The following class definition implements a `StringList` class. Use this class definition for the first four questions. (Portions will be repeated if needed in the individual questions; most of the class definition is given here for reference.)

```
/** String List Collection Class */
public class StringList {
  // instance variables
  private String[] strings;     // Strings in this StringList are stored
  private int      numStrings;  // in strings[0..numStrings-1]


  /** Construct a new empty StringList with the given capacity */
  public StringList(int capacity) {
    strings = new String[capacity];
    numStrings = 0;
  }

  /** Get the array strings */
  public String[] getStrings() {
    return strings;
  }

  /** Get the number of strings in the list */
  public int getNumStrings() {
    return numStrings;
  }

  // other methods omitted to save space

  /** Return whether this StringList contains str */
  public boolean contains(String str) {
    // linear search
    for (int k = 0; k < numStrings; k++) {
      if (str.equals(strings[k])) {
        return true;
      }
    }
    return false;
  }
}
```

Question 1. (7 points) Write the Java specification and implementation for a public method in the `StringList` class called `remove` that takes a `String str` as a parameter and removes `str` from `strings` if `str` is in the array. The method should return `true` if an element was successfully removed from `strings` and `false` if an element is not removed. The method should shift elements toward the front of the array and update `numStrings` appropriately.

Fine points: If there is more than one copy of the string to be removed in the array, only the first one should be removed. Unused positions in the Strings array should contain null.

Example:
Before remove("mary") is called:
strings → ["joe", "mary", "ann", "bob", "carol", "mary", null]
numStrings → 6

After remove("mary") is called:
Strings → ["joe", "ann", "bob", "carol", "mary", null, null]
numStrings → 5

Question 2. (7 points) Assume we have the following sorting method in the class StringList.

```
/** mystery sort
 *  Postcondition: the strings in array strings are in alphabetical
 *  order */
public void mysterySort() {
  String str;
  int position;
  for (int k = numStrings - 1; k >= 0; k--) {
    str = strings[k];
    position = k;
    while ((position < (numStrings - 1)) &&
           (str.compareTo(strings[position + 1]) > 0)) {
      strings[position] = strings[position + 1];
      position++;
    }
    strings[position] = str;
    // INDICATE CONTENTS OF strings AT THIS POINT
  }
}
```

(a) (4 points) Trace the method and indicate the contents of the array `strings` at the end of each iteration of the `for` loop. Assume that the array `strings` contains the following elements before `mysterySort` is called:

```
strings = ["joe", "tammy", "hal", "carter"]
numStrings = 4
```

Use the following chart for the trace.

Variables	**Contents of `strings`**
	Iteration 1:
`k =`	Iteration 2:
	Iteration 3:
`str =`	Iteration 4:
	(keep going if needed until mysterySort completes execution)
`position =`	

Question 2 (cont). (b) (2 points) What is the invariant for the mysterySort sorting algorithm? You can either draw a picture or explain the invariant in English, or a combination of the two.

(c) (1 point) Which sorting algorithm is mysterySort **most** similar to with respect to its operation? (Circle the correct answer.):

A. Bubble Sort

B. Insertion Sort

C. Selection Sort

Question 3. (4 points) Suppose the `strings` array in `StringList` is unsorted. There are N strings in the list `(i.e., numStrings`=N). What is the **average** number of required `String` comparisons to find an element in the StringList using **linear search**?

Now suppose the `strings` array in `StringList` is sorted. Again, there are N strings in the list. What is the **maximum** number of required String comparisons to find an element in the StringList using **binary search**?

Question 4. (12 points) Complete the following method for class `StringList`. Method `mergeList` takes as a parameter a `StringList sl` and merges the strings in `sl` with the current object's array of Strings. See the method header for pre- and post-conditions.

Notes: You must be sure that there is enough room in `strings` for the merged list. Your code should **merge the lists directly** and not add the new strings at the end and then sort the array. *Hints: Keep position variables for each list. Check to see which string at the positions is alphabetically smaller and insert the string where necessary. Then update the corresponding position variable. You may find it helpful to allocate a new array to hold the merged lists.*

