APCS Cool Set 6

1. Write a program that generates the following statistical output:

This program looks at 344 responses between 0 and 20
The mean (average) value for this run is: 9.89

****

The unsorted array of 344 responses is:

```
19  1  11   1  14   4   8  11   3   2   4  18  20   6   4   2   2   3   7   9
 3   8   5  19   0  17  16  18  20   3  12   8  18   3   7  12  14   2   6  15
 2  15   1   7  11   7   1  18   1  14   0  10  15  14  12   1  18   8  19  10
19  17   0   2   7   1   2   1   3  14   9   8  17  20   6  13  10   4  18   3
 1  10   8   6   4   9   6   6   3   1  13   9  10  13  11   1   5  14  10  11
 1  17   8  19  20   4  14   3   3   8   0   2  19  11  13   6  16   7   7   9
 5   3  12   4  19   2   5  10  17  17  13  10  20  20   4  17   8  20   3   3
 5  10   8   2   1   8  19   4  18   4   6  17  20  20  19  13   2   0   2   3
 3   9   5  14  20  14  19  10  19  16   2  19  18  17  17  16  20  18  15   0
15  15   6   3  17  13   2  14   4   8  13  17  11  16   4   7  11   9  17   9
 5  20  11   1   7   0  19  18   7  13   7  18   9   3  15   5  19  10   5   4
16   6  17  15   9  10  19  10   3   5   0  11  18  13   6   7   9  15   7   2
 7   5  17  16   2   5   3   4   6   7  10   0  13  10   5  20   0   2  13  10
14  13  12   3   3  12  18   0  13   6   3  10  19  19  14   3   7   6   3  14
20   7  16   5   0   9  19  16   3   4  13  18   5   0  12  16  17  20  19   6
10   4  20  15   7  10   7  16  11  18  11  18  15   7  15   4  13  14  14  18
 6  14   7  13   5  12   4  13   8  13  11  14  10  16  19  14  17  10   9  19
15   3   5   9
```

The sorted array is:

```
 0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1   1   1   1   1
 1   1   1   1   1   1   1   2   2   2   2   2   2   2   2   2   2   2   2   2
 2   2   2   2   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
 3   3   3   3   3   3   3   3   3   3   4   4   4   4   4   4   4   4   4   4
 4   4   4   4   4   4   4   4   5   5   5   5   5   5   5   5   5   5   5   5
 5   5   5   5   5   6   6   6   6   6   6   6   6   6   6   6   6   6   6   6
 6   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7
 7   7   8   8   8   8   8   8   8   8   8   8   8   8   8   9   9   9   9   9
 9   9   9   9   9   9   9   9   9  10  10  10  10  10  10  10  10  10  10  10
10  10  10  10  10  10  10  10  10  10  11  11  11  11  11  11  11  11  11  11
11  11  11  12  12  12  12  12  12  12  12  13  13  13  13  13  13  13  13  13
13  13  13  13  13  13  13  13  13  13  14  14  14  14  14  14  14  14  14  14
14  14  14  14  14  14  14  14  15  15  15  15  15  15  15  15  15  15  15  15
15  16  16  16  16  16  16  16  16  16  16  16  16  16  17  17  17  17  17  17  17
17  17  17  17  17  17  17  17  17  17  18  18  18  18  18  18  18  18  18  18
18  18  18  18  18  18  18  19  19  19  19  19  19  19  19  19  19  19  19  19
19  19  19  19  19  19  19  19  20  20  20  20  20  20  20  20  20  20  20  20
20  20  20  20
```

This array contains 344 elements. The median is 10.

Response	Frequency	Histogram
		1 1 2 2 3
		1 5 0 5 0 5 0
0	13	*************
1	14	**************
2	17	*****************
3	26	**************************
4	18	******************
5	17	*****************
6	16	****************
7	21	*********************
8	13	*************
9	14	**************
10	21	*********************
11	13	*************
12	8	********
13	19	*******************
14	18	******************
15	13	*************
16	12	************
17	17	*****************
18	17	*****************
19	21	*********************
20	16	****************

The mode is the most frequent value. For this run the mode(s) is/are: 3

2. Write a program that will randomly generate 30 numbers between 1 and 40 inclusive. Store these numbers into an array as they are generated. Call this array c. Also, as the numbers are generated, store each number into a separate array only if it is not a duplicate of a number already read. In other words, this second array will only contain numbers that appeared once. I call this array a. Then call a sorting routine of your choice that will sort this second array (array a). Display your results as shown below.

