National University of Computer and Emerging Sciences, Lahore Campus

	Course:	**Operating Systems**	**Course Code:**	**CS 2006**
	Program:	**BSCS- 4E**	**Semester:**	**Spring 2025**
	Due Date	**23 March, 2025 11:59 PM**	**Total Marks:**	**40 marks**
	Instructor	**Mubashar Hussain**	**TA**	**L226598@lhr.nu.edu.pk**
	Type:	**Assignment 2**	**Questions:**	**3**

Important Instructions:

1. Submit separate code files for each question with your roll number (e.g., Q1_23L1011), along with screenshots of your terminal.
2. **Heavy penalties will be given to all students involved in plagiarism.**
3. Late submission of your solution is not allowed.
4. Your codes will be evaluated thoroughly so make sure you understand all the concepts well.
5. Viva maybe conducted hence it is advised to come prepared
6. In case of any queries, you can always reach out to me via email.

Note: Error handling carries marks throughout the assignment.

Question 1: Secure Forensic Data Analyzer 2.0 🔐 **[15 Marks]**

A cybersecurity research team is working on Forensic Data Analyzer 2.0 to analyze security logs. In Forensic Data Analyzer 1.0, they used system calls such as read() and write() to access files. However, in version 2.0, they are **restricted** from using read(), write(), or any other file manipulation methods (like fgets(), fprintf(), etc.). They may **only** use open() and close() if necessary.

Roles & Responsibilities:

1. **Evidence Collector (Person 1):**
   - Extracts the content of **forensic_logs.txt without using read()**.
   - Ensures the logs are transferred securely to the next stage of analysis.
2. **Threat Analyst (Person 2):**
   - Identifies and filters out log entries containing suspicious activity indicators (such as the keyword "**ALERT**").
   - Passes the cleaned log data forward for further processing.

3. **Report Generator (Person 3):**
    - **Removes duplicate** log entries to avoid redundancy.
    - Stores the final sanitized report in **forensic_report.txt without using write()**.

Hints:

- Use fork, pipe, dup2 and execvp.
- How can you extract file content without read()?
- What command-line can filter specific words from a file?
- Which command helps in removing duplicates efficiently?
- What alternative methods can be used to write data to a file without write()?

Note: In addition to implementing the solution in C++/C, you must also provide the exact terminal command that can achieve the same result as a comment in your code.

Sample Input (forensic_logs.txt)

```
ALERT: Unauthorized access detected in sector 5
System reboot initiated by Admin
ALERT: Multiple failed login attempts from IP 192.168.1.100
File integrity check completed, no anomalies found
ALERT: Suspicious file transfer detected to external device
ALERT: Unauthorized access detected in sector 5
Network traffic anomaly detected, investigating further
ALERT: Multiple failed login attempts from IP 192.168.1.100
Scheduled backup completed successfully
```

Expected Output (forensic_report.txt):

```
ALERT: Unauthorized access detected in sector 5
ALERT: Multiple failed login attempts from IP 192.168.1.100
ALERT: Suspicious file transfer detected to external device
```

Question 2: Library Management System 📚 **[15 Marks]**

Problem Statement:

You will implement a server-client model using **FIFO pipes**. The server maintains a **book.txt** which contains information about books, while multiple clients can **borrow** or **return** books.

Requirements:

The server reads from a FIFO named **library_fifo**, where clients send their requests. When a client connects, the server first prompts for the client's name and greets them, asking for their desired request.

The client then chooses an action:

- Borrow a book:
    - The client sends the book name and quantity.
    - The server checks availability:
        - If available, it updates the stock and confirms how many books were borrowed.
        - If not enough copies exist, it partially fulfills the request and informs the client.
- Return a book:
    - The client sends the book name and quantity.
    - If the book exists, the server updates the quantity.
    - If the book isn't in the inventory, give an error because the system only tracks books that were **originally borrowed** from the library.

Note:

- The server will not terminate and will be waiting for other clients to send requests.
- File manipulation is to be done through system calls only (open,read,write,etc). There will be no credit for using any other file manipulation method such as ofstream/ ifstream/ fscanf/ fgets/ fprintf/ fputs etc.

Sample Output:

Books.txt

```
harry potter 10
the hobbit 7
calculus 7th edition 15
computer networks 12
clean code 5
```

(Book names should be in lowercase for simplicity.)

```
./server
Server started. Waiting for client request..
Client connected: Ali
```

```
./client
Enter your name: Ali
Hi Ali! Would you like to borrow or return?
borrow
Enter Book Name: harry potter
Enter Quantity: 5
5 copies of "harry potter" borrowed successfully.
```

Waiting for client request... Client connected: Maham	./client Enter your name: Maham Hi Maham! Would you like to borrow or return? return Enter book name: deep learning Enter quantity: 2 Error: "deep learning" does not exist in the library. Cannot return a non-existing book.

Waiting for client request…

Question 3: Multi-Threaded Prime Number Finder 🔍 [10 Marks]

Objective:

Use **threads** to perform computations based on command-line arguments.

Problem Statement:

Write a **multithreaded** program that finds all **prime numbers** within a given range [L,R] (inclusive), where L and R are **command-line arguments**.

Requirements:

1. Accept two integers L and R from the command line.
2. Main program creates **N threads**, where each thread processes a subrange of max 10 numbers.
    - e.g [0,30] will be handled by 4 threads.
3. Each thread checks for **prime numbers** in its assigned subrange and prints them together along with its own thread ID and subrange of its search.
4. Use pthread_join( ) to ensure all threads complete before the program exits.

Sample Output:

```
./prime_finder 10 45
Thread <thread ID> (30-39): 31 37
Thread <thread ID> (10-19): 11 13 17 19
Thread <thread ID> (40-45): 41 43
Thread <thread ID> (20-29): 23 29
```