The following example illustrates this method. Suppose list is a StringList with the following contents:
list.strings → ["apple", "banana", "pear", "strawberry"]
and list2 is a StringList with the following contents:
list2.strings → ["broccoli", "carrot", "potato"]
Then after execution of list.mergeList(list2); the contents of list should be
list.strings → ["apple", "banana", "broccoli", "carrot", "pear", "potato", "strawberry"]

```
/** Merge strings in sl with current object's strings
 *  Precondition: sl.getStrings() are in sorted order and strings are in
 *  sorted order.
 *  Postcondition: strings contains all strings it originally had plus
 *  all strings in s1.getStrings(), and all strings are in sorted order.
 */
public void mergeList(StringList sl) {
```

} (end of questions involving class StringList)

Question 5. (5 points) You are working for an online retail store that sells clothing. Your manager has asked you to modify the Customer class so that it assigns a new unique customer ID number when a customer object is created. The current implementation constructs a new Customer object using the ID number supplied as a parameter.

Your manager no longer trusts client code to assign unique ID numbers when creating Customer objects. **Show the changes** that you need to make to the existing Customer class below so each new Customer object has a unique ID number. The ID number of the first newly created customer object should be 1, the next customer object created should have ID number 2, and so on.

Restriction: you **may not** use any arrays or lists (ArrayLists) for this problem. They are not needed.

```
/** A class representing a Customer */
public class Customer {

   // instance variables
   private int idNumber;               // customer ID number
   private String name;                // customer name
   private ShoppingCart items;         // customer's shopping cart of
                                       // to-be-purchased items
   private Address mailingAddress;     // customer's mailing address



   /** Construct a new customer object with idNumber, name, and
    * mailing address */
   public Customer(int idNumber, String name, Address mailingAddress) {

       this.idNumber = idNumber;

       this.name = name;

       this.mailingAddress = mailingAddress;

       this.items = new ShoppingCart();

   }

   // other methods omitted to save space
}
```

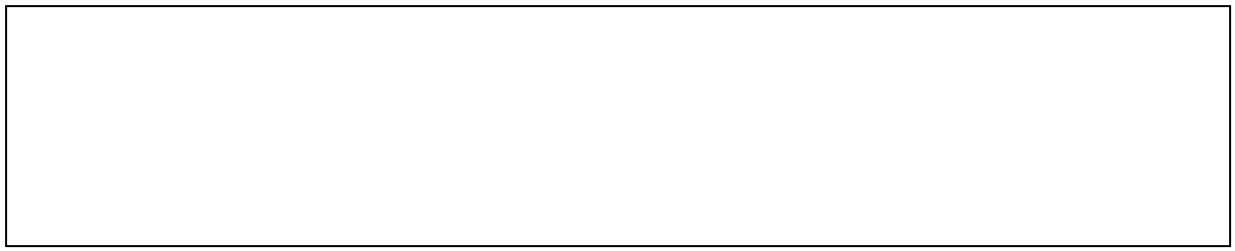
Question 6. (11 points) The online retail store sells shirts, pants, and jackets. Here are the classes representing RetailItems and, on the next page, Shirts.

```
/** A class representing a RetailItem */
public class RetailItem {
   // instance variables
   private double price;                // price of item
   private String manufacturer;         // manufacturer of item

   /** Construct a RetailItem with given price and manufacturer */
   public RetailItem(double price, String manufacturer) {
      this.price = price;
      this.manufacturer = manufacturer;
   }

   /** return price */
   public double getPrice() {
     return price;
   }

   /** return manufacturer */
   public String getManufacturer() {
     return manufacturer;
   }

   /** put item on sale by reducing cost by percentage given,
    * return true if successful, and false otherwise */
   public boolean putOnSale(double percentage) {
      if (0.0 < percentage && percentage < 1.0) {
          price = price * (1.0 - percentage);
          return true;
      }
      return false;
   }

   /** put item on sale by reducing cost by 50 percent */
   public boolean putOnSale() {
       return putOnSale(.50);
   }

   /** return a String representation of this RetailItem */
   public String toString() {
      return "RetailItem[price = " + price + ", manufacturer = " +
             manufacturer + "]";
   }
}
```

(continued on next page)

Question 6 (cont).

```
/** A class representing a Shirt */
public class Shirt extends RetailItem {
   // instance variables
   private String size;                 // size of shirt - S, M, L, XL

   /** Construct a Shirt item with given price, manufacturer, and size*/
   public Shirt(double price, String manufacturer, String size) {
      super(price, manufacturer);
      this.size = size;
   }
   /** return size */
   public String getSize() {
       return size;
   }
   /** return a String representation of the shirt */
   public String toString() {
```



```
   }
}
```

(a) (2 points) Complete the `toString` method, which should return a String that contains complete information about the state of a Shirt object, above.