The program randomly generates 30 integers between 1 and 40 inclusive.

The values generated are:
30 19 12 31 27 40 32 28 33 28 5 15 15 3 6 3 20 35 10 38 16 34 17 31 40 24 28 19 26 21

There are 23 nonduplicate values:
30 19 12 31 27 40 32 28 33 5 15 3 6 20 35 10 38 16 34 17 24 26 21

The sorted nonduplicate values are:
3 5 6 10 12 15 16 17 19 20 21 24 26 27 28 30 31 32 33 34 35 38 40

3. Write a program that sorts the following words in alphabetical order using Selection Sort. Display the correct sequence of words for each value of outer after its inner loop and swap have been completed:

programs java always very well document

4.

```
public class SortPhoneList
{
//Creates an array of Contact objects, sorts them, then prints them
    public static void main (String[] args)
    {
       Contact[] friends = new Contact[19];
       friends[0] = new Contact("Paige", "Brunda", "IM-L8-BLM-TIA");
       friends[1] = new Contact("Tia", "Enevoldsen", "I-HV-BUM-KNEE");
       friends[2] = new Contact("Mimi", "Bury", "ILL-B-SCK-TDY");
       friends[3] = new Contact("Ryan", "Panella", "IM-COOL-GEEK");
       friends[4] = new Contact("Garret", "Heiser", "SOCCR-SKIPR");
       friends[5] = new Contact("Tim", "Lange", "I-AM-TO-SXY-4U");
       friends[6] = new Contact("Grant", "Jackson", "I-PLAY-TRMPT");
       friends[7] = new Contact("Jason", "Wittman", "IM-A-FOOL-N-LV");
       friends[8] = new Contact("Stephanie", "Badum", "BLM-D-TRFFIC");
       friends[9] = new Contact("Emily", "Sin", "I-CUT-MY-HAIR");
       friends[10] = new Contact("Rob", "Joslin","IM-MR-KOOL-2-U");
       friends[11] = new Contact("Michael", "Puncel", "I-LIKE-TO-RUN");
       friends[12] = new Contact("Gabe", "Santos", "NVR-N-TME-2-CLS");
       friends[13] = new Contact("Murphy", "Hitchcock", "LV-MY-SCOOTR");
       friends[14] = new Contact("Heide", "Rivera", "MY-NAME-S-BOB");
       friends[15] = new Contact("Josh" , "Glodich", "I-LV-2-ANNOY-U");
       friends[16] = new Contact("Evan", "Fullerton", "I-LV-THS-CLSS");
       friends[17] = new Contact("Matt", "Bissonnette", "IM-TRACK-STR");
       friends[18] = new Contact("Josh", "Lithgow", "CHIEF-HOPPR");

        Sorts.alphabetizeArray(friends);
        System.out.println("NAME              PHONE" + "\n" +
                  "---------------------------");
        for (int index = 0; index < friends.length; index++)
        System.out.println( friends[index] );
    }
}

public class Contact implements Comparable
//remember that when a class implements Comparable it has to provide a
//compareTo method
{
  private String firstName, lastName, phone;

  public Contact( String first, String last, String telephone )
  {
        firstName = first;
        lastName = last;
        phone = telephone;
  }

  //this method concatenates and returns a description of this contact
  //as one String.
  public String toString( )
  {
        return lastName + ", " + firstName + "\t" + phone;
  }
```

```
  //this method uses String methods equals and compareTo.
  //Remember that compareTo returns a value of 0 if the argument string is equal to
  //this string; a value less than 0 if this string is lexicographically less than
  //the string argument; and a value greater than 0 if this string is
  //lexicographically greater than the string argument.
  public int compareTo( Object other )
  {
    int result;
    if( lastName.equals( ((Contact)other).getLastName() ) )
    result = this.firstName.compareTo( ((Contact)other).getFirstName() );
    else result = this.lastName.compareTo( ((Contact)other).getLastName() );
    return result;
  }

  public String getLastName()
  { return lastName;  }

  public String getFirstName()
  { return firstName; }
}

public class Sorts
{
  //Comparable[] object means that object is an array of objects that implement
  //Comparable
  public static void alphabetizeArray( Comparable[] objects ) //insertionSort
  {
    .
    .
    .
  }
}
```

