Programming Assignment 10-3

In this exercise, you will create a sorting program based on BSTs, in the way that was described in the Lesson 10 slides, and you will compare its performance to the `MinSort` program; the performance test will be carried out in a test harness that has been provided for you.

To write your code so that it can be used in the test harness, copy the project `Sorting` (you will find this in your lab folder) to your local drive and import it into your workspace as a new Java project. You will add your new classes to the `sortroutines` package. In that package, you should place `MyBST.java` (from assignment 10-1) and also a new class called `BSTSort.java`, which you will create.

First, you will need to create two methods in MyBST.java like following –
// It takes as input an array and builds a BST tree from it.
public void insertAll(int[] array){…}

//It traverses the BST and returns all its elements in a sorted array
public int[] readIntoArray() {…}

Your class `BSTSort` should inherit from the class `Sorter` (which is in the runtime package of the `Sorting` project). When you inherit from `Sorter`, you will be required to override the `abstract` method `int[] sort(int[] arr).` Your code will accept any input array of `int`s, load them into an instance of MyBST (by calling the `insertAll` method you just created), and then, will use the `readIntoArray` method to obtain a return value, which will be the original array of `int`s, now in sorted order.

Remember that by autoboxing, Java will automatically convert between `int` and `Integer` types.

To compare your `BSTSort` program with `MinSort`, type the string "BSTSort" below "MinSort" in the text file `sorters_to_be_run.txt` (be careful not to change the location of this file in the project). Then run the class `SortTester`. `SortTester` will read the names of the classes specified in `sorters_to_be_run.txt`, and will (by reflection) create instances of each and run them through thousands of sorting tests. In the console window, you will see how well each sorter performed, from fastest to slowest.