

Topic: Parallel Sorting with Futures and Fork Join **[Total Marks: 100]**

In this Assignment you are required to complete a Java project by implementing three different versions of the Merge Sort algorithm . Your merge sort should implement the following interface

```
public interface Sorter {  
    <T extends Comparable<? super T>> List<T> sort(List<T> list);  
}
```

Note: ISequentialSorter (an insertion sorter) is already implemented as an example.

---

**Task1****[60 Marks]****part a:[10 marks]**

Implement the MSequentialSorter class so that it uses merge sort and a sequential algorithm to sort the data in parallel.

**part b:[25 marks]**

Implement the MParallelSorter1 class so that it uses merge sort and futures to sort data.

You are allowed to delegate to an MSequentialSorter or an ISequentialSorter for the cases with less than 20 elements.

**part c:[25 marks]**

Implement the MParallelSorter1 class so that it uses merge sort and the forkJoin framework to sort the data in parallel.

You are allowed to delegate to an MSequentialSorter or an ISequentialSorter for the cases with less than 20 elements.

You should now be able to pass all the tests provided in all the 3 JUnit test suites of the project.

**Note: DO NOT MODIFY THE JUNIT TEST FILES!** You will lose marks if you modify the provided JUnit test suites.

**Task 2****[15 Marks]**

Add another test suit, similar to the three provided test suites, but using another kind of comparable datastructure to test the merge sort implementations.

**Task 3****[10 Marks]**

---

In the documentation of each sorting method that you implemented in task 1, write a paragraph, which should be no more than 6 sentences, showing:

- The benefit of the algorithm with respect to the others implemented in this assignment. (Guide: 5 sentences)
- Explaining what you learned from implementing merge sort in such way (Guide: 1 Sentence)

**Task 4****[15 Marks]**

---

The file TestPerformance shows how to check the performance of the various solutions.

- Complete the file to handle the new datatype of Task 2.
- Write documentation for all the methods of TestPerformance.  
The goal here is to show that you understand the code of TestPerformance in great detail.
- Feel free to include discussions about the pros and cons of the alternative implementations for testing the performance.