Lab Programming Assignment #2

Due: Sunday, March 1, 2020 by 11:59 pm PST.

Points possible: 20

Submission Requirements:

- You can work in teams up to 3. If working in teams, only one person should submit the required file. However, all team member names need to be listed within the Java file, as a comment at the top.
- You should submit just one file, and that file's name should be:
  - **MergeSort.java**
- Submitting a file with any other name will get you points deducted. Do not zip the file either.
- Because the Java file name should be what's listed above, the top level class name within it should also be named **MergeSort**
- Within the java file, include your name (at the top as a comment). If working in a team, the full names of all team members must also be within the Java file (at the top as a comment).
- Submit the Java file to BeachBoard. No email submissions will be accepted.
- The project must be in the Java programming language.
- I should be able to both pass in arguments and run your program via the command line.

Program Requirements:

- Implement merge sort using arrays. It must output the following:
  1. The input (unsorted sequence).
  2. The output (sorted sequence).
  3. The number of comparisons made.

- Use the same merge sort algorithm as the book (page 31 and 34).
- It must be able to sort integers, as I will only be testing with integers.
- It must be able to support up to 20 integers.
- Since this is merge sort, your program must use recursion.
- When calculating the midpoint, it should round down to the nearest integer. E.g. 1/2 = 0, 2/2 = 1, 3/2=1, 4/2=2, etc. This is how it's done in the book as well.
- Your algorithm should also be able to support just one element as your input, as well as duplicates.
- Executing your program should be done via the command line. If I cannot both run your program and pass in my test cases through the command line, you will get a lot of points deducted, up to and including receiving zero credit.

- After executing your program via the command line, I should see the following (the below input and output array values are just examples):

```
>java MergeSort 4 5 1 7 3 3 8
Unsorted Array
4 5 1 7 3 3 8

Sorted array
1 3 3 4 5 7 8

Number of comparisons: 14
```

- Here are some more examples:

```
>java MergeSort 1 2 1
Unsorted Array
1 2 1

Sorted array
1 1 2

Number of comparisons: 3

>java MergeSort 1
Unsorted Array
1

Sorted array
1

Number of comparisons: 0

>java MergeSort 5 5 5 5
Unsorted Array
5 5 5 5

Sorted array
5 5 5 5

Number of comparisons: 4

>java MergeSort 1 2 3 4 5 6 7
Unsorted Array
1 2 3 4 5 6 7

Sorted array
1 2 3 4 5 6 7

Number of comparisons: 11
```

Note that in Java, array index starts at 0. So when doing the above examples either via your program or by hand, ensure your array index starts at 0 and you will see the above comparison counts are accurate. Your array index within your code must start at zero so that your comparison counts will match with my test cases.

Comment your code enough so that in case it doesn't work, I can see what you're trying to do and possibly give you partial credit.

Failure to follow the above requirements may result in substantial points deduction, up to and including receiving zero credit. I recommend you read the instructions more than once to ensure your program meets the requirements.

Grading Guidelines:

- Does the program meet the requested requirements/criteria?
- Are the submission instructions followed?
- Is your code properly commented?
- Does your code compile?
- Does your code pass my test cases?