CSCI C343 - Data Structures	First Name: _____
Summer 2017	Last Name: _____
Lab 07 Quiz	IU email: _____

Quiz in preparation for the Midterm Exam

Lab 07 Quiz Due Date and Time

- Friday, May 19 2017, from 12:45PM to 2:00PM.

Work Policy

Work for Lab 07 Quiz is to be completed individually - no group solutions or cooperative work.

Note: this week's Lab 07 is a written quiz, to help you prepare for the Midterm Exam next week. This quiz needs to be completed at lab-time, in LH030.

Instructions

Please ***read these instructions carefully and completely***.
Ask the person proctoring the test if anything is unclear, or if you have any questions.

The C343 Midterm Exam (and thus also this practice quiz) must be taken in written form, pen/pencil-and-paper, *not on your computer/tablet/cellphone/etc.* (those have to be all turned off and put away for the entire duration of the test). During the Midterm Exam, any book or other source, will be off-limits. If you consult any source of information during the Midterm Exam, at any time before submitting the test, you will receive a score of 0 on the Midterm Exam, and the incident will be reported to the university.

Note: since this is a practice quiz, you may consult any written notes, your (printed) textbook, etc. while taking the quiz at Lab 07 ; however, that will not be allowed during the Midterm Exam!

1. You have available 75 minutes to work on the Midterm Exam (and thus also this practice quiz). You may take a few more minutes if necessary. All submissions have to be turned in by 2:00PM ***at the latest***. Don't wait until the last few minutes. If you finish early, we recommend that you double-check all your answers before turning in your work and leaving the lab.
2. When a question asks to answer with an "expression", it means a mathematical expression, not program code.
3. When a question asks to answer with program code, it does not have to be precisely *exact* Java, as long as the meaning is clear and the sentences are Java-like.
4. For short answer problems, be as clear as possible. When in doubt, give details. Partial credit is possible on many problems, so don't leave anything blank.
5. During the Midterm Exam, you will not be allowed to consult any books, phones, computers, calculators, or neighbors. The Midterm Exam for C343 is closed-book, closed-everything, with one exception: you are allowed ***one*** 3"x5" index card with anything you want personally handwritten on it (both sides). ***No other index card sizes will be allowed, nor cards written by anyone else, printed material (not even on the index card), etc***. Before you leave the lab, write down your name on the index card, and turn it in to the person proctoring the exam.
   (for this practice test, you may consult written notes, the textbook, etc -- but no electronic devices nor any internet connection).

6. The Midterm Exam (and thus also this quiz) is to be turned in before leaving LH030. Late submissions of the Midterm Exam will not be considered for grading (nor for this quiz).
7. ***Mark*** your answer parts clearly when necessary, e.g. with (a), (b), (c), etc. and please ***write legibly***. Handwriting that is unreadable or very hard to read can not be graded and will receive 0 points.
8. Do ***not*** copy, email, transmit, etc. any part of the exam/quiz (neither questions nor answers). Cheating will be dealt with harshly.

Tasks

[total: 100 points]

A. Arrays and Lists

Below shows a class for an **array-based *list***, which lacks the implementation for an operation, `removeFirst()`.
The required functionality of the `removeFirst()` method is to *remove* the *first element* from the array-based list.
Write your implementation for the `removeFirst()` method below.
Note: you're allowed to use basic Java (or Java-like) operations in your solution code, but do not assume *any* calls to external help functions to be available.

```
public class MyList {
     private int[] numList;
     private int maxSize;
     private int currSize;
     public MyList (int size) {
          maxSize = size;
          numList = new int[maxSize];
          currSize = 0;
     }
     public void add (int anum) {
          assert currSize < maxSize : "exceeds maxsize";
          numList[currSize ++] = anum;
     }
     // TODO: implement your removeFirst() method here
```

Answer:

```
     public void removeFirst() {
          if currSize == 0 return; // ensure that the list is not empty
          // if you need the first element: tmp = numList[0];
          for (int i = 0; i < currSize; i++) { // shift down all elements by
-1
               numList[0] = numList[i+1] numList[0];
          }
          currSize-- ; // update list size
     }
```

```
}
```

B. **Coding and time complexity analysis**

a. Complete **a method for finding the maximum number in a Linked List, and a main method in the class to test your implementation**.
The binary node class (`ListNode`), and parts of the Linked List class (`List`) are given.
You ONLY need to complete the method `findMaxHelp()` and the `main()` method.
Note: you're allowed to use basic Java (or Java-like) operations in your solution code, but do not assume *any* calls to external help functions (e.g. `max()`) to be available.

```
// ----------------------------
public class ListNode {
     private int value;
     private ListNode next;
     public ListNode(int e) {
          value = e;  next = null;
     }
     public int getValue() { return value; }
     public ListNode getNext() { return next; }
}
// ----------------------------
public class List {
     private ListNode head; // the starting node of the List
     public List(int[] nums) {
          // build a List from an input array of integers
          // assume that there is code here and that it works correctly
     }
     public int findMax() {
          return findMaxHelp(head);
     }
     public int findMaxHelp(ListNode anode) {
     // TODO: complete the method findMaxHelp here
```

Answer:

```
     if ( anode.getRight() == null ) {
          return anode;           // base case: test anode's right node
     } else {                     // recursion:
          return findMaxHelp(anode.getRight());
     }                            // note: only need to visit right
subtree
```

```
     }
     // TODO: write your main/test/client method here
```

Answer:

```
public static void main () {
     int[] myData = [2,3,4,10,1,0];     // create data
     BST myBST = new BST(myData);       // instantiate class
```

```
        int myMin = myBST.findMinHeap();  // call method

    }
```

b. Show the time complexity of `findMax()` in terms of the total number of nodes (***n***), in the average case, and in the worst case. Explain your answers.

Answer:
average case: Θ(log(n))
worst case: Θ(n)
because on average, the height of a BST with n nodes is log(n), but a completely unbalanced tree has height n.

C. **Linked Lists**

a. ***Draw*** all distinct Linked Lists that can result from inserting any permutation of values 1, 2, and 3.

(you may assume that you have a method `insert(value)` available for the Linked List, that does the correct work for you)

Answer:

- [1]→[2]→[3]→
- [1]→[3]→[2]→
- [2]→[1]→[3]→
- [2]→[3]→[1]→
- [3]→[1]→[2]→
- [3]→[2]→[1]→

b. How many linked lists are there?

c. What are the probabilities of each linked list's occurring, if all permutations are equally likely?

D. Sorted Arrays

a. ***Draw*** the result of inserting 3, 1, 4, 6, 9, 2, 5, and 7 (these are all separate `insert()` operations) in an initially *empty* Sorted Array.

(you may assume that you have a method `insert(value)` available for the Sorted Array, that does the correct work for you)

b. ***Draw*** the result of deleting the first element.

Lab 07 Submission Instructions:

- Turn in your written quiz to the instructor by the end of lab time: Friday, May 19 2017, 2:00pm.

Last updated: May 18, 2017

mitja⊛indiana·edu