# CSE211 DATA STRUCTURES

## LAB 3 FALL 2024

### QUEUE OPERATIONS

## Prerequisites

Open the terminal and execute the following commands after downloading the `tarball` file:

```
cd /mnt/c/Users/user/Downloads && tar -xvf lab3_2.tar.gz --one-top-level=lab3_2
cd /mnt/c/Users/user/Downloads/lab3_2 && make all
code .
```

## Introduction

In this lab, you will implement advanced operations on a Queue data structure using C++. The Queue is implemented as a template class that can store elements of any type T. Your task is to implement the following challenging operations:

1. `processParallel`: Schedule tasks across multiple processors
2. `routePackets`: Route network packets with bandwidth constraints
3. `processOrders`: Process restaurant orders across cooking stations

## Project Structure

```
.
├── bin/
│   └── queue
├── include/
│   ├── Queue.hpp
│   └── Color.hpp
├── obj/
│   ├── Queue.o
│   ├── Color.o
│   └── main.o
├── src/
│   ├── Queue.cpp
│   ├── Color.cpp
│   └── main.cpp
├── instructions.md
└── Makefile
```

# Implementation Details

## 1. processParallel

- **Purpose**: Schedule tasks across multiple processors considering priorities
- **Parameters**: Vector of {duration, priority} pairs and number of processors
- **Return**: Vector of Results containing {task_id, start_time, processor_id}
- **Example**:

```
Input: tasks = {{4,1}, {2,2}, {3,1}, {1,3}}, processors = 2
Output:
Task 0 starts at 0 on processor 1  // Priority 1
Task 2 starts at 0 on processor 2  // Priority 1
Task 1 starts at 3 on processor 2  // Priority 2
Task 3 starts at 4 on processor 1  // Priority 3
```

## 2. routePackets

- **Purpose**: Route network packets considering bandwidth limitations
- **Parameters**: Vector of PacketInfo (destination, size, priority, arrival_time) and bandwidth limit
- **Return**: Vector of {packet_id, processing_start_time}
- **Example**:

```
Input: packets = {
  {dest:1, size:100, priority:2, arrival:0},
  {dest:2, size:50, priority:1, arrival:1},
  {dest:1, size:75, priority:3, arrival:2}
}, bandwidth = 50

Output:
Packet 0 processed at time 1  // Takes 2 time units (100/50)
Packet 1 processed at time 2  // Takes 1 time unit (50/50)
Packet 2 processed at time 4  // Takes 2 time units (75/50)
```

## 3. processOrders

- **Purpose**: Process restaurant orders across multiple cooking stations
- **Parameters**: Vector of OrderRequest (table_id, items, priority, order_time) and number of stations
- **Return**: Vector of Results containing {order_id, completion_time, station_id}
- **Example**:

```
Input: orders = {
  {1, {"steak", "salad"}, 2, 0},     // 30 units total
  {2, {"pasta"}, 1, 1},              // 15 units
  {3, {"soup", "dessert"}, 3, 2}     // 20 units
}, stations = 2

Output:
Order 0 completed at 30 on station 1
Order 1 completed at 16 on station 2
Order 2 completed at 36 on station 2
```

## Testing

1. Build and run:

```
make clean  # Clean previous builds
make all    # Compile all files
make run    # Execute the program
```

## Restrictions

❌ Do not modify:

- Queue.hpp interface
- main.cpp test cases
- Project structure
- Build system

❌ Do not use:

- External libraries
- Global variables
- Additional data structures (except where specified)

## Academic Integrity

- Individual work only
- No code sharing
- No plagiarism
- Violations result in zero grade

## Submission

1. Test thoroughly
2. Clean build files: `make clean`
3. Send only the `Queue.cpp` file to the course portal

Good luck with your implementation!