OUTPUT (Note that I changed the phone list but didn't update output below)

NAME	PHONE
Badum, Stephanie	BLM-D-TRFFIC
Bissonnette, Matt	BISS-NN-ETTE
Brunda, Paige	IM-L8-BLM-TIA
Bury, Mimi	ILL-B-SCK-TDY
Enevoldsen, Tia	IM-ALWAYS-L8
Fullerton, Evan	I-LV-THS-CLSS
Glodich, Josh	I-LV-2-ANNOY-U
Heiser, Garret	SOCCR-SKIPR
Hitchcock, Murphy	LV-MY-SCOOTR
Jackson, Grant	I-PLAY-TRMPT
Joslin, Rob	IM-MR-KOOL-2-U
Lange, Tim	I-AM-TO-SXY-4U
Panella, Ryan	IM-COOL-GEEK
Puncel, Michael	I-LIKE-TO-RUN
Rivera, Heide	MY-NAME-S-BOB
Santos, Gabe	NVR-N-TME-2-CLS
Sin, Emily	I-CUT-MY-HAIR
Wittman, Jason	IM-A-FOOL-N-LV

5. Complete the following program so that it produces the output shown below.

```
import java.util.ArrayList;
public class DestinysChild
{
    public static void main( String[] args )
    {
      String teenIdol = "Mr. Dominguez";
      String sucks = "Kelly";
       ArrayList<String> band = new ArrayList<String>();
       band.add("Michelle");
       band.add("Kelly");
       band.add("Beyonce");
       band.add("Farrah");
       System.out.println("Destiny's Child: " + "\t" + band);

      //on the section below (the section you have to write) I used
      //the following ArrayList methods: get, indexOf, remove, add.
      //You'll definitely have to use some of these methods but you decide
      //which and how you use them.
        .
        ?
        .

 System.out.println("\nThe fans have decided that " + sackedMember +
            " is not cutting it and " + "\n" +
            "have sacked her in favor of up-and-coming new teen idol, " +
            band.get(band.indexOf(teenIdol)) + "." + "\n\n" +
            "The new Destiny's Child is:");

       for( int k = 0; k < band.size(); k++)
         System.out.println("\t\t\t\t" + band.get(k));
        //System.out.println("Size of the band: " + band.size());
    }
}
```

OUTPUT:

Destiny's Child: [Michelle, Kelly, Beyonce, Farrah]

The fans have decided that Kelly is not cutting it and
have sacked her in favor of up-and-coming new teen idol, Mr. Dominguez.

The new Destiny's Child is:

Michelle
Beyonce
Farrah
Mr. Dominguez

6. Write a driver program that:

- **creates an `ArrayList` of `Friend` objects (call it `myFriends`). Your program will read the names of friends, along with their corresponding phone numbers, from an already existing external text file (call it "*names.txt*". You can use the names and phone numbers in problem 4 but make sure the names in this file (i.e. "*names.txt*") are in random order and the driver program does not know ahead of time how many names there are). Each name and phone number is used to construct a `Friend` object (you have to write a separate `Friend` class). Each `Friend` object is then stored into the `myFriends` `ArrayList`.**

- **creates a second external file (call it "*namesAlphabetical.txt*") with the names in `myFriends` arranged alphabetically. Do this by calling method `alphabetizeNames` (this is a method you write) that, in effect, creates an `ArrayList` with the names in `ArrayList` `myFriends` alphabetized. This new `ArrayList` is then printed out to an external file. You can put this method inside your driver program; then, of course, you would call `alphabetizeNames` from inside `public-static-void-main`. You can use basically the same alphabetizing method you used in problem 4 modified to work with an ArrayList instead of an array.**

- **prompts the user for a name to search using a method called `nameSearch` (this is a method you write). Method `nameSearch` searches the second `ArrayList` (the one with the alphabetized list) for a given name. If the name is found, method `nameSearch` returns the `Friend` object with the sought-after name and phone number. If the name is not found then the method returns a message indicating so (think of a way to do this).**

The driver program will prompt you for another name to search or quit.