COMPUTER SCIENCE 112 - FALL 2012
FIRST 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 Circular Linked List	15	______
2 Lists	15	______
3 Lazy Binary Search	15	______
4 BST	15	______
TOTAL	60	______

1. Circular Linked Lists (15 pts)

Implement the following method to delete the last occurrence (from the front) of an item from a circular linked list, given a pointer to its rear (last node).

```
public class CLLNode<T> {
    public T data; public CLLNode<T> next;
}

// Deletes LAST occurrence (from front) of given item from a CLL
// Returns reference to rear node of the updated CLL
// Throws NoSuchElementException if item is not in the CLL
public static <T> CLLNode<T>
deleteLastOccurrence(CLLNode<T> rear, T item)
throws NoSuchElementException {
   // COMPLETE THIS METHOD
```

2. Lists (15 pts, 3+3+5+4)

For each of (a) and (b) below, assume that the lists are stored in arrays, and compute the big O running times *using the fastest possible algorithm for each*. Count only item-to-item comparisons towards the running time. Briefly and clearly describe your your algorithm, and your reasoning for the big O time. (No explanation = no credit.)

a) Worst case time to find the common elements in two *unsorted* lists, one of length n and the other of length m.

b) Worst case time to find the common elements in two lists, one *unsorted* of length n and the other *sorted* of length m. Assume that $n < m$.

(c) and (d):

The following method merges two **sorted** linked lists of integers, building a new **sorted** linked list that contains all the items of both linked lists (including copies):

```
Node merge(Node L1, Node L2) {
  if (L1 == null) { return L2; }
  if (L2 == null) { return L1; }
  if (L1.data < L2.data) {
     Node temp = new Node(L1.data, null); temp.next = intersect(L1.next, L2); return temp;
  } else {
     Node temp = new Node(L2.data, null); temp.next = intersect(L1, L2.next); return temp;
  }
}
```

Assume the L1 list has n nodes and the L2 list has m nodes, and the nodes contain integer data.

d) What is the **worst case** *number* (not big O) of int-to-int comparisons? Explain.

e) What is the **best case** *number* (not big O) of int-to-int comparisons? Explain.

3. Lazy Binary Search (15 pts, 12+3)

The following is a *lazy* version of binary search on an array, A, of integers:

```
boolean lazySearch(int[] A, int target) {
   int lo=0, hi=A.length-1;
   while (lo < hi) {
      int mid = (lo+hi)/2;
      if (target > A[mid]) {
         lo = mid + 1;
      } else {
         hi =  mid;
      }
    }
   return target == A[lo];
}
```

Suppose this lazy search is used on the following array of integers:

```
15   25   29  65  75  76  92
```

a) Draw the comparison tree that will illustrate all possible search paths through the tree. Include failure nodes. Use the array items themselves (instead of array indexes) for the nodes of the tree. Be sure to mark target-to-array item comparisons on the tree so the comparisons can be counted for all successes and failures.

b) What is the average number of comparisons for successful search, assuming all matches in the array are equally likely? Show your work.

4. BST (15 pts, 6+4+5)

a) Starting with an empty BST, a sequence of n items is inserted into it, one at a time. The BST is not allowed to have any duplicate keys. What is the *worst case* big O running time to insert all n items? Show your derivation. Count item-to-item comparisons towards running time. (First get the answer in number of comparisons, then translate to big O).

b) and c):

Consider two binary search trees A and B. BST A has m nodes, and BST B has n nodes. Assume m is much greater than n. We wish to find the intersection of values from the two trees; i.e., we wish to identify all values in BST A which are also found in BST B. For each of the following, describe the reasoning for your answer. No reasoning = no credit.

b) We will perform an inorder traversal of BST A, and for each value we will perform a search in BST B for that value. What is the **worst-case** big O running time of this approach?

c) We will perform an inorder traversal of BST A, dumping the values into a list. We will do the same for BST B. Then we will find the intersection of the two lists. What is the **worst-case** big O running time?

SCRATCH PAGE

SCRATCH PAGE

SCRATCH PAGE