CSCI 460 Operating Systems
Assignment 2

This assignment is related to process/thread synchronization. This is a variation of the producer/consumer problem.

Assume you receive the following details about the problem at hand:

- The buffer is a **doubly linked list** (you can assume that the linked list has a maximum size of 50 nodes). Each node can hold an integer value less than or equal to 100, but greater than 0.
- The LinkedList/buffer in this problem is sorted, which means that the head of the buffer contains the node with the smallest integer value and the tail of the buffer contains the node with the largest integer value.
- This problem involves two producers. Producer #1 will generate a node with the random integer value **n**, which is **odd**, and add it to the correct position in the LinkedList. The producer #2 will generate a node with a random integer value **n,** which is **even,** and add it to the correct position of the LinkedList. When the buffer (LinkedList) is full, each producer will generate a random integer value (corresponding to each producer—even or odd) and **wait until the buffer is not full**.
- This problem involves two consumers. Consumer #1 will read and delete from the head of the buffer if the head (node) of the buffer contains an odd number. If the head of the buffer contains an even number, Consumer #1 will wait. Correspondingly, consumer #2 will delete from the head of the buffer if the head of the buffer contains an even value; otherwise, consumer #2 waits until there is a node with an even number at the head of the buffer. When the buffer is empty, both consumers will wait until the buffer is not empty and consume accordingly.
- Each of the 4 processes or threads must print the status of the buffer before and after it modifies the buffer. You can write the output of the threads (consumers and producers) into 4 separate files. The status of the buffer means the values of the nodes in the buffer in the order they appear. **Remember: buffer/LinkedList contains the values in the sorted order, therefore, when a new value is generated, producer must put it in the correct position in the LinkedList/Buffer.**

- For example:

    - consumer_1_output.txt
    - consumer_2_output.txt
    - producer_1_output.txt
    - producer_2_output.txt

- You **must** use the <pthread.h> header file and pthread_create to create a thread for each producer and consumer. You cannot utilize any pre-existing libraries to construct a protected LinkedList.

Read the pthread programming information from the link at the end. In essence, **you must complete this assignment using C or C++ languages**. For more information about Pthreads, refer to the POSIX thread programming tutorial guide.

- You can use basic concurrency control mechanisms like mutex and semaphores to create this program.
- Your goal is to create a buffer, consumer, and producer code such that there will be no deadlocks, livelocks, or starvation.
- You should upload your code and output files for each consumer and producer.
- If your program becomes stuck, analyze the cause from your output file. In this scenario, add your explanation into a PDF file and submit it along with the remaining files.
- My suggestion for this assignment would be to implement concurrency control at the buffer—basically, make use of the idea of a monitor. However, if you prefer, you can delegate control of the buffer to the producer or consumer when it needs to modify the buffer. Make sure that this will not lead to deadlocks, livelocks or starvation.

POSIX Threads Programming: https://hpc-tutorials.llnl.gov/posix/

Rubric: Points 20

This is the rough idea of the rubric; there may be changes to the rubric when the TA grades this assignment, but this rubric should cover the basic areas of the assignment.

Implementation of doubly LinkedList: 5 points

- LinkedList is a doubly LinkedList.
- LinkedList can add and remove elements.

Implementation of Producer (5 points)

- The Producer generates a random odd/even integers.
    - Producer 1 generates odd values.
    - Producer 2 generates even values.
- Producers add the generated value into the buffer. (the correct position)
- Uses pthread to create the producer.

Implementation of the Consumer (5 points)

- The Consumer consumes the correct node from the buffer.
    - Consumer 1 consumes a node with odd number from the head of the buffer.

    - Consumer 2 consumes a node with even number from the head of the buffer.
- Uses pthread to create the consumer.

Implementation of this scenario without deadlocks, livelocks and starvation. (2 points)

Output is submitted in 4 output files (one output file for each consumer and producer). (2 points)

Readable code (1 point).