**CIS 657 Lab 2**

Submit two files:

* Create your Lab 2 document (doc or pdf), follow the instruction and then submit it to the **Turnitin link** on the blackboard (under Assignments).
  + You can submit multiple times before the due.
  + After the due, you cannot resubmit newer version.
* Do a “make clean” in code/build.linux, then compress your Nachos folder on the VM, download the compressed file into your local machine and name the compressed file **Lab2\_yourNetId.zip (Note: Zip only!!)**. Submit the zipped file to the **Assignment Code Submission link** of Blackboard.
* You have to make sure your submission is correctly made
  + If you don’t have a confirm email, you should check again.
* **You should raise an appeal within a week after grade is posted.**

Due: Sep. 13 (Friday, end of the day)

Late submission: you will have 2d penalty of your late days.

*Follow the Lab1 instruction and create a new fresh Nachos directory.*

**Overview**

We will practice C++ in Nachos by **writing a program that simulates checkout lines at a supermarket**. Each line is a queue object. The supermarket wants to build a simulation of checkout line to determine how many cashers they will need based on the average number of customers at different time of the day. Here is your situation.

You have been provided with the following known facts:

* A casher takes an average of 5 seconds to scan a single item for a customer.
* The average startup time for a check out (greeting a new customer, moving the conveyer belt to reach items, …) is 10 seconds.
* The average close time for a check out (getting the total, handling coupons, getting payments, …) is 1.5 minute.
* Customers, checking out have a random number of items, average range between 5 to 40 items.
* Peak hours are between 4 to 6 pm and random number of customers arriving for a check out is between 5 and 10 every minute.
* During regular hours customers between 0 and 5 arrive every minute.
* There are maximum 10 checkout lines and minimum 2 lines. Depends on waiting customers checkout lines can open/close.
* Each casher cannot have more than 5 customers in line.
* There is a single waiting queue of arriving customers for a casher when all open lines are full. If there are more than 10 customers in this queue, open a new line.
* If there is no customer in the waiting queue and the last customer of a checkout line is served, the line will be closed.

If the average arrival rate is larger than the average service rate, the queue would grow fast and this may cause long lines. Run the supermarket simulation from 2 pm to 7 pm using the following:

* When a casher starts checking out a customer,
  + Determine the customer’s service time with his/her items, startup time and close out time
  + If total service time is fractional in minute, round up
* For each minute of the day:
  + If a new customer arrives,
    - Check whether there is an available open line (standing customers < 5), enqueue the customer to the open line
    - Otherwise, enqueue the customer into the waiting queue
  + If service was completed for the last customer of a casher,
    - Dequeue the next customer to be served
    - If the waiting queue is not empty, move customer(s) from the waiting queue to the checkout line
    - If the waiting queue and current checkout line are empty, close the line.

The supermarket wants to know the followings:

* for every hour
  + average number of customers arriving for checkout
  + average/shortest/longest waiting time
  + average/shortest/longest service time
  + average number of open lines
  + maximum number of open lines
  + average time each casher will have more than 3 customers standing in line
  + average/smallest/largest number of customers in the waiting queue
* for entire simulation
  + average/shortest/longest waiting time
  + average/shortest/longest service time
  + maximum number of customers in the waiting queue at any time

**Output:**

You should print out appropriate messages including the information that the supermarket wants to know.

* Every hour summary
* Entire simulation summary

**Implementation Requirements:**

* Design/define classes, which contains all the information you need for this project
  + Implement all necessary class in the new files (declaration in .h file and definition in .cc file)
  + The member variables have to be private. Implement public getter/setter functions for accessing these variables
  + Change code/build.linux/Makefile to include the new files to the NachOS compilation process
* Call rand() function to randomly generate a number.
* Implement queues using the functions defined in code/lib/list.h and list.cc
  + Read all the codes and comments in list.h, list.cc, and code/threads/threadtest.cc
  + Hint: ThreadTest() function is called by main()
* The entire process should be done in threadtest.cc.
* Your implementation must be incorporated and compiled with NachOS under thread.

**Lab2 document must include the followings:**

* Cover Page
* Disclosure Form
* Design/your solution for each requirement
  + We are looking for your own answers
* Implementation of your solution (Code snippet with good comments)
  + Do not include Nachos original code
    - You do not need to change list.[cc|h]
  + We want to see your own code/modification
    - You need to modify threadtest.cc
    - Your new classes
  + Should be text (no image/photo of code)
* Testing
  + How to run your tests
  + What each test case is testing (you need to prove your implementation of each requirement)
* Output Snapshots

**Testing:**

**We will build and run your Nachos on the VM.** You must test your program on the VM and also should provide proper test scenario in the Lab 2 document.

**Grading:**

* Syntax/Runtime/Logic Errors with proper Makefile [-50, -15]
* Class declaration/definition respectively [-20, -10]
* Satisfy all implementation requirements [-50, -5]
* Interactive input/output design [-20, -10]
* Output(by Students)/Test(by TAs) [-50]