SWEN 342 Concurrent and Distributed Systems

Bank Simulator

Design and implement a stand-alone program to simulate the workflow in a typical banking environment. Use a single customer queue and three teller threads or processes. When a teller is available the customer at the head of the customer queue is assigned to that teller.

Platform choices are any Linux system, choice of development platform, ST microcontrollers like the Nucleo-L476rg and the STM32L476 Discovery board, macOS, and Cygwin. In all cases make sure that you confirm that your technology base (threads, processes, mutexes, timers, and sleep functions) works as desired.

Problem Statement:

- Skip Phase 1 timing and only do Phase 2!
- Skip this timing! ~~Customers enter the bank to transact business on a regular basis. In Phase 1 each new customer arrives every one to four minutes, based on a uniform random distribution.~~
- Each new customer enters a single queue of all customers.
- In Phase 2 customers arrive every 0.5 to 2.0 ~~seconds~~ minutes in a uniform random distribution. All other parameters are the same for Phase 2.
- Three tellers are available to service customers in the queue. As tellers become available, customers leave the queue, approach the teller and conduct their business. Each customer requires between 30 seconds and 8 minutes for their transaction with the teller. The time required for each transaction is based on a uniform random distribution.
- The bank is open for business between the hours of 9:00am and 4:00pm. Customers begin entering when the bank opens in the morning, and stop entering when the bank closes in the afternoon. Customers in the queue at closing time remain in the queue until tellers are available to complete their transactions.
- The enhancement where the tellers take breaks is not required. It is a bonus activity.

Metrics:

To monitor the performance of the business while running the simulation, display in real time:

1. The simulated time.
2. The number of customers waiting.
3. The status of each teller (busy, idle, etc.) and number of customers they have each served.
4. The updates should be around every 5 to 10 seconds. Use your judgement for the exact frequency.

Metrics are gathered and reported at the end of the day. All times are simulated times. The metrics are:

1. The total number of customers served during the day.
2. The number of customers served by Teller 1, by Teller 2, and by Teller 3.
3. The average time each customer spends waiting in the queue.
4. The average time each customer spends with the teller.
5. The average time tellers wait for customers.
6. The maximum customer wait time in the queue.
7. The maximum wait time for tellers waiting for customers.
8. The maximum transaction time for the tellers.
9. The maximum depth of the customer queue.
10. Only applicable if doing the bonus. Bonus additional metrics (20% for full implementation, pro-rated for partial):
    - Number of breaks for each of the three tellers
    - Average break time for each of the three tellers
    - Longest break time for each of the three tellers
    - Shortest break time for each of the three tellers

If you are on an ST microcontroller you can use UART2 to display the metrics and information in the terminal.

Design Constraints and Guidance:

- You are free to use any supported concurrency mechanism to implement this project. Be sure to describe your selected architecture in your project report. It is recommended that you fully define the architecture before you begin coding and implementation.
- You may want to use a mutex or semaphore to communicate the number of available tellers, but other options include message sending and reply blocking. You need to fully understand your chosen technology base.
- Using a queue is an appropriate approach.
- Each teller is to be modeled as an independent thread or separate process.
- Understand how threads are implemented and how they interoperate.
- Except for the Phase 1 and Phase 2 parameters the simulation parameters as described in the Problem Statement can be "hard-coded" as internal constants. However, only define those values in one location – a header file is strongly recommended.
- You need to be able to easily switch to the Phase 2 customer arrival timing. Use a command line parameter for non-embedded solutions. For embedded solutions use a startup keypress or jumper to switch between Phase 1 and 2.
- The simulation time is scaled such that 100 milliseconds of absolute clock time represents 1 minute of simulation clock. Therefore, this program will run about 0.1 seconds * 7 hours x 60 minutes.
- The output must be presented in simulation clock time (9 AM through closing time around 4 PM).
- In order to get the simulation timing accurate, it is strongly recommended to use timers to keep track of time elapsed, by waiting for the timer to complete. Do not use any CPU-consuming spin loops. Substantially inaccurate simulations will be penalized.
- If on an ST microcontroller I strongly recommended using the STM32CubeIDE software to generate the initial project skeleton; enable the use of the FreeRTOS, RNG, UART2 device to save time.
- Note: Every FreeRTOS thread requires additional HEAP memory. For each thread that you run, you will need to allocate more HEAP properly.

Bonus (20% for full implementation with partial credit available):

- Include random breaks for each of the tellers.
- Each teller will take a break every 30 to 60 minutes for 1 to 4 minutes. If a teller break time occurs while serving a customer they will go on break as soon as they finish the current customer.
- The next break for a teller occurs from 30 to 60 minutes from when they started their previous break.
- A break can only occur after the completion of the current customer transaction.
- The break timing and duration is based on a random uniform distribution.

Submissions and Reports:

Design Phase:

Before coding, you must get approval of your software design and technology selections. This will take form as a Software Design Document and technology demonstration software. This must include appropriate diagrams and explanation to cover the following topics:

- Design of the customer queue including randomized customer arrival time and randomized amount of time required at the teller window.
- Description of how you measure the customer metrics.
- How you assign a customer to a teller -- what is your algorithm?
- How you measure all teller metrics.
- If you want to attempt the bonus describe the design of the break times including how you collect required additional metrics.
- Technology selections including developing software that demonstrates base technologies such as queuing, mutexes, timing, and random number generation on your selected host platform. This is a technical risk spike where you demonstrate that your technology base works as intended.
- Store this in the git folder ***Bank Simulator/Design Phase.***

Phase 1 and 2:

Complete the features as described above. Copy your Software Design from the Design Phase and update it to match your actual implementation.

You will demonstrate your project by following a provided test procedure.

Use the Report Specifications for the required content and format for your project report.

Include two runs of your output in your report. Please submit your report as either a Word compatible document or a PDF document. Use this standard report template for your report: Report

Store this in the git folder ***Bank Simulator/Implementation.***

Due Dates:

Refer to the class schedule for the due dates.

Grading Criteria:

- Program Operation and Demo – 50%
  - Hardware setup is orderly and well organized – 10%
  - Demo sheet functions all completed – 30%
  - Demo operates without faults or restarts – 10%
- Program Design --- 15%
  - Proper initialization
  - Correct use of functions (no copy/paste/edit slightly)
  - Separation of hardware related code from pure software (e.g. the results reporting code)
- Source Code Structure and Readability – 10%
  - Appropriate use of white space – 2%
  - Consistent and good indentation – 2%
  - Appropriate comments at the function and paragraph levels (such as a for loop) – 2%
  - Following C style guide (good names, etc.)
- Report Content – 25%
  - Report is at least 2 pages (not counting pictures, cover page, diagrams) – 5%
  - Demonstrates team understands the problem, solution, and technology (hardware and

software) – 10%
- Report contains all required sections per the report guidelines – 10%