != null) {
        if (target == front.data) { return true; }
        if (target < front.data) { return false; }
        front = front.next;
    }
    return false;
}
```

What is the average number of target-to-linked list item comparisons for failure on a linked list with 10 items?

```
A. 10
B. 10.5
C. 11
D. None of the above
```

Explain in 3-4 lines.

b) Starting with an empty AVL tree, items are inserted one at a time. Which of the following sequence of insertions would result in rebalancing with rotation(s)?

```
A. 5, 3, 7, 2, 6, 8
B. 7, 5, 8, 6, 3, 2
C. 20, 15, 35, 45, 10, 25
D. 17, 13, 21, 19, 7, 25
```

Show the final AVL tree with balance factors for all nodes.

c) Suppose you are given two max heaps, one of size n and the other of size m, stored in arrays. What is the running time of the most efficient algorithm to merge the heaps into a single max heap of size $n + m$? Assume the heap arrays are directly accessible, and the resulting heap is also stored in an array.

A. $O((n + m)log(n + m))$
B. $O(nlogn + mlogm)$
C. $O(n + m)$
D. $O((n + m)^2)$

Describe your algorithm in 3-4 lines.

2. **Sorting (20 pts, 5+7+8)**

a) A stable sorting algorithm is one that preserves the order of duplicate key entries when sorted. In other words, if entries e_1 and e_2 have the same key k, and e_1 appears before e_2 in the input, then e_1 will appear before e_2 in the sorted output. Is quicksort stable or not? Prove your answer.

b) Suppose you use radix sort to sort countries according to ascending order of population. The country names and populations are read in as text, and the digits of the population are extracted using the `Character.digit(char ch, int radix)` method. What is the worst case number (exact number, NOT big O) of unit time operations that would be performed to sort counting only the following unit time operations: extracting a digit at a given position of a population number, inserting a (population,country) pair into a bucket's list during scatter, and appending the list of one bucket to the ”master” list during gather. (A bucket is an array slot.) Show the derivation of your result.

c) Continuing with the sorting problem of (b), you are now given some facts about world population: There are about 240 countries, China has the largest population, of around 1.3 billion, and Pitcairn Islands has the smallest population, of under 100. Also, 2 countries have population over 1 billion, and about 150 countries have populations in the millions, widely spread. Using these facts, describe the fastest (in real clock time, not big O) possible implementation you can think of (even in the worst case). Detail the data structure(s) and the process. You can use any of the sorting algorithms you have learned in class (except radix sort, which you have already analyzed in part b), or invent one!

3. **Graph Connected Components (15 pts)**

You are given an undirected, unweighted graph that may be disconnected i.e. some vertices may not be reachable from other vertices. Every group of mutually reachable vertices forms an island, called a *connected component*. There is no path between vertices in different connected components. If a graph is not disconnected, then its vertices are in a single connected component, which is the entire graph itself.

Implement a method using depth-first search that will number each vertex with the connected component that it belongs to. If there is a single connected component, then all vertices will be numbered 0 (zero). If there are two islands, then vertices in one island will all be numbered 0, and those in the other will all be numbered 1. And so forth.

Write your implementation in the `components` method in the following graph class. Assume that when that method is called, the `adjLists` structure is already populated. You can use helper methods as necessary– you must fully implement all helpers you use.

```
class Neighbor {
   int vnum;
   Neighbor next;
}
public class Graph { // undirected, unweighted
   Neighbor[] adjLists;
   ...
   // returns an array with connected component numbers for all vertices
   // i.e. returned[i] is the compnonent number for vertex i
   public int[] components() {
      /* COMPLETE THIS METHOD */
```

4. **Dijkstra's Algorithm (20 pts, 2+3+15)**

Dijkstra's single-source shortest paths algorithm is run on a directed graph with n vertices and e edges, with positive integer weights on the edges, and only stops when ALL vertices are done. (Assume all vertices are reachable from the source.) The graph is represented using adjacency linked lists, and the fringe vertices are stored in an AVL tree, with full access to the tree via a root pointer.

Answer the following questions to analyze the running time:

a) List all data structures, other than the fringe and adjacency linked lists, that are used so that at the end of the algorithm, the shortest distance and path from source to destination can be printed out. Ignore translation between vertex name and number.

b) Clearly identify and describe the *key steps* in the algorithm that are counted toward the running time.

c) Describe/derive clearly the worst case big *O* running time contributed by each of the key steps you identified in (c) above, over the entire run of the algorithm. Show the total big *O* time.

5. **Huffman Coding (20 pts, 13+7)**

Given the following set of character-probability pairs:

