CSE201 Advanced Programming

Assignment 3

Release date: 27/11/2022
Due by: 02/12/2022

Syllabus : Multithreading, Generics, I/O

This assignment is a take-home lab assignment. No extensions whatsoever will be provided. Any submission after the deadline will not be evaluated. If there is any ambiguity or inconsistency in a question, please seek clarification from the teaching staff. Please read the entire text below very carefully before starting its implementation.

Plagiarism: All submitted lab assignments are expected to be the result of your individual effort. You should never misrepresent someone else's work as your own. In case any plagiarism case is detected, it will be dealt as per IIITD plagiarism policy and without any relaxations:

https://www.iiitd.ac.in/sites/default/files/docs/education/AcademicDishonesty.pdf Please note that you are not allowed to discuss the lab assignment's design/solution (e.g. classroom page discussions, etc.). Anyone who is found doing this will be treated as a plagiarism case. No Excuses!

NOTE that we won't respond to any query which we feel can be avoided after reading the problem description carefully. We will ONLY respond to valid questions. Make sure you ask all your doubts in advance and not at the last minute.

Submission Guidelines

- you have to submit a pom.xml (build) file and a readme.txt for each part
- the readme should contain steps to run your code and get the required results as mentioned in each of the parts above
- the TAs should be able to do this just from the readme, without any other assistance from you, the student
- in case there is any ambiguity in running the codes, or if there are compilation errors during the maven build, you will be awarded ZERO marks. Therefore, check your build files thoroughly before submitting.
- note that since there are a total of 3 assignments in the course with an n-1 policy, there is a choice to skip this assignment depending on your score in the previous two
- There will be no extension to the deadline. However, there is a late submission policy. For every 6 hours, 5% will be deducted as a penalty. So there will be a 20% deduction each day and you can submit up to 5 days late.

The aim of this assignment is to acquaint you with the concepts of parallelization, concurrency and multithreading using Java. This assignment contains two parts -

Part A

Congratulations on building the IIITD Placement Cell management portal “Future Builder", which was made in Assignment-1!!!!! . However, the number of registered students has grown manifold, and some functionalities are working too slow. IIITD Placement cell wants the functionality of arranging students in the descending order of their CGPA. This functionality will be making use of **Odd Even Transposition Sort (search what this is)**

You have to do the following:

- Compare the execution time with and without parallelization by varying the CGPA of the students (number of students should be [1, 10, 100, 1000, 10000, 100000, 1000000])
- the CGPA of students should be randomly generated (ensure it’s between 0.0 and 10.0)

Part B

Implement a recursive balanced binary tree package with and without parallelization. You are allowed to use Java Collection Framework in this part.

The number of input nodes ‘N’ should be [10, 1000, 10^6], and total number of threads ‘T’ should be {2, 4} i.e. you have to show 3*2 = 6 simulations of your implementation for the parallelization variant and 3 simulations for different values of ‘N’ for the non-parallelization variant, thus making the total simulations as 9 (6 for parallelization and 3 otherwise). For the elements of the tree you can choose integers randomly from [-10^9, 10^9] and create the balanced binary tree. Ensure that for any given ‘N’ and ‘T’, the elements in the tree should be the same for both the cases for fair comparison.

For each combination of ‘N’, ‘T’ and the implementation technique (parallel or not) (total 9), you have to report -

- time taken to construct the tree
- height of the tree
- time taken to search for an element in the constructed tree (note that for any given ‘N’ and ‘T’, the element should be the same for fair comparison across both variants of parallelization)