Assessment Tool

Create a C program that simulates the **Dining Philosophers Problem**. Consider N philosophers at a round table. The round table has N bowls for each philosopher and there are also N chopsticks placed beside each bowl. Use the template provided. Implementing a solution for deadlock and starvation will earn you bonus points.

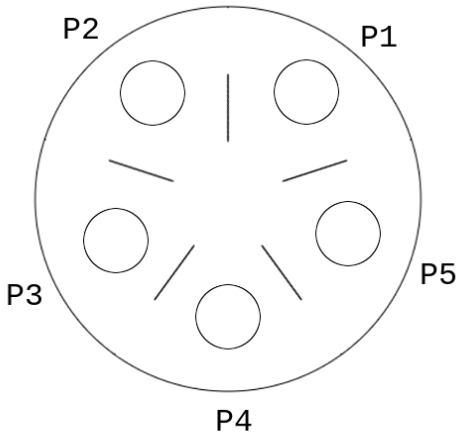

Fig 1. An example of the Dining Philosophers Problem with N=5

The Dining Philosophers Problem is a well-known synchronization problem where philosophers are treated as threads while the chopsticks are the shared resources. The critical region starts when the philosopher tries to acquire the shared resources, the chopsticks, and ends when the shared resources are released.

A philosopher *thinks* for a random amount of time and after this the philosopher starts to get hungry. A hungry philosopher then tries to pick up the chopsticks on their left and tries to pick up their right chopstick without letting go of the left one. After acquiring both chopsticks, the philosopher *eats* for a random amount of time. After eating, the philosopher releases each chopstick, one by one, and goes back to thinking.

Using the solution given above, it ensures that no two philosophers can use the same chopstick at the same time. However, this solution is prone to deadlock. A **deadlock** occurs when all threads are waiting for a shared resource to be released by another. This happens when all philosophers pick up the chopstick to their left at the same time and they are infinitely waiting for their right chopstick to be available. One solution to prevent deadlock is to allow philosophers to drop their left chopstick if the right chopstick is still being used and they go back to thinking. Introducing this deadlock-free solution is now prone to starvation. A **starvation** occurs when a thread cannot proceed to its critical region for a long period of time. This happens when a philosopher is always dropping its left chopstick to give way to other philosophers. One solution to starvation is to allow prioritization of threads.