[**Closed book and notes.**] Show all of your work clearly in the space provided or on the additional page at the end of the exam. If the additional page is used, clearly identify to which exam question it is related. Be sure to **read each problem carefully**. Note that the exam is double sided.

1. Consider the following method:

```
public static double minimum(List<Double> data) {
  if(data==null || data.isEmpty()) {
    throw new IllegalArgumentException("List cannot be null or empty");
  }
  double min = data.get(0);
  for(int i=1; i<data.size(); ++i) {
    if(min>data.get(i)) {
      min = data.get(i);
    }
  }
  return min;
}
```

(a) (10 points) What is the worst case asympototic time complexity for minimum when data is an ArrayList. Justify your answer.

(b) (10 points) What is the worst case asympototic time complexity for minimum when data is an LinkedList. Justify your answer.

1. continued

(c) (5 points) Suppose that the code for the `for` loop was "optimized" as follows:

```
for(int i=1; i<data.size(); ++i) {
  double value = data.get(i);
  if(min>value) {
    min = value;
  }
}
```

Would this change the answer to **(b)**? Why or why not?

(d) (5 points) Suppose that the code for the `for` loop was "optimized" as follows:

```
for(double value : data) {
  if(min>value) {
    min = value;
  }
}
```

Would this change the answer to **(b)**? Why or why not?

2. (10 points) Explain why the indexed based getDesiredDots() method required considerably more time to run than the iterator based getDesiredDots2() method when using a LinkedList in lab 3.

3. Consider an implementation of a LinkedList<E> that has three attributes and an inner Node<E> class as shown below:

```
public class LinkedList<E> implements List<E> {
  private Node<E> head = null;
  private Node<E> tail = null;
  private Node<E> mostRecent = null;
  private int mostRecentIndex = -1;

  private static class Node<E> {
    private E value;
    private Node<E> next;
    private Node<E> prev;

    private Node(E element, Node<E> next, Node<E> prev) {
      value = element;
      this.next = next;
      this.prev = prev;
    }
  }
  // ...
}
```

The mostRecent and mostRecentIndex attributes must store information about the element in the list that has been access most recently via a method that uses an index, e.g.,

```
public E get(int index);
public E set(int index, E element);
public void add(int index, E element);
etc...
```

See next page for questions...

3. continued

(a) (20 points) Implement get() to optimally make use of the mostRecent and mostRecentIndex attributes.

(b) (10 points) When will your get() implementation work best? When will your get() implementation work worst? Justify your answers.

4. (20 points) Implement an inner class for the ArrayList we developed in lecture that implements the Iterator<E> interface. You do not need to provide an implementation for the remove method.

5. (10 points) Recall that our implementation of the ArrayList<E> from lecture contains one attribute: data which is a reference to an array that stores all of the elements. Implement the add(E element) method.