CS 1332 Exam 1

Spring Semester: 10 February 2014

Name (print clearly including your first and last name): ____________________

Signature: ____________________

GT account username (gtg, gth, msmith3, etc): ____________________

- Signing signifies you are aware of and in accordance with the **Academic Honor Code of Georgia Tech**.
- Calculators, cell phones, laptops, headphones, etc. are not allowed.
- Extra paper is not allowed. If you have exhausted all space on this test, come and talk with me (Monica)
- Pens/pencils and erasers are allowed.
- All code must be in Java.
- Efficiency matters. For example, if you code something that uses O(n) time or worse when there is an obvious way to do it in O(1) time, your solution may lose credit.
- Style standards such as (but not limited to) use of good variable names and proper indentation is always required. (Don't fret too much if your paper gets messy, use arrows or whatever it takes to make your answer clear when necessary.)
- Comments are not required unless a question explicitly asks for them.

Question	Points Possible	Points Received	Points Lost	Graded by TA Name (print)
Page 2.	14			
Page 3.	20			
Page 4.	20			
Page 5.	40			
TOTAL	94			

[2] 1. (a) ______________ What is the big O of accessing the element at the middle position of an ArrayList? (Or the middle two elements if the length is even.) Assume the ArrayList has n values.

[2] (b) ______________ What is the big O of removing the last node of a doubly linked list having a tail reference? Assume the list has n nodes.

[2] (c) ______________ What is the big O of adding to the end of a singly-linked list with no tail reference? Assume the list has n nodes.

[2] (d) ______________ What is the big O of removing all occurrences of a given data item, x, from a doubly-linked list having a head reference and a tail reference. Assume the list has n nodes.

[2] (e) ______________ This ADT is FIFO. (Do not ask what an ADT is. Obviously do not ask what FIFO is. This is a test and you are being tested to see if you know.)

[2] (f) Write an import statement to import the Java generic class ArrayList.

[2] (g) Imagine you have a class called Customer already written. You plan to store 100 Customer objects in an ArrayList using Java's generics capability. Write one line of code to declare an ArrayList of Customer with initial capacity 100.

[20] 2. Finish the code for the two incomplete methods below given the code for *Node* and *DoublyLinkedList*. Do not alter the code that is given. Simply finish removeFirstOccurrence method and the Node constructor.

```
public class DoublyLinkedList {
    protected Node head;  // Note there is no tail reference.

    public void addToFront(String data) {
        head = new Node(data, head, null);
    }

    public void removeFirstOccurrence(String data) { // complete the code
        // It removes the node having the first occurrence of the target data.
        // The search starts at the head. If the target data is not in the list, then
        // the list remains unchanged.  The code should not fail.
        // The next field of the last node has value null.  There is no tail reference.
        // Write your code here:




    protected class Node {
        protected String data;
        protected Node next;
        protected Node previous;

        private Node(String data, Node next, Node previous) {




    } // end of Node class
} // end of DoublyLinkedList class
```

[20] 3. Write a subclass of the class *DoublyLinkedList* from the previous question here. The subclass is called *IterableDoublyLinkedList*. The subclass is to implement *java.lang.Iterable*. It must have a private inner class implementing *java.util.Iterator*.

The iterator should iterate from the head of the list to the end. Note that there is no tail reference, but the next field of the last node has value null.

Do not implement the *remove* method of *java.util.Iterator* - just ignore it completely for this question.

The method *next*() should throw a *java.util.NoSuchElementException* if it is called yet the iterator already finished iterating through the list.

[40] 4. Write a generic class called *Stack* that includes a constructor and also implements the generic *StackADT<T>* interface given below. This interface is designed for this exam and requires a few classic Stack ADT methods. (It may or may not be identical to any Stack Interface you may have seen, so as always, read carefully.) Your class *Stack* is to be **generic** and is to use a **generic array of T of size 100** as the backend.

```
public interface StackADT<T> {

    /**
     * Pushes an item onto the top of this stack.
     * Throw java.lang.RuntimeException if the array is full.
     * Do not increase the physical size of the array.
     */
    public void push(T item);

    /**
     * Removes the item at the top of this stack and returns that object.
     * Throws java.util.EmptyStackException if the stack is empty.
     */
    public T pop();

    /**
     * Tests if this stack is empty.
     */
    public boolean isEmpty();
}
```

Class Stack is to contain the above methods and also one **constructor** plus any instance variables needed.

The constructor is required, and it must override Java's default parameterless constructor. This constructor is to create an empty Stack having the required array backend.

Write the class *Stack* here.

This page is intentionally left blank.