INTERNATIONAL TECHNOLOGICAL UNIVERSITY
2711 N. 1st St., Santa Jose, California 95134
Phone: (888) 488- 4968 https://itu.edu

Project-3 (10 pts)
Instructor: Ahmed Ezzat

Multi-threaded Ticket Sellers

This project will give you experience writing a multithreaded program using C or C++ and the Pthreads library. Write a program that simulates ticket sellers simultaneously selling concert tickets during one hour.

Suppose there are **100 seats** available to a concert. One ticket seller named **H** sells high-priced tickets, three ticket sellers named **M1, M2**, and **M3** sell medium-priced tickets, and six ticket sellers named **L1, L2, L3, L4, L5**, and **L6** sell low-priced tickets. Each ticket seller has a separate customer queue.

All the ticket sellers work simultaneously during one hour. **Each seller can expect N customers to arrive at random times during the hour**, where **N** is a command-line parameter. Each seller serves customer in the order that they arrive in his ticket queue.

For simplicity, your program will keep track of time in units of a minute. Therefore, customers arrive only at the start of a minute. High-price ticket customers get the fastest service, and each requires randomly exactly **1 or 2 minutes** to complete a ticket sale. Medium-price ticket customers each requires randomly exactly **2, 3, or 4 minutes** to complete a sale. Low-price ticket customers each requires randomly exactly **4, 5, 6, or 7** minutes to complete a sale.

All ten ticket sellers work collaboratively. When a seller serves a customer, the first thing the seller does is check to see if any seats are available, and if so, the seller assigns a seat to the customer and then completes the ticket purchase using the required amount of time. Ensure that two sellers don't assign the same seat to two different customers.

The concert has **ten rows of seats** with **ten seats** in each row. The high-priced seller assigns seats starting with row 1 (the front) and works towards the back. The low-priced sellers assigns seats starting with row 10 (the back) and works towards

the front. The middle-priced sellers start with **row 5, then row 6, then row 4, then row 7**, etc.

Your Program Output

Your program should print a line indicating each event as it occurs. An event is:

- A customer arrives at the tail of a seller's queue.
- A customer is served and is assigned a seat, or is told the concert is sold out, in which case the customer immediately leaves.
- A customer completes a ticket purchase and leaves.

Start each event print line with a time stamp, such as 0:05, 0:12, etc.

After each ticket sale, also print the concert seating chart as a 10-by-10 matrix that shows which customer was assigned to each seat. Identify ticket seller H's customers as H001, H002, H003, ...; the customers of ticket sellers M1, M2, …, as M101, M102, …, M201, M202, …; and the customers of ticket sellers L1, L2, … as L101, L102, …, L201, L202, ... You can indicate still-unsold seats with dashes.

At the end of one hour, each ticket seller should complete whatever purchase may still be in progress and close the ticket window. Any remaining customers in the queues should leave immediately. Of course, if the concert sells out before the hour is up, all the ticket windows should close after the last seat is taken.

Write your program in C or C++ using the Pthreads library.

Run your program for N = 5, 10, and 15 customers per ticket seller, where N is a command-line parameter.

At the end of each run, print how many H, M, and L customers got seats, how many customers were turned away, etc.

Tips

Simulate each ticket seller and each customer with a separate thread. To have customers arrive at random times during the hour, make each customer arrival time to the queue a random integer 0 – 59 minutes. Customers are sorted in the seller ticket queue based on arrival time.

Identify the critical regions. Determine what process synchronization is necessary.

What to turn in

Create a zip file containing:

- Your C or C++ **source files**.
- A text file containing **output** from your simulation runs.
- **A 1- or 2-page report** that describes your software design. What parameters did you adjust to make the simulation run realistically? What data was shared and what were the critical regions? What process synchronization was required?

Email your answers to the grader. Your subject line should be **CSC 502 Assignment #3, Group-n**. Cc all your group members so that the grader can do a "Reply all" when needed. This is a group assignment; all group members will receive the same score.