CPS 210 Lab 12

TIPS:

- Type all the programs from scratch each time. If you use copy and paste, it will take much longer to memorize code that is used often.
- Don't be worried about getting it on the first try; a lot of programming comes down to trial and error.
- Remember to save and recompile **every time** you make changes to your program.
- If you get any errors when you compile, go back and make sure you have proper **capitalization** and all of your **;** and matching **{ }**. If you can't find the error, **ask for help**.

PROBLEMS:

1. Create an Integer ArrayList.
   a. Write a method to create and fill the ArrayList with n random integers between [25,100).
   b. Write a method to compute the average of the ArrayList.
   c. Write a method to remove all the values that are greater than the average. Use your method from 1b.
      - **Hint:** index values shift when you remove things from an ArrayList. What should you do to i in your for loop every time you remove a value?
   d. Call each method to test parts a, b and c. Print the ArrayList after 1a and 1c.

2. Write a program to read in doubles until the user enters 0. Store the inputs in an ArrayList. Print the ArrayList.

a. Then, write a method that finds the maximum double from the ArrayList (remember how to find the min and max values from an array except apply it to ArrayLists). Call the method and display your result.

3. Create a String variable. Print the result for each:

   a. Get the length of the variable

   b. Get the first letter of the variable

   c. Get the last letter of the variable

   d. Print the String one character per line

   e. Print the String backwards

   f. Make the String all upper case using the toUpperCase() method.

   g. Make the String all lower case using the toLowerCase() method.

4. A valid password has:

   - A length of at least 8 characters
   - At least 1 uppercase letter
   - At least 2 numbers
   - At least 1 symbol

   Write a method that determines if a password is valid or not. If it is, return true, if not, return false. Use Scanner in the main header to get a word from the user to test out your method. If the password is not valid, make the user try again. They should keep trying until they enter a valid password (use a while loop).

- **HINT:** For your method, use “count” variables as you go through your String. **For example**: If a character is a number, add 1 to your count for numbers. If all of your counts meet the requirements, return true. Else, false.

- **Another HINT:** To see if a character is a symbol, you can use the isLetterOrDigit method from the Character class. **For example:** statement **if (!Character.isLetterOrDigit(*char*)) { …**

  The **!** In front of the method means **not** — “if char is **not** a number or digit…(must be a symbol!)”

  Sample run:

```
Password: sfsgs
Password must be at least 8 characters long, contain at least 1 uppercase character, 1 symbol, and 2 numbers.
Try again: hefdsa
Password must be at least 8 characters long, contain at least 1 uppercase character, 1 symbol, and 2 numbers.
Try again: Ds45!@fddfg
Ds45!@fddfg is valid.
```