CS5103: Assignment 13 // Date: 16-11-24

Title: "Coding Bootcamp Challenge Simulation"

A group of programmers are participating in a coding bootcamp, where they need shared access to computers arranged in a circular setup to complete coding challenges. Each programmer requires both the computer on their left and the one on their right to work on a task. Once a programmer completes a task, they release both computers and take a break to brainstorm. Programmers alternate between brainstorming and coding, with only one programmer allowed to use a computer at a time. Your task is to simulate this scenario, ensuring no deadlock (where programmers are stuck waiting indefinitely) or starvation (where any programmer is prevented from accessing computers) occurs.

Task: Implement this simulation in C programming language, representing each programmer as a separate thread. Use mutexes or locks to control access to the computers, ensuring that only one programmer can use a computer at a time. Design the solution to prevent deadlock and starvation, allowing each programmer to complete multiple cycles of brainstorming and coding challenges. The number of programmers (and computers) should be set as user input, with a maximum of 10.

Sample Input:

- Enter the number of programmers (<10): 5

Sample Output:

Programmer 1 picked up Computer 1
Programmer 1 picked up Computer 2
Programmer 1 is solving a challenge.
Programmer 1 released Computer 1 and Computer 2
Programmer 2 picked up Computer 2
Programmer 2 picked up Computer 3
Programmer 2 is solving a challenge.
Programmer 2 released Computer 2 and Computer 3
...

Question 2: Platform Synchronization Problem

Implement a C program to simulate a train station with multiple platforms where trains arrive and depart. Use semaphores and condition variables to ensure that only one train can occupy a platform at any given time. If all platforms are occupied, arriving trains must wait until a platform becomes available. When multiple trains are waiting, they should be served in the order of arrival (FIFO). Each train should signal when it departs, allowing the next waiting train to access

a platform in a smooth and orderly manner. This setup should ensure efficient platform use while avoiding deadlocks and maintaining arrival order.

Take number of platforms and number of arriving trains as user input (both <10).

Sample Input:

- Enter number of platforms: 3
- Enter number of arriving trains: 6

Sample Output:

Train 1 has arrived and is occupying a platform.
Train 1 has departed.
Train 2 has arrived and is occupying a platform.
Train 2 has departed.
Train 3 has arrived and is occupying a platform.
Train 3 has departed.
Train 4 has arrived and is occupying a platform.
Train 4 has departed.
Train 5 has arrived and is occupying a platform.
Train 5 has departed.
Train 6 has arrived and is occupying a platform.
Train 6 has departed.