Project 2 – Simulating a Store Part A: Checkout Lane

Learning Objectives

- Implement Nodes and Pointers
- Implement Queues
- Implement Basic Statistics

Overview

A very common use of computers is to run simulations. A simulation is a program designed to model a real system of interest. By setting parameters and selecting appropriate input values, results are produced that mimic the behavior of the real system. If the simulation is a valid representation of the real system, it can even predict the future performance of the real system.

In this project, you will begin to construct a program to simulate a grocery store. Your goal is to use your simulation to answer a specific question: Should the owner of the store remodel her checkout lanes to include express self-checkout lanes and, if so, how many? **FOR THIS PROJECT WE WILL ONLY MODEL A CHECKOUT LANE!**

Model

Customer Data

To prepare for your analysis the owner has collected the following data for you:

- Arrival time at checkout lane for a customer
- Order size for customers

All times are in minutes, and arrival times are set in reference to store opening time of zero minutes. This data is in the file customer_checkout_data.txt. Each line has 2 values: <Arrival Time At checkout lane> <Order Size>.

You are going to simulate a single checkout lane. The checkout lane should be constructed in such a way that it can either be a normal checkout lane, or an express self-checkout lane. (i.e. use variables for time it takes to scan an item and time it take to handle payment.) For this project you can assume you are at a normal checkout register. The flexibility is to implement Part B of the simulation.

Checkout Information

Cashier takes .01 minutes per item to scan purchases and 1.5 minutes to pay.

Data Collection

Collect the following statistics:

- mean waiting time per customer in your line
- median waiting time per customer in your line
- Percentage of Customers who waited for more than 2 minutes
- Percentage of Customers who waited for more than 3 minutes
- Percentage of Customers who waited for more than 5 minutes
- Percentage of Customers who waited for more than 10 minutes
- total customers passing through each line
- maximum length of each line

Note: a customer begins waiting when they enter the checkout line. They end waiting when they begin checking out.

Clock

We will use a discrete event simulation to simulate the grocery store. This means time is managed by using a simulation clock that is advanced whenever an interesting event occurs. For example, if the simulation time is 0.0 minutes and a customer enters the checkout line at a time 2.0 minutes, we mark the passage of time by advancing the simulation clock to 2.0 minutes (clock = time of enter checkout line = 2.0). For this portion of the simulation, we will only need to update the clock when we transition from someone finishing checking out and the next person begins checking out.

Implementation Details

-You must build a register queue using a nodes and pointers structure. You must implement it in separate .hpp and .cpp files. It should have the following functions:

- default constructor
- deconstructor
- enqueue
- dequeue
- calculateStats – this should be a private method called from print.
- print – This method will print the statistics described above.
- isEmpty – A method that returns whether of not another customer is in the line
- Class Variable: Clock – this holds the “time” in minutes since store opened.

The data type for your register queue is a Customer.

You must build a Customer class. The customer should have the following variables:

- Time reached checkout
- Items to checkout

- Parameterized constructor (timeEnteredCheckoutLane, numitems)

The Customer class can be implemented in a single .hpp file since it is a relatively small Class.

You are welcome to add additional methods and variables to both classes to help you solve the problem.

-You must perform a code analysis (aka provide a Big Θ) for each method in your register queue class.

Grading*

Implemented with Nodes & Pointers	10
RegisterQueue()	5
~RegisterQueue()	5
RegisterQueue::Enqueue	5
RegisterQueue::Dequeue	5
RegisterQueue::calculateStats	20
RegisterQueue::print	5
RegisterQueue::isEmpty	5
Separate .hpp & .cpp files	5
Clock is updated	5
Customer Class	10
Works with main with no changes	5
Code Analysis	15

*Up to one and a half letter grades (15% of total points) could be removed for bad style and/or poor testing. Projects must compile to receive a letter grade.