**Lab 11 arraySortTest & Test Recurs**

**Due 3/31/2022**

**Part 1: Sorting**

* Create a new project called arraySortTest.
* Implement our three sorting algorithms (bubble sort, selection sort, quick sort) as static methods. Use the code given in class, which are classic examples of the algorithms. Do not use any modifications of the algorithms. Include any helper methods that night be needed.
* In main(), initialize three different arrays (array1, array2, array3) of type int. Consider what would make them different if they are all of type int.
* In main(), sort each array using each sorting algorithm. Perform each test on each array in a separate run. You will have a total of 9 tests. Take a screenshot of each run, noting the run-time.
* Note: This test has two variables, the type of array, and the sorting algorithm. Otherwise, the test runs should be identical.

**Part 2: Testing Efficiency**

* In Lab 10 Recursion I, you wrote the pow() function and the fib() function using recursion.
* In the same lab file as those functions, write pow() and fib() functions that work iteratively. Note: The functions will have to have different names. Maybe powR() and powI().
* To each of the four functions, add a print statement that will show and calculate the number of processes each function required.
  + In recursive functions, a good place to do this right before the return statement
    - Process 1
    - Process 2
  + In the iterative functions, a good place is inside the loop. Same sort of output as above.
* Remember to test each using several input values.
* In addition to code and output, write a brief statement about which version of pow() and which version of fib() was more efficient based on the number of processes. Remember that **iterations** are less expensive than method calls.