Bhavya Patel

AERSP 424 HW2

Question 5 (5 points):

From Question 2, what issues may happen if the robot that finished the task is assigned a new task immediately? And what will be a potential solution for the issues?

Note: The code is optional for this question. You can submit the discussion as a text file.

**Ans:** One potential issue for this code is that if the robot has finished its task assigned and is assigned a new task immediately, the robot may not have enough time to return the tools and prepare for the next task. This can lead to delays in task completion and inefficiencies in the overall process. A potential solution to this issue is to introduce a delay period for the robots after they finish a task. During this delay, the robots can return the tools, reset their state, and prepare for the next task. The delay can be determined based on factors such as the time required to return the tools and the time needed for the robots to get ready for the next task.

Question 6 (5 points):

From Question 3, what issues you might obtain during the development of the program? What issues do you think will happen if some of the actions, e.g., waking up the ATC, checking the traffic pattern, entering the traffic pattern, diverging to another airport, etc., take time to finish? And what will be a potential solution for the issues?

**Ans:** There are several potential issues that may arise during the development and the execution of this program, especially if some actions take time to finish. Race Conditions and Thread Safety is one of the potential issues for the code. The code uses threads to simulate aircraft landings and interacts with the AirTrafficController class, which involves shared data and operations. Without proper synchronization, race conditions can occur, leading to unpredictable behavior and incorrect results. One solution is to use mutex locks and lock guards to ensure thread safety when accessing shared data and when performing critical sections of code. Mutexes help prevent multiple threads from simultaneously accessing shared resources, ensuring data integrity, and avoiding race conditions. ATC State Consistency is another potentials issue for the code. The AirTrafficController class manages the state of the air traffic controller (ATC), such as waking up, falling asleep, and managing the traffic pattern. If actions like waking up or falling asleep take time to complete, there may be inconsistencies in the ATC's state. Using mutex locks and guard blocks when modifying the ATC's state to ensure that operations related to ATC state changes are synchronized and consistent across threads. This ensures that the ATC's state is correctly updated and maintained regardless of the timing of actions.

By addressing these issues and implementing the suggested solutions, the code can be improved for the reliability and accuracy of the program while ensuring proper synchronization and thread safety in concurrent operations involving shared resources.