CS 201, Summer 2015 — July 9th — Exam 2 Name: Solution

Question 1. [7 points] State a big-O upper bound on the running time of the following method, where the problem size N is the number of elements in the array passed as the method's parameter. Explain your answer briefly.

```
public static int q1(int[] arr) {
  int sum = 0;
  for (int i = 0; i < arr.length; i++) {
    for (int j = 0; j < arr.length * arr.length; j++) {
      sum += arr[(i+j) % arr.length];
    }
  }
  return sum;
}
```

N times

N^2 times

O(1)

$N \cdot N^2 \cdot O(1)$ is $O(N^3)$

Question 2. [7 points] State a big-O upper bound on the running time of the following method, where the problem size N is the number of elements in the array passed as the method's parameter. Explain your answer briefly.

```
public static int q2(int[] arr) {
  int sum = 0;
  for (int i = 0; i < arr.length; i++) {
    for (int j = 0; j <= i; j++) {
      sum += arr[(i+j) % arr.length];
    }
  }
  return sum;
}
```

dependent inner loop → 1, 2, 3, 4, ..., N times

O(1)

$$\sum_{i=1}^{N} i = \underbrace{1 + 2 + 3 + \ldots + (N-1) + N}_{N/2 \text{ pairs, each pair sums to } N+1} = \frac{N}{2}(N+1) = \frac{N^2}{2} + \frac{N}{2}$$

which is $O(N^2)$

$O(N^2) \cdot O(1)$ is $O(N^2)$

Question 3. [7 points] State a big-O upper bound on the running time of the following method, where the problem size N is the number of elements in the `ArrayList` passed as the method's parameter. Explain your answer briefly.

```
public static int q3(ArrayList<Integer> list) {
  int sum = 0;
  for (int i = 0; i < list.size(); i++ ){
    sum += list.get(i);
  }
  return sum;
}
```

N times

O(1) for ArrayList

$N \cdot O(1)$ is $O(N)$

Question 4. [7 points] State a big-O upper bound on the running time of the following method, where the problem size N is the number of elements in the `ArrayList` passed as the method's parameter. Explain your answer briefly.

```
public static int q4(ArrayList<Integer> list) {
  int sum = 0;
  while (list.size() > 0) {
    sum += list.remove(0);
  }
  return sum;
}
```

cost proportional to number of elements shifted: 0, 1, 2, ..., N-1

$$\sum_{i=0}^{N-1} i = \overbrace{0+1+2+\ldots+(N-2)+(N-1)}^{N-1} = (N-1)\frac{N}{2}$$

N/2 pairs, each sums to N-1

which is $O(N^2)$

$O(N^2)$ overall

Question 5. [7 points] State a big-O upper bound on the running time of the following method, where the problem size N is the number of elements in the `LinkedList` passed as the method's parameter. Explain your answer briefly.

```
public static int q5(LinkedList<Integer> list) {
  int sum = 0;
  for (int i = 0; i < list.size(); i++) {
    sum += list.get(i);
  }
  return sum;
}
```

get(i) — time is proportional to i, so 0, 1, 2, ..., N-1

$\sum_{i=0}^{N-1} i = (N-1)\frac{N}{2}$ which is $O(N^2)$

Question 6. [7 points] State a big-O upper bound on the running time of the following method, where the problem size N is the number of elements in the `LinkedList` passed as the method's parameter. Explain your answer briefly.

```
public static int q6(LinkedList<Integer> list) {
  int sum = 0;
  Iterator<Integer> i = list.iterator();
  while (i.hasNext()) {
    sum += i.next();
  }
  return sum;
}
```

hasNext(): N times

next(): O(1)

hasNext(): O(1)

$N \cdot O(1)$ is $O(N)$

Question 7. [8 points] Complete the generic `findMin` method below. The method should return the minimum element in the `Collection` parameter `c`. You can assume that `c` will contain at least one element. Use the `Comparator` parameter `comp` to compare elements to each other. Hint: use an iterator to access the elements in the collection.

```
public static<E> E findMin(Collection<E> c, Comparator<E> comp) {
    E min;
    Iterator<E> i = c.iterator();
    min = i.next();
    while (i.hasNext()) {
        E elt = i.next();
        if (comp.compare(elt, min) < 0) {
            min = elt;
        }
    }
    return min;
}
```

Question 8. [8 points] Complete the generic `Box` class below. An instance of `Box` should store one value of type `E`, where `E` is the element type. Use the following JUnit tests (which begin on the left and continue on the right) specify the `Box` class's required methods and behavior:

```
Box<String> bs =
  new Box<String>("hello");
Box<Integer> is =
  new Box<Integer>(42);

assertEquals("hello", bs.get());
assertEquals((Integer)42, is.get());
```

```
bs.set("yeah");
is.set(17);

assertEquals("yeah", bs.get());
assertEquals((Integer)17, is.get());
```

```
public class Box<E> {
    private E value;

    public Box(E v) {
        value = v;
    }

    public E get() {
        return value;
    }

    public void set(E v) {
        value = v;
    }

}
```