# Lab 4: Queues

COSC-2436

Spring 2025

**Due: April 27**

## Introduction

In this lab we’ll be creating the banking simulation described in the textbook in Chapter

13.4. You’re given starter code with the simulation finished, but key bits are

unimplemented. After finishing the implementation, the starter code will print out the time for various numbers of tellers to finish the given workload. This should tell the manager of the bank how many tellers to hire to improve the wait time for customers and the working time of tellers.

We won’t be implementing our own ADTs, instead we’ll use the ones in the standard

library. The standard library is used heavily within industry and is generally a good choice.

We’ll also be exploring a few new additions to the C++ and the standard library; optional

and variant. Optional and variant are from the world of functional programming (FP) and

are abstract data types that also provide additional control structures for your code.

Optional helps avoid issues common with pointers in a friendly way that doesn’t involve

checking for nulls. Variant is like a union but with better type safety. You’ll also see

transform with lambdas being used in the code, which is like map in FP. A lambda is also called an anonymous function, a function without a name. It’s useful for passing a

bespoke function as a parameter, a common thing in FP. There is an example of this in the starter code.

Functional programming (FP) focuses on using functions to write extensible code whereas object-oriented (OO) programming uses objects instead. Both existed in the early days of computing, but FP wasn’t practical at the time due to slow hardware and a requirement for automatic memory management. Today, most modern languages offer a mix of paradigms to choose from and C++ is no exception.

## Part 1: Event Processing

The code and events largely follow the description in the text. The main difference is the

addition of a vector for tracking multiple tellers and a teller index added to the departing

event; to know which teller is being departed from during event processing. Also, methods were added to the teller class to easily track how busy they are.

Your task is to finish the implementation wherever there are TODOs, run the simulation,

then answer the questions in part 2.

## Part 2: Short Answer Questions

You may include these in the main file as comments if you’d like to just turn in one file.

1. What number of tellers should the branch manager hire? Explain your reasoning.
2. What kind of simulation is this and why?
3. Why use the priority queue for the event queue and a regular queue for the bank line?
4. Can you think of any other problems, aside from banking, that an event simulation could solve? What values would it track?