University of Victoria

University of Victoria
Midterm 2 PRACTICE QUESTIONS
Fall 2025

Student Name	
V-Number	
Section (CRN)	

Course Name & No.	CSC 360: Operating Systems
Instructor	Wenjun Yang
Duration	50 Minutes

- This exam has **4 questions** across 6 pages, including this cover page. Students must count the number of pages and report any discrepancy immediately.
- This exam is to be answered on the paper provided.
- A basic calculator may be used, although you should not need to use one.
- This is a closed-book exam, but a one-page cheat sheet is allowed.
- Ensure you do not look into cellphones during the exam. You must obtain permission from an invigilator to temporarily leave the examination room.
- We strongly recommend you read the entire exam through from beginning to end before beginning to write your answers.
- The total number of marks in this exam is **100**.
- **Please bring your ONECard for the ID check**.

CPU Scheduling Analysis (25 points)

Consider the following processes arriving at the times shown, with the given burst times and priorities (lower number = higher priority):

Process	**Arrival Time**	**Burst Time**	**Priority**
P1			
P2			
P3			
P4			

- Know how to calculate a variety of time (**waiting time, turnaround time**, etc) for algorithms: **FCFS**, **non-preemptive/preemptive SJF**, and **non-preemptive/ preemptive Priority Scheduling**.
- Understand the pros and cons of each scheduling algorithm.

Sync Concepts (15 points)

Understand:

- mutex
- semaphores (binary vs counting)
- monitor
- condition variables

You may be asked about the differences among them and use cases for each of them.

Sync Problems: Deadlock & Starvation (20 points)

Consider the following pseudocode:

```
Thread 1:
lock(mutex_A)
lock(mutex_B)
// critical section
unlock(mutex_B)
unlock(mutex_A)
```

```
Thread 2:
lock(mutex_B)
lock(mutex_A)
// critical section
unlock(mutex_A)
unlock(mutex_B)
```

- Does this code have a potential **deadlock**? If yes, explain how.

- Provide ONE specific solution to prevent the deadlock.

The Real-World Sync Problem (40 points)

Problem Description

A barbershop located in Dragon Alley consists of a waiting room with **n** chairs plus an additional chair for the barber. If there are no customers to be served, the barber sits in his chair and goes to sleep. If a customer enters the barbershop and all **n** waiting-room chairs are occupied, then the (disappointed) customer leaves the shop. If the barber is busy cutting someone's hair, but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep in his own chair, then the customer wakes up the barber.

Shared Data

The threads have available to them the following shared data with initial values:

```
int customers = 0;                  // Number of customers waiting
int n = 5;                          // Number of waiting chairs
semaphore mutex = Semaphore(1);     // Protects customers counter
semaphore customer_sem = Semaphore(0); // Barber waits on this
semaphore barber_sem = Semaphore(0);   // Customer waits on this
```

Complete the Implementation

Fill in the blanks in the following code. For each blank, write either `wait` or `signal` followed by the appropriate semaphore name (e.g., `wait(mutex)` or `signal(customer_sem)`).

```
// Barber thread function
void barber() {
    while (true) {
        _______(_______);

        _______(_______);

        customers_______;

        _______(_______);

        _______(_______);

        cut_hair();                 // Perform the haircut
    }
}

// Customer thread function
void customer() {
    _______(_______);

    if (customers < n) {            // Check if chairs available

        customers_______;

        _______(_______);
```

```
        _______(_______);

        _______(_______);

        get_haircut();              // Get the haircut

    } else {
        _______(_______);
        // Customer leaves without haircut
    }
}
```

End of Exam