[**Closed book and notes.**] Show all of your work clearly in the space provided. 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. Explain your answer.

**(b)** (10 points) What is the worst case asympototic time complexity for minimum when data is an LinkedList. Explain 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.** (15 points) Suppose a java.util.LinkedList<String> contains three elements: “This”, “is”, and “fun”. Draw a model of how all of the data is stored in memory. Be as detailed as possible.

**3.** (15 points) Recall that the Roster class used in lab 3 had three attributes (the first two inherited from AbstractRoster):

```
protected int size;
protected int capacity;
private Student[] students;
```

Implement the clone() method for the Roster class.

**4.** (10 points) Consider the following partial implementation of the SinglyLinkedIterator that could be used to replace the iterator implementation discussed in lecture.

```
private class SinglyLinkedIterator implements Iterator<Integer> {
  private int index;

  private LinkedRosterIterator() {
    index = -1;
  }

  public boolean hasNext() {
    boolean hasNext = true;
    try {                // Try to get the next element
      get(index + 1);
    } catch(IndexOutOfBoundsException e) {
      hasNext = false;  // Return false if unable to get the next element
    }
    return hasNext;
  }

  public Integer next() {
    Integer value = null;
    if(hasNext()){
      ++index;
      value = get(index);
    }else{
      throw new NoSuchElementException("Iteration has no more elements");
    }
    return value;
  }
}
```

Which implementation (this one or the one from lecture) is preferred. Justify your answer.

**5.** (15 points) Consider the simple ArrayList class that we developed in lecture. Recall that the class had one attribute:

```
E[] array;
```

Implement the following method for the class. You may use size() but no other methods from the ArrayList class in your implementation.

```
public boolean addAll(Collection<? extends E> collection) {
  boolean isChanged = false;
```

```
  return isChanged;
}
```

**6.** (15 points) Consider the simple LinkedList class that we partially implemented in lecture. Recall that the class has two attributes:

```
private Node head;
private int size;
```

and that the inner Node class has two attributes:

```
private E element;
private Node next;
```

Implement the contains() method for the LinkedList class. You may not make use of any other methods in the LinkedList class.

}

Additional work area for any problem. Clearly identify which problem is associated with the work on this page.