OOPS LAB ASSIGNMENT 10 Date – 07/11/2023

Roll No - 21BCS043

Title: Exploring Multithreading in Java through the Application of Merge Sort

Introduction: Multithreading is a powerful concept in Java that allows concurrent execution of multiple threads, enhancing program performance and responsiveness. This report delves into my learning experience with multithreading, applying it to the classical sorting algorithm, Merge Sort.

Multithreading in Java:

Multithreading allows the simultaneous execution of multiple threads, enabling parallelism and concurrency. In Java, it is achieved by extending the **Thread** class or implementing the **Runnable** interface.

Merge Sort:

Merge Sort is a divide-and-conquer algorithm for sorting arrays. It recursively divides the array into two halves, sorts each half, and then merges the sorted halves. Merge Sort is well-suited for parallelization due to its divide-and-conquer nature.

Thread Class and Runnable Interface:

In the context of multithreading, I explored both extending the **Thread** class and implementing the **Runnable** interface. Extending **Thread** provides a convenient way to create threads, while implementing **Runnable** allows for more flexibility and code reuse.

Benefits and Challenges:

Performance Improvement:

Multithreading in the context of Merge Sort provided performance improvements, especially for large arrays. The parallel execution of sorting tasks reduced overall execution time.

Challenges in Synchronization:
Ensuring proper synchronization and avoiding race conditions posed challenges. Careful consideration was required when accessing shared data structures to maintain data consistency.
Load Balancing:
Efficient load balancing was essential to maximize the benefits of multithreading. Balancing the workload among threads ensured that each thread contributed meaningfully to the sorting process.