(a) Declare a reference to a Collection or Strings and have it refer to a newly created instance of an ArrayList containing two elements: "Quiz" and "1".

(b) List two methods that are part of the List interface that are not part of the Collection interface.

(c) Recall that our ArrayList implementation from lecture had one instance variable, data, and a default constructor with the following implementation:

```
public ArrayList() {
    data = (E[])new Object[0];
}
```

Implement the add(E) method.

(a) Give the big-O notation for the time complexity of the following method as a function of the length of data – n. Justify your answer.

```
1 public static int[] addFront(int[] data, int value) {
2   int[] newArray = new int[data.length+1];
3   newArray[0] = value;
4   for(int i=0; i<data.length; ++i) {
5     newArray[i+1] = data[i];
6   }
7   return newArray;
8 }
```

(b) Recall that the LinkedList class we started developing in lecture has one attribute, head that is `null` when the list is empty. The head attribute is of type Node – an inner class with two attributes: E value and Node next. Complete the implementation of the size() method below.

```
@Override
public int size() {
```

Assume we are using the LinkedList implementation developed in lecture. Consider the following method:

```
1  public static int countOdd(LinkedList<Integer> numbers) {
2      if(numbers==null) {
3          throw new IllegalArgumentException("LinkedList 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 }
```

Assume n is the size of the LinkedList passed to countOdd()
True/False (**T** or **F**)

_____ The asymptotic time complexity for countOdd() is $O(n)$.

_____ The asymptotic time complexity for countOdd() is $O(n^2)$.

_____ The asymptotic time complexity would be different if we used a java.util.LinkedList instead of the LinkedList implemented in lecture.

_____ The asymptotic time complexity would be different for a list containing all even numbers than for a list containing all odd numbers.

_____ The asymptotic time complexity would be different if numbers.get(count); is inserted on line 9.

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

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

_____ size()

_____ iterator()

_____ isOdd()

_____ iterable()

List the methods that are part of a pure stack interface.

Explain how a pure stack interface could be implemented using a LinkedList.

(a) We have a triangle made of blocks. The topmost row has 1 block, the next row down has 2 blocks, the next row has 3 blocks, and so on. Compute recursively (no loops or multiplication) the total number of blocks in such a triangle with the given number of rows.

For example,

- triangle(0) → 0
- triangle(1) → 1
- triangle(2) → 3
- triangle(5) → 15

```
public int triangle(int rows) {
```

(b) Given a string, compute recursively a new string where all the x chars have been removed. For example,

- noX("xaxb") → "ab"
- noX("abc") → "abc"
- noX("xx") → ""

```
public String noX(String str) {
```

Recall that our BinarySearchTree class had an inner, Node, class that contained three attributes: value, lKid, and rKid. Implement the recursive version of BinarySearchTree.toString() that is called by the method below such that the following method returns a string containing all of the elements in the binary search tree listed in order (separated by commas).

```
public String toString() {
    String result = toString(root);
    return "[" + (result.length()>2 ? result.substring(0, result.length()-2) : "") + "]";
}
```

What is a Set? How does it differ from a List?

Explain how a Set can be implemented by adapting a class that implements a Map.

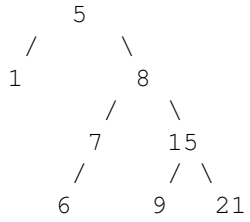
(a) Suppose the following Integers are added (in order) to a hash table using chaining and a capacity of 6. Illustrate the resulting data structure. Note that the hashCode() method for the Integer class just returns the value of the integer.

0, 8, 1, -5, 6, 7

(b) Repeat the process for open hashing with linear probing.

(c) What is the load factor, L, for this hash table?

(a) Write the AVL balance factor next to each node, and indicate which node(s) violate the properties of an AVL tree.



(b) Draw the tree shown in part (a) after left rotate has been performed on the node containing 8.