```
(A,0.2), (B,0.1), (C,0.3), (D,0.15), (E,0.05), (F,0.2)
```

(The probability associated with each character is the probability of the character occurring in the text to be compressed.)

(a) Build a Huffman tree for this character set. Fill in the following table to show the leaves queue (S) and the trees queue (T) at the end of each step. (Make sure to draw the actual trees in the Trees queue.)

```
Step         Leaves Queue (S)                       Trees Queue (T)
--------------------------------------------------------------------------------
  1                                                      Empty
```

b) Assume that enqueue, dequeue, peek (looking at the front of the queue without actually dequeuing), creating a leaf node, creating a new tree out of two subtrees, and picking the minimum of two probabilities all take unit time. Ignore the time for all other operations. How many units of time did it take in all to build the tree in this example? Show your work.

6. **Heap Update Priority (20 pts, 14+6)**

Suppose you are given the following heap class (assume that the constructor and all methods are already implemented, only public members are shown):

```
public class Heap <T extends Comparable<T>> {
   ...
   public Heap() { ... }
   public insert(T item) { ... }
   public T deleteMax() throws NoSuchElementException { ... }
   public boolean isEmpty() { ... }
   public int size() { ... }
   ...
}
```

Suppose an application (client) uses this heap class to store instances of the following class:

```
public class HeapItem implements Comparable<HeapItem> {
   public String id;    // unique
   public int priority;
   public HeapItem(String id, int priority) {
      this.id = id; this.priority = priority;
   }
   public int compareTo(HeapItem h) {
      return priority - h.priority;
   }
}
```

a) Implement a client method (i.e. in the application) that will update the priority of an item in a heap. Note: your code may ONLY use the given public constructor and methods of the `Heap` class. You may use, additionally, an `ArrayList` if you need, but no other data structure.

```
// update the priority of the item in the heap with the given id,
// to the new priority - throws exception if item is not found
public static void updatePriority(Heap<HeapItem> heap, String id, int newPriority)
throws NoSuchElementException {
```

b) Analyze the worst case big O running time of your implementation. Show your derivation. Count number of operations at the level of the method calls on the data structures, and multiply by the running times of those methods.

7. **AVL Tree (20 pts, 8+8+4)**

You have been asked to write a program to count the number of times each word occurs in a text file, and print them grouped according to word length. Below is a sample text file on the left, and the expected output of your program on the right:

```
This is the first line,
and this line is the ultimate line.
```

```
is: 2
the: 2
and: 1
this: 2
line: 3
first: 1
ultimate: 1
```

Note that words are counted ignoring case, and are printed in order of word length. In any group of words of the same length, the printout can be in any order (e.g. *the* and *and* are both printed before words of greater length, but they need not be in any specific relative sequence.)

<u>You choose to store the words in an AVL tree.</u> Each node in the AVL tree has a word (with its count), and words (alphabetical) are the basis of the AVL ordering.

a) If there are k <u>distinct</u> words, and n words in all (in the sample input file, $k = 7$ and $n = 12$), how much time will it take in the worst case (big O) to store all the words with counts? Show your derivation. Assume that all word-to-word comparisons take constant time, and ignore the time taken to read a word from input.

`198:112 Spring 2012 Final Exam; Name: ______________________________`

b) What would be the worst case time (big O) to print the output, if the maximum word length is m? Describe the process, and clearly state the basic operations (ignore the time to actually write to disk). Clearly describe any additional data structures you may use to get the job done. Show your work in summing up the basic operations toward the final big O answer. (Assume that determining the length of a word takes constant time.)

c) Would it be better or worse to use a hash table to count the number of times each word occurred in the input, and then print words and counts according to their lengths? Explain.

SCRATCH SPACE

198:112 Spring 2012 Final Exam; Name: ______________________________

SCRATCH SPACE

SCRATCH SPACE