Home Work 3

- Any coding platform and programming language is allowed
- Code working Insertion Sort, Quick Sort, Merge Sort and Heap Sort algorithms as methods
    - You can use Insertion Sort, Merge Sort and Quick Sort algorithms you have coded in your Home Work 1 and Home Work 2
- For Quick Sort algorithms, you need to use 3 algorithms we have put on GitHub at week 5's lesson. The link is as follows >
    - https://github.com/FurkanGozukara/Analysis-of-Algorithms-2019/blob/master/source%20codes/week%205%20quick%20sort/week%205%20quick%20sort/Program.cs
    - Quick sort algorithm names are Quick_Sort_Algo_1, Quick_Sort_Algo_2 and Qucik_Sort_Median_3
    - If you use other than C# programming language, convert provided Quick Sort algorithms on GitHub to your own programming language. It should almost effortlessly work on C++ and Java. For Python and PHP etc. it should be easy to code
    - The difference between all these 3 variations of Quick Sort algorithms is, they use different pivot selection thus different optimization
- Make sure that your sorting methods are working properly and correctly
- Calculate method run times as milliseconds

Main method

- Generate an array or a list that contains 100,000 random integers (between 0 and 2.1 billion). If you use C++ and Visual Studio, you have to increase stack size otherwise it would give stack overflow error
    - *How to increase stack size > PROJECT->Properties->Configuration Properties->Linker->System->Stack Reserve Size=4194304*
    - *If you get **stack overflow** error in C#, it can be increased by wrapping your methods inside a new thread. An example code is provided on GitHub (source codes / week 7 increase csharp stack size) with the following link > https://github.com/FurkanGozukara/Analysis-of-Algorithms-2019/blob/master/source%20codes/week%207%20increase%20csharp%20stack%20size/week%207%20increase%20csharp%20stack%20size/Program.cs*
- This first generated list will be the seed list and it won't be given to the methods
- Write this list to a text file named as "seed.txt"
- Generate a temporary list or array
- Copy seed list to into this generated temporary list or array whatever you use
- Write temporary list to a text file named as "temp_as_seed.txt"
- Define a stopwatch/timer according to your programming language
- Start timer
- Call Quick_Sort_Algo_1 on temporary list. **Make sure that the temporary list itself is sorted. If it returns another list, assign it to the temporary list so it would be sorted**

- End timer
- Calculate method quick sort time and print to the screen with appropriate description
- Write temporary list to a text file named as "Quick_Sort_Algo_1_temp_1.txt" //this should have a sorted list
- Start timer again
- Call Quick_Sort_Algo_1 on temporary list again (this time we are sorting a sorted list)
- End timer
- Calculate time and print to the screen with appropriate description again
- Write temporary list to a text file named as "Quick_Sort_Algo_1_temp2.txt" //this should have sorted list
- Reverse sort temporary list. So this temporary list will now have reverse sorted list
- Write temporary list to a text file named as "Quick_Sort_Algo_1_temp3.txt" //this should have reverse order sorted list
- Start timer again
- Call Quick_Sort_Algo_1 on temporary list again (this time we are sorting a reverse sorted list)
- End timer
- Calculate time and print to the screen with correct definition as Quick_Sort_Algo_1 call on reverse sorted array
- Write temporary list to a text file named as "Quick_Sort_Algo_1_temp4.txt" //this should have sorted list
- This time change last element of the sorted temp list into a new random integer. So if your list is 100,000 integers, the 99,999th integer of the list is

changed into a new random integer. Then write this temporary list into a file named as “Quick_Sort_Algo_1_temp5.txt”

- Start timer again
- Call Quick_Sort_Algo_1 on temporary list again (this time we are emulating a dynamic list sorting because we can consider the last element of the list as a streaming data incoming 1 by 1)
- End timer
- Calculate time and print to the screen with correct definition as Quick_Sort_Algo_1 call on dynamic list sorting
- Copy seed list into the temporary list again
- Write temporary list to a text file named as “temp_array_seed_2.txt” // this should have same unsorted list as seed you have generated in the beginning
- **Repeat the same steps for Quick_Sort_Algo_2, Qucik_Sort_Median_3, Insertion Sort, Merge Sort and Heap Sort as well. So all of the algorithms performances can be compared on unsorted random array/list, sorted array/list, reverse order sorted array/list and dynamic sorted array/list**
- When repeating the steps, don’t forget to change txt file names according to the other Quick Sort Algorithms, Insertion, Merge and Heap Sort algorithms names

Home Work Delivery

- The software you have coded will be checked and evaluated at the week 12’s lesson (26.04.2019) on your computer or in a lab’s computer

- So you need to bring your laptop to show or make your software work on the lab's computer and there we can check
- I ask questions about your coding
- **Prepare a report about your software coding and explain each line of code clearly**
- Also please RAR(Winrar) or ZIP(Winzip) your software (delete debug and obj folder in C# otherwise it won't allow to be attached to email) project and email to furkan.gozukara@toros.edu.tr with including your full name and your student number
- I will individually and carefully check everyone's project source code so do not try to cheat, get from your friends. Otherwise you will get very low score (both the code giving person and the cheater)
- If you fail to deliver your project at the (26.04.2019), each delayed day will cause you a 1 points lower score up to 8 points out of 10
- Before that time, you can bring your homework and show me at my room A015
- Also you can ask any questions about your homework by coming to my room or from email