CS558: Computer Systems Lab
(January–May 2025)
Assignment - 2

Submission Deadline: 11:55 pm, Wednesday, 12th March, 2025

Question 1

Consider a bus station with a single gate for boarding. The station allows N passengers to board a bus at a time. Once N passengers have entered, the boarding gate closes, and the passengers board the bus. The bus departs, and then the gate reopens for the next N passengers. This cycle continues indefinitely. Assume that multiple buses operate simultaneously at the station.

Write a C/C++ program to simulate the bus boarding process as threads in a multi-threaded environment. The threads must be synchronized to ensure:

- No passenger can board the bus unless the gate is open.
- The bus cannot depart until exactly N passengers have boarded.
- The gate cannot reopen for new passengers until the bus has departed.

Question 2

A narrow bridge connects two busy streets. Only one vehicle can cross the bridge at a time to prevent congestion. Northbound and southbound vehicles compete to cross, potentially leading to deadlocks.

Write a C/C++ program to simulate this scenario using semaphores or mutex locks to prevent deadlocks. Implement the following cases:

- Case 1: Vehicles in one direction wait until the bridge is free of vehicles traveling in the other direction.
- Case 2: Vehicles in one direction can interrupt the other direction after some time has passed, but a maximum of K vehicles can cross in one direction before giving priority to the other.

Ensure the program represents vehicles as separate threads and models their entry, traversal, and exit from the bridge.

Question 3

You are a software developer for a major e-commerce platform that handles millions of transactions per day. Your platform runs on a distributed system consisting of multiple servers, and you are responsible for designing a process scheduling algorithm that can handle these transactions efficiently.

System Overview

- The system consists of multiple servers, each running a different service. Each service handles a specific type of transaction, such as payment processing or order processing.
- Each service consists of a pool of worker threads responsible for processing incoming requests.
- Each worker thread has a priority level and allocated resources, determining the number of requests it can handle simultaneously.
- Incoming requests are queued and assigned to the appropriate service based on transaction type. Further assignment occurs based on priority levels.

Scheduling Algorithm Requirements

Your task is to design a process scheduling algorithm that efficiently allocates resources to worker threads while considering:

- Priority level of each worker thread.
- Available resources assigned to each worker thread.
- Type of transaction associated with each request.

The algorithm should:

- Prioritize worker threads with higher priority levels.
- Allocate resources accordingly.
- Assign requests to the next available lower-priority worker thread if necessary.
- Consider that some transaction types require more resources (e.g., payment processing vs. order processing).

Input Format

The input consists of:

- The number of services n in the system.
- The number of worker threads m for each service.
- Each worker thread's priority level and resources in the format: `priority_level resources` (one per line).
- Each request's transaction type and required resources in the format: `transaction_type resources_required` (one per line).

Output Format

The output should include:

- The order in which requests were processed.
- The average waiting time for requests.
- The average turnaround time for requests.
- The number of requests rejected due to lack of resources.

Additionally, during high traffic periods, output:

- The number of requests forced to wait due to resource shortages.
- The number of requests blocked due to unavailable worker threads.