CMSC 132 Quiz 2 Worksheet

The second quiz for the course will be on Wednesday, Sept 20, during your lab session. The following list provides more information about the quiz:

- The quiz will be a written quiz (no computer).
- Closed book, closed notes quiz.
- Answers must be neat and legible. We recommend that you use pencil and eraser.

The following exercises cover the material to be included in this quiz. Solutions to these exercises will not be provided, but you are welcome to discuss your solutions with the TA or instructor during office hours. **We strongly recommend you do not use Eclipse to write the code associated with these exercises.** Try to answer the exercises in a piece of paper and then use Eclipse to verify your solutions. This approach will better prepare you for the quiz.

**Exercises**

Implement the methods below based on the following Java class definitions.

```java
public class Node {
    Object data;
    Node next;
}

public class LinkedList {
    Node head;
}
```

1. Define a constructor for the *LinkedList* class that creates an empty list.

2. Define a method called *addFirst* that adds an object to the beginning of the list.

3. Define a method called *addLast* that adds an object to the end of the list.

4. Define a method named *size* that returns the size of the list (number of nodes in the list).

5. Define a method named *find* that has as parameter an object reference. The method will return true if the parameter object is part of the list and false otherwise. Use the *equals* method to compare elements.

6. Assume a LinkedList is storing *String* objects. Define a method called *insert* that has the following prototype:

   ```java
   public boolean insert(String entry, String target);
   ```

   The method will look for the first instance of *target* in the list and will insert *entry* before that instance. The method will return true if a *target* instance is found and false otherwise. The list should not be modified if a *target* instance is not found.

7. Assume a LinkedList is storing *Integer* objects. Define a method called *filter* that has the following prototype:

   ```java
   public LinkedList filter(Integer value);
   ```

   The method will return a new LinkedList with values that are less than or equal to the parameter. The method should not modify the original (current object) list.