PROGRAM 11A, 11B
Sorts and Searches In Arrays
And The String Thing
AP Computer Science
Mr. Clausen

PROGRAM 11A I'm_Sort_of_Searching (20 more points now, 60 points when all parts are finished)

Continue this program from program 9A. We won't finish this exercise until we finish Chapter 12.

Your program should set up the main menu as shown below:

Main Menu for Sorts and Searches

1. Generate Random Numbers For the Original Array
2. Copy the Original Array
3. Sequential (Linear) Search
4. Binary Search (Iteration: The Array Must Be Sorted)
5. Binary Search (Using Recursion)
6. Selection Sort
7. ~~Bubble Sort~~
8. Insertion Sort
9. ~~Quick Sort~~

M. Merge Sort
0. Display the numbers
I. Insert a number into the Array
D. Delete a number from the Array
Q. Quit the program

Enter your Choice:

For program 11A we are implementing the choices for Selection Sort, ~~Bubble Sort~~, Insertion Sort, Binary Search (Using Iteration), Insertion, and Deletion.

The insertion (not Insertion Sort) should make sure that there is room in the physical size of the array, ask the user at what index number they would like to insert the element, check to see if this is in the range of the logical size + 1, ask the user what number to insert, and insert this number without losing any of the existing elements (i.e. make room for this new number by moving other numbers first), and don't forget to increment the logical size.

The Deletion should ask the user for the index number of the element that they would like to delete, check to see if this is in the range of the physical and logical sizes, delete this element, move the elements up so there is not an empty element, and decrease the logical size.

Each time you sort, time how long it takes with a stopwatch. After each sort, copy the duplicate array back into the original array, so that we will be sorting the same numbers each time for a better comparison of each sort.

Also in your menu should be choices for binary search using iteration (remember that a binary search assumes that the list of numbers is previously sorted). Ask the user to enter a number to search for from 1 to the logical size of the array. Search through the array for that number, keeping track of how long it takes with each search. Make sure your program takes care of the case where the number is not found in the list. Also make sure to consider the case where the number occurs more than once (list the first index where the number occurs for the binary searches).

When displaying the contents of the array **"list",** display the numbers using rows and 10 columns so that we can see more numbers on the monitor screen. You will need an "if" statement to force a new line after every 10 numbers. Format these numbers right justified with a width of 7, and make the "table" look nice.

Don't worry about classes for this program. **Write everything in one source code file named: LastNameFirstNameP11A.java. Use methods for each menu choice**. We will understand the reasons for this later (Ch 10), but start your method names with **public static** and then **void** or **int**, etc. The methods should be at the end of your program after the } that ends the **public static void main (String [] args)** method and before the final } that ends your **public class LastNameFirstNameP11A** class.

PROGRAM 11B String Thing (30 points)
This program should have a menu that follows:

Main Menu for String Thing

1. Names
2. Palindromes
Q. Quit the program

Enter your Choice:

For the Names portion of the program:
This part of the program should ask the user for their full name: first name, one space, middle name, one space, and last name, then press enter. Your job is to separate this full name into three other separate variables: one for the first name, one for the middle name, and a third for the last name. Display these separate names on the screen. You will need to use the String Methods included in Java. Here's a hint, the spaces tell you where the first, middle and last names can be found. Display each name separately, and then display the last name, a comma and the first name.

For the palindromes portion of the program:
This program should ask the user for just one word. Then tell the user if the word is a palindrome or not (a palindrome is a word that is the same forwards and backwards, i.e. deed is a palindrome). This part of your program will practice using the concatenation and the **equals** methods in the String class. Hint: you can also use the indexing feature of String to help compare the "string" a letter at a time and process this inside a "loop". Use one String instance for the original word, and another instance for the word backwards. Don't forget to use methods.

Don't worry about classes for this program. Write everything in one source code file named: LastNameFirstNameP11B.java as described for program 11A.