[**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. (10 points) Consider the following method:

```
1  public static int countOdd(List<Integer> numbers) {
2      if(numbers==null) {
3          throw new IllegalArgumentException("List cannot be null");
4      }
5      int count = 0;
6      for(Integer number : numbers) {
7          if(number%2!=0) {
8              count++;
9
10         }
11     }
12     return count;
13 }
```

Suppose that a java.util.LinkedList with n elements is passed to countOdd().
True/False (**T** or **F**)

____ The worst-case asymptotic time complexity for countOdd() is $O(n)$.

____ The worst-case asymptotic time complexity for countOdd() is $O(n^2)$.

____ The worst-case asymptotic time complexity would be different if we used a java.util.ArrayList instead of a java.util.LinkedList.

____ The worst-case asymptotic time complexity would be different if we used the ArrayList implemented in lecture instead of a java.util.LinkedList.

____ The worst-case asymptotic time complexity would be different for a list containing all even numbers than for a list containing all odd numbers.

____ numbers could be a Collection<Integer> instead of List<Integer> and not require **any** other changes.

For countOdd() to function correctly, the following methods must be implemented in the LinkedList class passed to the method: (indicate **Y** or **N** for each method)

____ size()

____ iterator()

____ isOdd()

____ iterable()

2. (5 points) Consider the following method:

```
1  public static int doSomethingStrange(List<Integer> numbers) {
2      int count = 0;
3      int sum = 0;
4      for(Integer number : numbers) {
5          if(number%2!=0) {
6              count++;
7          }
8          sum = numbers.get(count);
9      }
10     return sum + count;
11 }
```

Suppose that a java.util.LinkedList with n elements is passed to doSomethingStrange(). True/False (**T** or **F**)

____ The worst-case asymptotic time complexity for doSomethingStrange() is $O(n)$.

____ The worst-case asymptotic time complexity for doSomethingStrange() is $O(n^2)$.

____ The worst-case asymptotic time complexity would be different if we used a java.util.ArrayList instead of a java.util.LinkedList.

____ The worst-case asymptotic time complexity would be different for a list containing all even numbers than for a list containing all odd numbers.

____ numbers could be a Collection<Integer> instead of List<Integer> and not require **any** other changes.

3. (7 points) Explain how black-box and white-box testing differ.

4. (10 points) Assume n is the number of lists in the listOfLists. Explain why the following algorithm is $O(n)$ for the java.util.ArrayList implementation and $O(n^2)$ for the wk1.ArrayList implementation.

```
public static List<Integer> getSizes(List<List<String> listOfLists) {
    List<Integer> sizes = new ArrayList<>();
    for(List<String> list : listOfLists) {
        sizes.add(list.size());
    }
    return sizes;
}
```

5. (8 points) Describe the differences between the Iterable<E> and Iterator<E> interfaces.

6. (15 points) Recall that our ArrayList had one attribute: E[] data. Implement the ArrayList.add(int index, E element) method.

7. Suppose we modify the LinkedList implementation from lecture so that there is only one attribute: Node<E> head (no tail).

(a) (12 points) Implement the add(E element) method. You may assume that the Node inner class has been implemented; however, you may not make use of any other methods in the LinkedList class to complete your implementation.

(b) (3 points) What is the Big-O time complexity for your implementation in part **(a)**? Justify your answer.

8. (15 points) Complete the implementation of the following method that will return the average length of every other element in $O(n)$ time. You may assume that phrases contains at least one element, no elements are `null`, and that we are using the java.util implementation.

For example, [”may”, ”your”, ”shoes”, ”always”, ”fit”] should result in $3.67 = (3 + 5 + 3)/3$.

```
public static double averageLengthOfEveryOtherElement(ArrayList<String> words) {
```

9. (15 points) Complete the implementation of the following method that will return the average of every other element in $O(n)$ time. You may assume that numbers contains at least one element, no elements are `null`, and that we are using the java.util implementation.

For example, [1.0, 2.0, 3.0] should result in $2.0 = (1.0 + 3.0)/2$.

```
public static double averageOfEveryOtherElement(LinkedList<Double> numbers) {
```

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