The exam has 5 numbered questions that are worth 20 points each.

This is a closed-book, closed-notes, closed-Internet exam. You can type your answers into Microsoft Word or your favorite text editor, or if you prefer you can take with pen and paper, and scan your answers into a PDF file. Either way, email your answers back to me by noon on Saturday, March 6 (Oberlin time). Time yourself. If you do not have extended time on exams you have one hour for this exam.

After you have finished the exam please indicate whether you followed the Honor Code on the exam. Also give the time you started working on this exam, and the time when you finished it.

I □ did    □ did not
follow the Honor Code while taking this exam.

Start time: _____________________
Finish time: _____________________

__________________________________
Signature
1. Suppose S is a stack that starts off empty and we do the following sequence of operations:

```plaintext
S.push(1);
S.push(2);
S.push(3);
S.pop();
S.push(4));
S.pop();
```

**What will S.top() return?**

Now we continue with this same stack and do
```plaintext
S.pop();
S.push(5);
S.push(6);
S.pop();
```

**What will S.pop() return now?**

**What will another S.pop() return?**

**What will another S.pop() return?**

Next, we do this with a queue. Suppose Q is an empty queue and we do
```plaintext
Q.enqueue(1);
Q.enqueue(2);
Q.dequeue();
Q.enqueue(3));
Q.dequeue();
```

**What will Q.front() return?**

Now we do two enqueues:
```plaintext
Q.enqueue(4);
Q.enqueue(5);
```

**What will Q.dequeue() return?**

**What will another Q.dequeue() return?**
2. Consider the following abstract class that holds two objects of class E:

```java
public abstract class Pair<E> {
    abstract E getFirst(); // return the first element of pair
    abstract E getSecond(); // returns the second element
    abstract E setFirst(E item);
    abstract E setSecond(E item);
    void isTwin() {
        return getFirst() == getSecond();
    }
    void switcheroo() {
        E temp = getFirst();
        setFirst(setSecond());
        setSecond(temp);
    }
}
```

Explain what is involved in making a non-abstract version of Pair<E>. You don’t need to write any code; just say what needs to be done. Alternatively, if you prefer coding write the following class:

```java
public class MyPair<E> extends Pair<E> {
    ....
}
```
3. Remember your code for MyArrayList from Llab 2. It probably started

```java
public class MyArrayList<E> extends AbstractList<E> {
    private int size;
    private E[] data;
    ...
}
```

Give code for a new MyArrayList<E> method

```java
public void reverse()
```

that reverses the order of the data in a list. This modifies the list; it does NOT make a new list. So if L starts as the MyArrayList<Integer> with data {1, 2, 3} then after we do L.reverse() L will have data {3, 2, 1}
Here is a new operation with lists. Method increment(L) works with lists of integers by adding 1 to the value of each element. Here is the way it is implemented:

```java
public void increment( LIST L) {
    for (int i = 0; i < L.size(); i++) {
        int oldValue = L.get(i);
        L.set(i, oldValue+1);
    }
}
```

In this code the “type” LIST is a placeholder for either ArrayList<Integer> or LinkedList<Integer>. If L is a list with values such as {4, 8, 10}, increment(L) adds 1 to each of those values, producing the list {5, 9, 11}.

a) Give a Big-Oh estimate for the number of steps it takes to run increment(L) on an ArrayList of size n. Also give a 1-sentence explanation of your estimate.

b) What is your estimate if L is a LinkedList? If your estimate and explanation are the same as in part (a) just write “Same”. If something is different for LinkedLists please explain.
5. In this question we will make a new data structure that I call BobStruct. I have a copyright on that name, so don’t try to steal it. BobStructs hold int values and only have two methods:

```java
public void add(int x) // this adds x to the structure
public int removeBiggest() // removes the largest value from the
// structure and returns it
```

Here is the start of an implementation based on AArrayLists:

```java
public class BobStruct {
    ArrayList<Integer> data;
    public BobStruct() {
        data = new ArrayList<Integer>();
    }
    ...
}
```

Give code for the add(int x) and removeBiggest() methods. Also give a Big-Oh estimate of the running time for each method on a BobStruct that contains n ints. You do not need to explain your Big Oh estimates.