AP Computer Science Java
Mr. Clausen
Program 9A, 9B

PROGRAM 9A I'm_Sort_of_Searching (20 points now, 60 points when all parts are finished)

The purpose of this project is to set up a program that will practice using arrays and the sorts and searches necessary for the AP Exam. We won't finish this exercise until we finish Chapter 12. Don't worry about classes for this program. **Write everything in one source code file named: LastNameFirstNameP9A.java.**
DO NOT USE a break statement to escape from any type of loop in this class regardless of what the book says or models.

1) Type comments at the beginning of the program to display your name and other information just like those used for program 2A.

2) Import the packages:
   import java.util.Scanner;
   import java.util.Random;

3) Declare the class name in the format LastNameFirstNameP9A. Don't forget that the filename needs to be the same when you save your program. This would be a good time to save your program if you haven't done so already, LastNameFirstNameP9A.java.

4) Declare the main method:
   public static void main(String [] args)

5) Next, declare the constant(s) necessary for this program. Declare a constant of type **int** for PHYSICAL_SIZE = 1000.

6) Instantiate an object of the Scanner class in order to use input from the keyboard. Also instantiate an object of the Random class in order to use random numbers. Use the statements:
   Scanner reader = new Scanner(System.in);
   Random generator = new Random( );

7) Declare all of the variables necessary for this program. You will need an integer for logicalSize, a string for menuChoice, declare and instantiate two arrays: origList and list to be arrays of type integer and size using the constant PHYSICAL_SIZE. Make sure that you use descriptive variable names to create readable, self-documenting code.

8) Type the following comment:
   //-------------------------Display My Information-------------------------

Follow this comment with println statements to display your name and period output just like those used for program 2A. Do not repeat this information each time the program repeats.

9) This client should be a menu driven program as described in Section 7.3, with the following menu choices:

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:

10) Of course, the program will need to be surrounded by a **while loop this time (NOT a do…while loop like we used in Program 7)** to see if the user wants to repeat the program. **The program should continue repeating until the user enters a "Q" or a "q".** Do not repeat displaying your information every time the program repeats

11) Instead of sections for input, calculations, and output, we will need a section for each of the menu choices listed above. You will need extended if statements to cover each of the menu choices listed above. For choice "Q", thank the user for using your program. Begin each section with a line of comments according to the menu choice.

12) For each portion of the program that is not yet implemented, have System.out.println statements saying that this feature of the program is not yet implemented.

13) **Use methods for each menu choice that is implemented**. We will understand the reasons for this later, 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 LastNameFirstNameP9A** class.

14) For menu choice 1, ask the user how many elements they would like to use for this array. Error trap this choice to make sure that the logicalSize is less than the PHYSICAL_SIZE and greater than zero. (NOTE: I will test your program with 10 or 100 elements)! Then, call the method to **fill this array with random numbers from 1 to the logical size of the array** (You will have to use the Random method included in Java).

15) Make a duplicate copy of this array (menu choice 2) including the random numbers, named **list**. **You will be doing all the sorts and searches on the array named list and NOT on the original array (origList).** You can test the program with 10 or 100 numbers, then when your program works, test the array with more numbers.

16) For now we will only fill in the details for the Sequential (Linear) Search and set up the menu for the program. We will add the other sorts and searches at a later date. For the Sequential (Linear) Search, we will count how many times the target number was found, and list the index numbers where the target was found.

17) 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.

18) Since our methods are **public static**, you will need to include another instantiation of the Scanner class in the LinearSearch method. You will also need to declare all of the local variables necessary to do the search.

Use blank lines and comment lines to separate each of the program sections listed in all of the steps above. When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "W" network mapping, in the Program 9A folder. **Turn in the file named LastNameFirstNameP9A.java, I don't need the ".class" file for this program.**

PROGRAM 9B
Two Dimensional Arrays
AP Computer Science
Mr. Clausen

PROGRAM 9B Two Dimensional Arrays (25 points)

This program is designed to give you practice with the two dimensional arrays. Declare two 2D arrays named matrix and transposed, both with 10 rows and 10 columns; this is the physical size of the arrays. The two dimensional array named matrix will store integers, and the other two-dimensional array named transposed, to contain the transposed values of matrix, also integers. Use primed while loops to error trap these numbers to make sure they are not negative numbers, zero, or larger than 10.

The transposition of a two-dimensional array is a new two-dimensional array with the row and column positions reversed. The transposition of a two-dimensional array called matrix which has M rows and N columns (an M x N two dimensional array: that's M by N) is an N x M two-dimensional array called transposed, with each element matrix [M] [N] being stored in transposed [N] [M].

This program will also test to see if the two-dimensional array is symmetric. A two-dimensional array is symmetric if it has the same number of rows and columns (called a "square" two-dimensional array) AND if each element of matrix [M] [N] is equal to each element of transposed [N] [M]. By the way, a symmetric two-dimensional array will be the same as it's transposed two-dimensional array.

Before the menu, have a section of code to initialize every element of both arrays to –1.

In the menu, ask the user how many rows and columns they would like in their two-dimensional array, this will be the logical size of the arrays. Below is what the program Menu should look like:

Main Menu for 2 Dimensional Arrays

1) Choose the number of rows and columns for the array
2) Fill the array with random integers.
3) Enter all of the elements
4) Test the matrix for symmetry
5) Transpose the matrix
6) View the elements in the original matrix
7) View the elements in the transposed matrix
Q) Quit the program

Enter your Choice:

As you can see, you will need to fill the two-dimensional array with random numbers (integers from 1 to row size times column size), allow the user to enter data values for **every** element in the two-dimensional array (so we can enter values that do give us a symmetric 2D array: you create a loop providing the index numbers for each row and column and ask the user for the elements.), test for symmetry, display the original two-dimensional array (matrix), and display the transposed two-dimensional array (transposed). Format the output in rows and columns and right justify each integer with a width of 5 spaces.

Let's look at examples of transposed and symmetric two-dimensional arrays…

Let's start with the following 2 x 3 two-dimensional array as our original two-dimensional array, named matrix.

matrix: a 2x3 two-dimensional array.

1	2	3
4	5	6

The transposed two-dimensional array named transposed, will be a 3 x 2 two-dimensional array with the row and column elements reversed. The element at index number [0] [0] will stay there, since reversing two zeros does not change the position of the element. The element at index number [0] [1] will be relocated to index position
[1] [0]. The element at index number [0] [2] will be relocated to index position [2] [0], etc.

transposed: a 3 x 2 two-dimensional array

1	4
2	5
3	6

A symmetric two-dimensional array must have the same number of rows and columns, and transposed must be equivalent to matrix. Here is an example of a symmetric two-dimensional array, use this as test data for your program.

matrix: a 3 x 3 symmetric two-dimensional array.

4	1	2
1	5	7
2	7	6

You can see that index numbers [0] [0], [1] [1], and [2] [2] will not move when we switch the row and column numbers (the diagonal going down from left to right).
The element "1" at index [0] [1] must be the same as the element at index [1] [0], the element "2" at index [0] [2] must be the same as the element at index [2] [0], etc. In other words, if you were to "fold" the two-dimensional array across the diagonal line, the elements must match each other. You are already familiar with graphs symmetric to a line from previous math classes, and you should be familiar with two-dimensional array from your Algebra 2 class.

Don't worry about classes for this program. Write everything in one source code file named: LastNameFirstNameP9B.java.