COMPUTER SCIENCE 112 - FALL 2016
MIDTERM EXAM 1

Name: _____

- Be sure your test has 3 questions.
- **DO NOT TEAR OFF THE SCRATCH PAGES OR REMOVE THE STAPLE.**
- Be sure to fill your name and circle your recitation 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

Question	Max	Score
1 Linked List Difference	25	
2 Modified Sequential Search	25	
3 Lazy Binary Search	25	
TOTAL	75	

1. **Linked Lists Difference (25 pts)**

   Suppose you are given two **sorted** lists of integers. The difference of the first list with respect to the second is the set of all entries that are in the first list but not in the second, *in the same order as in the original first list*. For example:

   ```
   L1: 3->9->12->15->21
   L2: 2->3->6->12->19

   Difference L1-L2 : 9->15->21
   ```

   Implement a NON-RECURSIVE method to compute the difference and return it in a NEW linked list. The original linked lists should not be modified. The new linked list should have a complete new set of `Node` objects (not shared with the original lists). You may assume that neither of the original lists has any duplicate items.

   Your implementation MUST use the efficient algorithm covered in Problem Set 2 on a similar problem: using a pointer per list to track simultaneously, traversing each list exactly once. If you use some other inefficient algorithm, you will get AT MOST HALF the credit. You may implement helper methods if needed.

   ```
   public class Node {
      public int data;
      public Node next;
      public Node(int data, Node next) {
         this.data = data;
         this.next = next;
      }
   }

   /**
    * Creates a new linked list consisting of the items that are in the first
    * list but not in the second, in the same order as in the first list
    * @param frontL1 List 1
    * @param frontL2 List 2
    * @return The front of this new linked list
    */
   public static Node difference(Node frontL1, Node frontL2) {
      // COMPLETE THIS METHOD - YOU MAY ***NOT*** USE RECURSION
   ```

CS 112 Fall '16 Exam 1; Name: _____

2. **Modified Sequential Search (25 pts,** 10 + 10 + 5**)**

   You are given the following modified sequential search code for searching in an array in which values are in **sorted** increasing order:

```
public boolean search(int[] A, int target) {
   for (int i = 0; i < A.length; i++) {
      if (target == A[i]) {
         return true;
      }
      if (target < A[i]) {
         return false;
      }
   }
   return false;
}
```

   (a) Assuming the length of the array is n, derive an exact formula as a function of n (**not** big O) for the **average** number of target-to-array item comparisons for searches on an array of size n, assuming that all possibilities of successful match are equally likely. (Count the == and < comparisons in the `if` conditions, IGNORE the < comparison made in the for loop header condition). Show your work by clearly explaining all the steps towards your formula - if you simply write a mathematical expression, you won't get any credit. However, once you have a formula, you are not required to simplify it down to a single term.

(b) Assuming the length of the array is n, derive an exact formula as a function of n (not big O) for the **average** number of target-to-array item comparisons for FAILED search on an array of size n, assuming equal likelihood of failure at all possible failure locations. (Same comparisons to count as in part (a) above.) Show your work by clearly explaining all the steps towards your formula - if you simply write a mathematical expression without you won't get any credit. However, once you have a formula, you are not required to simplify it down to a single term.

(c) Under what conditions would the modified sequential search be better than the regular sequential search? Explain.

3. **Lazy Binary Search (25 pts,** $8 + 7 + 10$**)**

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

```
boolean lazySearch(int[] A, int target) {
   int low = 0, high = A.length - 1;
   while (low < high) {
      int middle = (low + high) / 2;
      if (target > A[middle]) {  // C1
         low = middle + 1;
      }
      else {
        high = middle;
      }
   }
   return target == A[low];      // C2
}
```

   For parts (a) and (b), lazy search is done on the following array of integers:

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

   (a) Show the sequence of > and == comparisons (in the statements marked `C1` and `C2` respectively in the code) that would be done against the items in the array when searching for 65.

(b) Show the sequence of > and == comparisons that would be done against the items in the array when searching for 80.

(c) Given a sorted array of length 5, find the number of comparisons required by the lazy search code above to find a match against items in each of the positions. Count each > as one comparison, and each == as one comparison. Do not count the comparison in the `while` loop condition. Fill in the second column of the following table with your answers:

Array position	Number of comparisons
0	
1	
2	
3	

SCRATCH PAGE