Lehrstuhl für Computergestützte Modellierung und Simulation
TUM School of Engineering and Design
Technical University of Munich

Professional Software Engineering

Assignment 6

Deadline: 12^{th} of February

The aim of this assignment is to test your understanding of Generics, Functional Programming, extension methods and LINQ.

Once you are done with the assignment, make sure you zip your folder before uploading. Once again only one submission is required per group along with a text file containing the amount of contribution by each member.

1 Background

Quicksort is a divide-and-conquer type sorting algorithm. It is one of the fastest algorithms to sort through the randomized data. It works by selecting a random 'pivot' element from the array and then partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot. The sub-arrays are then sorted recursively. In the end, the sorted sub-array with elements less than or equal to pivot is concatanated with the pivot element and the second sub-array, to get the final array.

Further reading: `https://en.wikipedia.org/wiki/Quicksort`

2 Programming

Write a C# program which has the following:

1. An extension method that uses the Quicksort algorithm described in section 1 to sort an integer list, *List<int>* (return a new list rather than sorting the original one in place)
2. A generalized implementation of (1) that takes a generic *List<T>* and a *Comparison<T>* [1] delegate
3. An extension method to print elements of a generic list to the console
4. A main function to test the methods implemented above

Note: You are required to use LINQ while implementing the sorting algorithm but are **not** allowed to use predefined *Sort()*. Use *<T>* as *<double>* to test the generalized implementation.

Some useful tips:

- Use the first element of the list as the pivot element
- Look up the use of *ToList()* to convert an IEnumerable linq query to List
- Pay attention to the recursion in the algorithm
- You'll need to pass an integer function with return values -1, 0 or 1 to the delegate in order to test your general implementation. These return values are based on the comparison criteria [1]

[1] Further information: `https://learn.microsoft.com/en-us/dotnet/api/system.comparison-1?view=net-7.0`