CSC172 LAB 2

LINKED LISTS

1 Introduction

The labs in CSC172 will follow a pair programming paradigm. Every student is encouraged (but not strictly required) to have a lab partner. Labs will typically have an even number of components. The two partners in a pair programming environment take turns at the keyboard. This paradigm facilitates code improvement through collaborative efforts, and exercises the programmers cognitive ability to understand and discuss concepts fundamental to computer programming. The use of pair programming is optional in CSC172. It is not a requirement. You can learn more about the pair programming paradigm, its history, methods, practical benefits, philosophical underpinnings, and scientific validation at http://en.wikipedia.org/wiki/Pair_programming

Every student must hand in their own work, but every student must list the name of their lab partner if any on all labs.

This lab has six parts. You and your partner(s) should switch off typing each part, as explained by your lab TA. As one person types the lab, the other should be watching over the code and offering suggestions. Each part should be in addition to the previous parts, so do not erase any previous work when you switch.

The textbook should present examples of the code necessary to complete this lab. However, collaboration is allowed. You and your lab partner may discuss the lab with other pairs in the lab. It is acceptable to write code on the white board for the benefit of other lab pairs, but you are not allowed to electronically copy and/or transfer files between groups.

2 A Simple Linked List

The goal of this lab is to gain familiarity with simple linked lists.

1. Begin this lab by implementing a simple class that represents a “node” in a linked list, as follows. Compile the following class.

```
public class MyNode<AnyType> {
     public AnyType data;
     public MyNode<AnyType> next;
}
```

2. Have the second member of your pair type in the code for this simple linked list interface:

```
public interface SimpleLinkedList<AnyType> {
     public void insert(AnyType x);
     public void delete(AnyType x);
     public AnyType lookup(AnyType x);
     public boolean isEmpty();
     public void printList();
}
```

3. Implement your own linked list class that implements `SimpleLinkedList`. A linked list class will typically contain a `MyNode` reference for the start of the list that is set to null at construction time to indicate an empty list. Implement the `isEmpty()` method according to this specification. Then, implement the `insert()` method in your class. Include in your comments the expected runtime of the insert method.
4. Implement the `printList()` method. Also implement a test program class with a main method that creates an instance of your list class, adds several items to the list, and then prints the list. Include in your comments the expected runtime of the `printList()` method.
5. Implement the `lookup()` method. The lookup method should return true if the object is contained in the list and return false otherwise. Modify your `insert()` method so as to prevent duplicate items (only insert an item if `lookup()` returns false). Modify your test program to demonstrate that the lookup method works.
6. Implement the `delete()` method. The delete method should do nothing if the item is not found in the list. If the item is found then it should modify the list to remove the item. Add appropriate tests in your test program to demonstrate these behaviors.

3 Hand In

Hand in the source code from this lab at the appropriate location on the BlackBoard system at my.rochester.edu. You should hand in a single zip file (compressed archive) containing your source code, README, and OUTPUT files, as described below.

1. A plain text file named README that includes your contact information, your partner's name, a brief explanation of the lab (A one paragraph synopsis. Include information identifying what class and lab number your files represent.), and one sentence explaining the contents of all the other files you hand in.
2. Several Java source code files representing the work accomplished for this lab. All source code files should contain author and partner identification in the comments at the top of the file. It is expected that you will have a file for the Node class, a file for the SimpleLinkedList interface, a file for your own linked list class and a file for the test program class.
3. A plain text file named OUTPUT that includes author information at the beginning and shows the compile and run steps of your code. The best way to generate this file is to cut and paste from the command line.

4 Grading

Your grade will be determined based on the correct implementation of the following (total 90%):

MyNode class and SimpleLinkedList interfaces – 15%

Generic linked list class and its constructor – 15%

insert() method – 15%

isEmpty() method – 5%

printList() method – 10%

lookup() – 15%

delete() method – 15%

README file accounts for the remaining 10%