National University of Computer and Emerging Sciences, Lahore Campus

	Course:	**Operating Systems**	**Course Code:**	**CS 2006**
	Program:	**BSE (5A-5B)**	**Semester:**	**Fall 2024**
	Due Date	**24 November, 2024 11:59 PM**	**Total Marks:**	**75 marks**
	Type:	**Assignment 3**	**Page(s):**	**3**

Important Instructions:

1. Submit the solution in a folder as your roll number. Do not submit Zip Files else there will be a deduction of marks.
2. You are not allowed to copy solutions from other students. We will check your code for plagiarism using plagiarism checkers. If any sort of cheating is found, heavy penalties will be given to all students involved.
3. Late submission of your solution is not allowed.
4. Do not use Fstream and Fget in the questions.

Question 1: Parallel Data Exploration Quest **[Marks 30]**

Imagine a land of Data Science wonders where students are about to embark on an exciting journey. Their mission: to decode the secrets hidden in a massive text file filled with thousands of numbers. But beware, this file has tricky -9999999 values, like little guardians protecting its mysteries. To conquer this challenge, students must create a clever program in C/C++ that ventures through four stages: Loading, Cleaning, Analysis, and Reporting.

Requirements:

1. **Data Loading (Thread 1):** Begin your quest by creating a thread that reads the entire text file, gathering the raw data, including those mysterious -9999999 guardians. This stage sets the stage for your epic adventure.
2. **Cleaning (Thread 2):** The journey gets more exciting as you encounter obstacles in the form of -9999999 values. Create a thread to carefully remove these guardians, cleaning the data to reveal its true potential. Discover hidden patterns by sorting numbers and removing duplicates. Be cautious and wait if the Data Loading thread is still on its way.
3. **Analysis (Thread 3):** With the data now polished, start the analytical phase of your quest. This thread will unravel the mysteries, calculating the mean, median, and standard deviation of the numbers. Take a moment to survey your achievements, waiting if the Cleaning thread is still overcoming challenges.

4. **Reporting (Thread 4):** As the sun sets on your data-driven adventure, compile the insights you've gained into a comprehensive report. This thread will craft the tale of your triumphs, writing the results into a new text file. Ensure patience as you wait for the Analysis thread to complete its final calculations.

Embark on this Parallel Data Exploration Quest, where each thread takes you a step closer to uncovering the secrets within the data's mysterious depths. May your code be strong, and your discoveries remarkable!

Note: Use system calls only for file handling.

Question 2: Producer-Consumer with Shared Memory and Semaphores [30 marks]

Write a C++/C program that takes the name of a source file as a command-line argument. The program acts as a producer that reads 20 characters at a time from the file and writes them to a shared memory buffer created by the producer. The buffer has a capacity of 20 characters. A separate program acts as a consumer, attaching the shared memory to its address space, reading the 20 characters from the buffer, printing them on the screen, and waiting for the user to press the Enter key before continuing.

The producer can only write the next 20 characters from the file to the shared memory buffer once the consumer has read the previous 20 characters. To ensure proper synchronization between the two processes, semaphores must be used. You are only allowed to synchronize these processes using semaphores — no other synchronization methods are permitted.

When the producer has completely written the file's data to shared memory, it writes a $ symbol to indicate that the file is fully processed. After this, both programs should clean up all resources, including detaching and deleting the shared memory.

Assumptions:
● The file's size is guaranteed to be a multiple of 20 characters (e.g., 20, 40, 60, etc.).

Example Workflow:

Let's say the file contains 40 characters:

1. The producer writes 20 characters to shared memory and waits for the consumer to read them.
2. The consumer reads the 20 characters, prints them on the screen, and waits for the user to press Enter.
3. The producer then writes the next 20 characters and waits for the consumer to read them.
4. The consumer reads these 20 characters and prints them on the screen.
5. Once the end of the file is reached, the producer writes a $ symbol to shared memory to indicate that the entire file has been processed.
6. The producer detaches the shared memory from its address space, deletes the shared memory, and exits.
7. The consumer reads the $ symbol, which indicates the end of the file, detaches the shared memory, and exits

Question 3: Semaphore-based Cinema Ticket Booking System [15 marks]

Design a ticket booking system for a cinema that has multiple seat categories (VIP and Regular) and multiple customers trying to book seats concurrently. Implement semaphores to ensure that the booking process is synchronized, where only one customer can book a seat at a time. If no seats are available, customers must wait for a seat to be freed.

Requirements:

1. **Seat Categories:** There are two types of seats—VIP and Regular. The system should handle both types of seats concurrently, with VIP seats being prioritized for VIP customers.
2. **Semaphore Synchronization:**
   ○ Initialize the semaphores for VIP and Regular seats with their respective seat counts.
   ○ The VIP customers can only book from the VIP seats first.
3. **Booking Process:**
   ○ A mutex must be used to protect the critical section of booking seats (ensuring only one customer books at a time).
   ○ The system should track which seats have been booked and display the seat number along with the customer's ID.
4. **Customer Priority:** VIP customers should be allowed to book VIP seats first. If there are no VIP seats available, they should be able to book a regular seat.
5. **Completion:** Once all seats are booked, the system should end and release all resources (semaphores and mutexes).

Example:

- If the cinema has 5 seats in total, with 2 VIP seats and 3 Regular seats:
  ○ Customer 1 to 2 are VIP customers.
  ○ Customer 3 to 5 are regular customers.
  ○ Each customer attempts to book a seat, and if seats are available, they are assigned and displayed.

Note: You are not allowed to synchronize processes using any method other than semaphores and mutexes.