(b) (5 points) For each of the following statements, answer the questions that appear indented under the statements. Assume that the statement(s) in each part is(are) written in a main method in a class called `Test` and are executed independently of other parts of the question. Note: this question is continued on the next page.

```
RetailItem r = new RetailItem(25.99, "Eddie Bauer");
```

What is/are the static type(s) of r? _____

What is/are the dynamic type(s) of r? _____

```
Shirt s = new Shirt(45.50, "Land's End", "L");
```

What is/are the static type(s) of s? _____

What is/are the dynamic type(s) of s? _____

(continued on next page)

```
RetailItem t = new Shirt(80.50, "Ralph Lauren", "M");
```

What is/are the static type(s) of t? _____

What is/are the dynamic type(s) of t? _____

```
RetailItem t = new Shirt(80.50, "Ralph Lauren", "M");
t.getSize();
```

Will the second statement produce a compiler error (Yes or No)? _____

Why or why not? _____

```
RetailItem t = new Shirt(80.50, "Ralph Lauren", "M");
t.toString();
```

Will the second statement produce a compiler error (Yes or No)? _____

If yes, what is the error? If no, what String will be returned?

(c) (4 points) Complete the sentences below with the best term/phrase from the following:

abstract, boolean, class, class definition, client, constructor, declaration, double, dynamic dispatch, expression, inherit, interface, instance variable, instance, invariant, local variable, loop, method, name thingy, overload, override, parameter, precondition, postcondition, return value, scope, scratch space, state, String, subclass, superclass, this, type, void

The class Shirt is a(n) _____ of the class RetailItem. Thus, the class Shirt _____(s) all public properties and responsibilities of RetailItem. The method toString in the Shirt class _____ (s) the toString method in the RetailItem class. The method putOnSale in the RetailItem class is said to be _____(ed/en). When putOnSale is called, Java chooses to execute the method whose definition for putOnSale matches in the number and _____(s) of the _____(s). When creating a new Shirt, the code super(price, manufacturer) is executed, which calls the _____ of the class RetailItem.

Question 7. (10 points) Use the `isPalindrome` method below to answer the following questions. Assume `isPalindrome` is defined in a class called `Word`.

```
public class Word {
  ...
  /** isPalindrome returns true if word is a palindrome -- a word that
   * has the same character sequence as the reverse of the word.
   * Precondition: start <= end, start >= 0, end < word.length
   * Postcondition: returns true if word is a palindrome, false if word
   * is not a palindrome
   */
  public static boolean isPalindrome(char[] word, int start, int end) {
    if (start == end) {       // looking at same character
                              // char[start] must be equal to char[end]
        return true;
    } else if ((start + 1 == end) && (word[start] == word[end])) {
        return true;
    } else if (word[start] != word[end]) {
        return false;
    } else return isPalindrome(word, start+1, end-1);
  }
  ...
}
```

(a) (2 points) Label the base case(s) and the recursive case(s) in the method definition above. Indicate the cases in the left margin next to the code above.

(b) (3 points) Suggest a set of test cases (typical cases, edge cases, “incorrect” cases) to test the `isPalindrome` method. For each test case, describe why you included it in your set of test cases.

(c) (4 points) Assume that the character array `racecar` has been initialized to the following in the Dr. Java interactions window:

```
char[] racecar = {'r','a','c','e','c','a','r'};
```

Draw the scoping diagram showing what happens during execution of the following method call in the Dr. Java interactions window:

```
boolean racecarPalindrome =
      Word.isPalindrome(racecar, 0, racecar.length-1);
```

Boxes for the scopes of the interactions window and class Word are drawn for you. Code repeated for reference:

```
public class Word {
   ...
  /** isPalindrome returns true if word is a palindrome ...  */
  public static boolean isPalindrome(char[] word, int start, int end) {
    if (start == end) {
        return true;
    } else if ((start + 1 == end) && (word[start] == word[end])) {
        return true;
    } else if (word[start] != word[end]) {
        return false;
    } else return isPalindrome(word, start+1, end-1);
  }
  ...
}
```

interactions

Word

(d) (1 points) What happens if the following is executed in the Dr. Java interactions window? (Just saying "the precondition is violated" is not enough – explain what will happen during execution in a sentence or two.)

```
boolean racecarPal = Word.isPalindrome(racecar, 4, 2);
```