Project#1

Bound-Buffer problem

Also known as the **Producer-Consumer problem**. In this problem, there is a buffer of n slots, and each buffer can store one unit of data.- There are ***multiple Producers*** and **multiple Consumers**. The producers try to insert data and the consumers try to remove data.

Write Java Multithreading program to solve the Multiple Producer and Multiple consumers Problem.

The solution must be free from Deadlock and starvation

Project#2

Multiple Sleeping Barber Problem

This problem is based on a hypothetical barbershop with ***k barbers***. When there are no customers, all barbers sleep in their chairs. If any customer enters, he will wake up the barber and sit in the customer chair. If there are no chairs empty, they wait in the waiting queue.

Write Java Multithreading program to solve the Sleeping Barber Problem Problem.

The solution must be free from Deadlock and starvation

Project#3

Dining Philosopher's problem

This problem states that there are K number of philosophers sitting around a circular table with one chopstick placed between each pair of philosophers. The philosopher will be able to eat if he can pick up two chopsticks that are adjacent to the philosopher. This problem deals with the allocation of limited resources.

Write Java Multithreading program to solve the Dining Philosopher's problem.

The solution must be free from Deadlock and starvation

Project#4

Readers and Writers Problem

This problem occurs when many threads of execution try to access the same shared resources at a time. There are ***N readers*** to read data and ***K Writers*** to write data to shared resources.

Write Java Multithreading program to solve the Readers and Writers Problem .

The solution must be free from Deadlock and starvation

- Requirements for all projects

- Implementing the problem without applying it to real world application
- Another version with applying the problem to real world application
- All projects are Terminal based (Without GUI)
- Documentation contains
  - pseudocode and solution details with explanation for the solution
  - Examples when the deadlock can occur and how did your solution solve it
  - Examples when the starvation can occur and how did your solution solve it
- Upload source code and documentation and cover sheet to GitHub
- code must be cleaned and organized (based on OOP concepts)

-Bounce Grade

- 5 Grades bounce for Projects based Multithreading Swing GUI
- 10 Grades bounce for Projects based Multithreading JavaFX GUI
- 5 Grades Bounce to illustrate the problem and the solution using any animation tool.

**Note: Please note that the evaluation is also based on the quality of the project and the effort made by each team.

The TA will compare between the submitted projects for each team (for the same project idea).So there may be an adjustment in the evaluation by one to three grades.

For example, Team#1 submitted the code and get 20/20 when Team#5 submitted the project for the same idea . The TA will compare between the projects and notes the project for team#5 better than Team#1, so TA will change Team#1 grade with (-1 to -3)(Max minus is only 3 grades)