Assignment - 05
CS341: Operating System Lab

General Instruction

- Markings will be based on the correctness and soundness of the outputs. Marks will be deducted in case of plagiarism.
- Proper indentation & appropriate comments (if necessary) are mandatory in the code.

The primary goal of this lab session is to familiarize students with the concept of threads in programming. By the end of this lab, students will be able to create multiple threads and establish effective communication between them.

1. Write a program in C that creates multiple threads to perform concurrent tasks. Implement a mechanism for communication and synchronization between these threads to ensure data consistency and proper execution order.
   (a) **Thread Creation:** Create a program that spawns three threads:
      - **Thread 1:** Generates a list of random integers and stores them in a shared buffer.
      - **Thread 2:** Sorts the integers in the shared buffer.
      - **Thread 3:** Calculates the sum and average of the integers in the sorted buffer and prints the results.

   (b) **Inter-thread Communication:** Implement appropriate synchronization mechanisms (e.g., mutexes, condition variables, or semaphores) to ensure:
      - Thread 2 only begins sorting after Thread 1 has completed generating and storing the integers.
      - Thread 3 only starts calculating and printing results after Thread 2 has finished sorting the buffer.

   (c) **Error Handlilng:**
      - Include error handling for potential issues, such as failures in thread creation or synchronization problems.

   (d) **Output:**
      - The program should display the unsorted list of integers, the sorted list, and the sum and average calculated by Thread 3.

2. Create a C program that implements a parallel merge sort algorithm on an array of integers. The program will utilize multiple threads to sort the array concurrently, ensuring thread safety during the merge process.
   (a) **User Input:**
      - Prompt the user to input the number of elements in the array and the elements themselves.

   (b) **Thread Creation:**
      - Create two main threads, each responsible for sorting one half of the array.
      - Each main thread will further spawn two sub-threads to sort smaller subarrays, resulting in a total of four levels of thread creation.

   (c) **Sorting Process:**
      - Each thread should print detailed messages indicating the indices of the array it is sorting and its completion status.

   (d) **Synchronization:**
      - Use mutexes to ensure thread-safe merging of the sorted subarrays.

   (e) **Merging:**
      - fter all threads have completed their tasks, merge the sorted halves of the array.

   (f) **Output:**
      - Display the fully sorted array after merging.

   (g) **Error Handling:**
      - Implement error handling for thread creation failures and other potential issues.

Deliverables:

- A complete C program for each question that meets the above requirements.
- Ensure the program is well-documented, with comments explaining the functionality of each section.