UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS 226 - Concurrent Systems

Assignment 2

- **Date issued:** 07 October 2022
- **Deadline:** 27 October 2022, 8:00 PM

1 Introduction

1.1 Objectives and Outcomes

This assignment aims to review on concurrent programming and synchronization approach via advanced data structures

You must complete this assignment individually. Copying will not be tolerated.

1.2 Submission and Demo Bookings

You are NOT provided with any skeleton code, you will have to implement everything yourself.

Submit your code to **clickup** before the deadline.

You will have to demonstrate each task of this practical during the **physical** practical lab session. So be sure to create copies of your source code for each task separately. Booking slots will be made available for the practical demo.

1.3 Mark Allocation

For each task in this practical, in order to achieve any marks, the following must hold:

- Your code must produce console output. (As this is not marked by fitchfork, formatting is not that strict)
- Your code must not contain any errors. (No exceptions must be thrown)

- Your code may not use any external libraries apart from those highlighted in the textbook.
- You must be able to explain your code to a tutor and answer any questions asked.

The mark allocation is as follows:

Task Number	Marks
Implementation	10
Explanation of Code	10
Total	20

2 Assignment: Practical Requirements

The city population is growing and so is traffic congestion due to the sharp growth in population. The road network in the city cannot keep up with this sudden growth in demand. The city council is seeking means to monitor and easily control the traffic flow.

The road network:

- Represented as a directed graph, https://en.wikipedia.org/wiki/Directed_graph
- Nodes
  - represent an intersection having multiple roads/edges connecting it to other intersections
  - controlled by a traffic light
  - traffic light has multiple states, which it alternate. Example:

```
time: 5, Edges: { e1: "red", e2: "green", e3: "red", e4: "green" }
time: 7, Edges: { e1: "red", e2: "yellow", e3: "red", e4: "yellow"}
time: 5, Edges: { e1: "green", e2: "red", e3: "green", e4: "red"}
time: 4, Edges: { e1: "yellow", e2: "red", e3: "yellow", e4: "red"}
```

- Edges
  - represent a road going from one intersection/node to the other
  - each road/edge has a maximum capacity, meaning there can only be as many cars on the road before the traffic light can turn green, i.e. when the traffic light is red.

2.1 Task

2.1.1 Implementation

You are tasked in implementing a system that show the traffic flow in real time. Simulate a road network with traffic flow.

- You are free to use any lock implementation, you must be able to justify your choice over the other locks.
- A thread will represent an edge and an intersection will be the critical section (controlled by a traffic light)

- Road network
  - create a grid network of 4 intersections, connected by edges. See figure 1
  - note that the outer nodes are not necessary, but you are free to add them if it makes your life easier
  - randomly initialize the traffic light states with a time between 200 and 500 ms
  - randomly initialize the road capacity
- Vehicles:
  - randomly initialize a variable number of vehicles on each road/edge
  - each vehicle has a route it must take, this route must be initialized before for each vehicle.
  - you can randomly initialize the route (if manually selected ensure that all vehicle don't take the same route)
  - You are free to determine the route length, that is the number of edges a vehicle will visit or pass-by.

2.1.2 Output

The following output must be produced:

- when a thread **ENTERS** the critical section, print the queue of vehicles on that particular road/edge the thread is simulating
- when a thread **LEAVES** the critical section, print the queue of vehicles on that particular road/edge the thread is simulating

2.1.3 Notes

- You will have to get creative with handling threads, queues and locks.
- The road network is basically a huge producer consumer network
- A general rule of thumb is that each intersection should have their OWN lock when accessed.
- You may add as much code and as many classes as you deem necessary.
- Any locks used must be FAIR.
- The assignment mainly relay on your creativity, how you translate and simulate the traffic light into synchronization

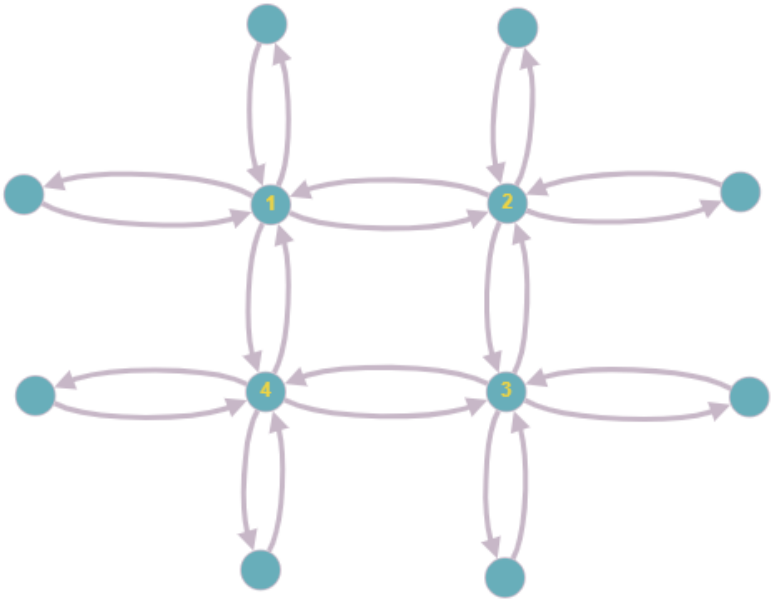

Figure 1: Directed graph