COMPUTER SCIENCE 112 - FALL 2012
SECOND MIDTERM EXAM

Name: ______________________________

CIRCLE your recitation: W 10:35 W 12:15 W 1:55 Th 3:35 Th 5:15 Th 6:55

- Be sure your test has 4 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.

Do not write below this line

Problem	Max	Score
1 Binary Tree/AVL Tree	25	______
2 Huffman Coding	25	______
3 Hash Table	25	______
4 Data Stucture Design	25	______
TOTAL	100	______

1. Binary Tree/AVL Tree (25 pts,10+15)

a) Given the preorder and inorder traversals of a binary tree with n nodes, what would be the worst case big O running time to build the tree, counting only comparisons between items in the preorder and inorder traversals toward the running time? Clearly show your derivation with explanation of the steps, otherwise you will not get any credit.

b) After an AVL tree insertion, when walking back up toward the root, a node `x` is found to be unbalanced. Further, it is determined that `x's` balance factor is the same as that of the root, `q` of its taller subtree (Case 1). Complete the following `rotateCase1` method to perform the required rotation to rebalance the tree at node `x`.

```
public class AVLTreeNode<T extends Comparable<T>> {
   public T data;
   public AVLTreeNode<T> left, right;
   public char balanceFactor;   // '-' or '/' or '\'
   public AVLTreeNode<T> parent;
   public int height;
}

// returns the root of the updated tree
public static <T extends Comparable<T>>
AVLTreeNode<T> rotateCase1(AVLTreeNode<T> x, AVLTreeNode<T> root) {
      // COMPLETE THIS METHOD
```

2. Huffman Coding (25 pts, 17+8)

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.

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

b) Assume that enqueue, dequeue, peek, 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 your tree? Show your work.

3. Hash Table (25 pts, 8+11+6)

You are given the following classes:

```
public class LLNode {
   String key; String value; int hashCode; LLNode next;
   LLNode(String k, String v, int h, LLNode nxt) {
      key=k; value=v; hashCode=h; next=nxt;
   }
}

public class Hashtable {
   LLNode[] table;
   int numValues;
   float loadFactorThreshold;
   Hashtable(float loadFactorThreshold) {...}
}
```

a) Implement a method in the `Hashtable` class to insert a key-value pair into the hash table, using the function h **mod** N to map a hash code h to a table location. N is table length (capacity):

```
   // inserts (key,value) into hash table,
   // calls rehash method (part b) if load factor threshold is exceeded
   public void insert(String key, String value) {
```

b) Implement a `rehash` method (called from `insert` method), which doubles the table length when expanding it and rehashes the items in the hash table. <u>NO new nodes</u> should be created.

```
    private void rehash() {
```

c) Suppose you insert 125 integer keys into a hash table with an initial capacity of 25 and a load factor threshold of 2. The hash code is the key itself. Assume the function key **mod** table_length is used to map a key to a table position. How many total units of work will be done by the time all keys are inserted if it takes one unit of work to do the mapping, one unit to check load factor against threshold, and one unit to insert an entry into a linked list? Assume a rehash doubles the table capacity, and that the load factor is checked AFTER an entry is inserted into its linked list. Ignore the work done to create an array and/or to initialize an array to nulls. Show your work. (If you have several terms to add, leave the terms as they are–you don't need to simplify to a single number.)

4. Data Structure Design (25 pts, 15+10)

An array of length n contains integers in the range 1 through k. There is at least one copy of every single integer in this range. Here's an example, with $k = 5$ and $n = 10$:

```
  0     1     2     3     4     5     6     7     8     9
-------------------------------------------------------------
|  2  |  5  |  2  |  1  |  1  |  5  |  3  |  5  |  4  |  2  |
-------------------------------------------------------------
```

The objective is to **rewrite** A like this:

```
Array A transformed:

  0     1     2     3     4     5     6     7     8     9
-------------------------------------------------------------
|  3  |  4  |  0  |  2  |  9  |  6  |  8  |  1  |  5  |  7  |
-------------------------------------------------------------
<--- 1's ---><----- 2's -----><- 3-><- 4-><------ 5's ------>
```

The array A will now have **indexes** of all the 1's, followed by indexes of all the 2's, etc. Within each group, the indexes are in ascending order.

Another array, B, records the number of entries in each group. The first entry of B is unused, so the actual information starts from B[1] (B[1] is the number of 1's in A, B[2] is the number of 2's, etc.):

```
Array B:
             1     2     3     4     5
       ----------------------------------
       | X |  2  |  3  |  1  |  1  |  3  |
       ----------------------------------
```

a) Devise and describe the data structures and algorithm that will transform A as required, and build array B. Draw pictures of the data structure, and write your algorithm clearly as a sequence of steps - use pseudocode if necessary. (Essentially each step of the algorithm/pseudocode should be equivalent to a Java statement.)

b) Analyze your algorithm for the worst case: first decide on the operations you will count, and then determine the actual number of these operations, and then give the resulting big O time.

SCRATCH PAGE

SCRATCH PAGE

SCRATCH PAGE