CS2303 Systems Programming Concepts **A14**

Program 5 {October 3, 2014} 68 Points

Simulation of Robots and Shoppers in RoboMall

Due: Wednesday, October 15, 2014 at 11:59 p.m.

{Note – As this is a large assignment that may not be totally completed by the due date, grading for this program includes some partial credit for programs that do not work completely. See README information required for partial credit at the bottom of this assignment.}

This program provides the students with the opportunity to implement a large and complex C++ program that includes data structures while reusing components from previous programming assignments. This assignment is to be done by the same two-person teams as for Program 4 (unless the instructor has subsequently approved a team change).

Your task is to simulate both robots and shoppers traveling through the two-floor RoboMall. The floor plans for the First and Second Floors are the same as in Program 2.

Stage 1 of this simulation involves robots traveling through the RoboMall at night before the mall opens. This stage involves reusing parts of Program 2 and converting this code to C++ while making a few important changes. To reduce congestion from concurrent robot deliveries, in this program, robots enter the mall at A1 with their start times **staggered** in 25 time unit intervals. For example, R1 enters the mall at time 0 and R2 enters the mall at time 25. To keep the assignment simple, assume multiple robots can occupy the same location in the RoboMall and more than one robot can fit in a store delivering items **at the same time**. One difference from Program 2 is the time a robot spends at a given store. Once a robot arrives at a store, it spends one time unit in the store for each **unique item type** it is delivering to that store. The restrictions on robot movement in the RoboMall are the same as in Program 2. When all the robots have left the RoboMall, stage 1 of the simulation is over. Think of the robots working at night and finishing deliveries just before store clerks arrive to open the mall stores. We will call this moment when the mall opens (namely, when the last robot leaves the mall) as **shop time 0**. {Note – this will be different than the current clock time in the simulation.} All robot deliveries are recorded in the RoboMall database (using the binary tree mechanism from Program 4).

Fortunately, for this simple simulation, the movement of shoppers will be restricted in the identical fashion as robots with shoppers obeying the one-way restrictions given in the floor plans from Program 2. Additionally, multiple shoppers can occupy the same mall location and more than one shopper can be in a given store. Unlike the robots, the shoppers have to interact with the store clerks. All First Floor clerks serve shoppers using **FCFS** scheduling while all Second Floor clerks use a **Round Robin** scheme to wait on shoppers. Clerks sell **one item per time unit** in all stores. That is, if a shopper wants to buy 7 items identified as C6, the shopper will need 7 service units from the clerk to purchase the required number of C6 items. Arrival time of each shopper to the RoboMall (part of the input script for the simulation) will be given relative to shop time 0.

A shopper arrives at the RoboMall with a shopping list containing items and quantities of items to buy. Fortunately, the shopper has a smart phone with an app that enables the shopper to access the RoboMall database. Upon arrival to the mall, a shopper searches the database for each item on its shopping list to find the store **currently** with the largest quantity available of that item in stock (accounting for robot deliveries and prior completed store sales). Once the shopper finds the first store in the ordered database with greater than or equal to the quantity that it has on its shopping list, that store is added to the shopper's traveling itinerary. If no store has adequate quantity to satisfy the shopper's initial demand for an item, the shopper deletes that item from its shopping list. Each shopper uses the app upon arrival to the simulation to run through its shopping list to create its full traveling itinerary in the RoboMall. The app is so fast that it runs in zero simulated time. The shopper then proceeds through the mall going to all the stores on its itinerary purchasing items on its list. **If a shopper is in a First Floor store,** once a shopper receives its complete quantity for a given item, the shopper database is adjusted to show the appropriate reduction in stock for that item at that store. **If a shopper is in a Second Floor store, once a shopper ends a Round Robin time slice, the shopper database is adjusted to show the appropriate reduction in stock for those items sold by the clerk during that time slice.** If by the time, a shopper reaches a store, the store's remaining stock of an item is less than the shopper wants, the shopper buys all the quantity available in that store and then deletes this item from its shopping list. Once a shopper has completed its shopping, it leaves the RoboMall via A1. The simulation terminates when the last shopper has exited the RoboMall.

Command Line Arguments

The rmall_sim simulator takes two command line arguments in the form

```
./rmall_sim   time_slice
```

where

time_slice   is the time slice to be used by round robin clerks on the Second Floor.

Assignment Inputs

Stage 1: Robots Input Data

In both stages of the simulation, the program will read in 'scripted' information for the simulation from the same ASCII file. To keep it simple, do **NOT** use C file I/O commands. The file should be accessed by using a LINUX command line that **redirects** the input from the scripted file. This enables you to test the program either using a file or using terminal input.

Program 5 begins by reading in from the first line of input script, the number of robots, n, in the simulation. Assume the maximum number of robots per simulation is 9 and the maximum number of stores any robot will visit is **12**. The number of robots determines the remaining number of lines of stage 1 input in the script file. The second line of input reads data in the form:

s1 s2 s3 … s_n

with each input data item corresponding to the number of stores R_i must visit (i = 1, 2, …n).
For example, given the first two input lines:

```
4
3  8  4  10
```

The simulator would simulate four robots, R1 to R4 with R1 traveling to 3 stores; R2 going to 8 stores; R3 visiting 4 stores and R4 going to 10 stores. Remember unlike Program 2, Program 5 simulates **concurrent robot movement with staggered start times**.

Next, for each robot, for each store the robot will visit, the program reads one line in the following format:

row col floor i-items $item_1$ $count_1$ $item_2$ $count_2$ … $item_i$ $count_i$

where

row col floor	indicate the location in the mall for the store receiving a robot delivery.
i-items	is the number of unique items listed on this line of input.
$item_i$	is the two digit ASCII code for the ith item on the delivery list.
$count_i$	is the quantity of $item_i$ delivered to the store by the robot.

This is all the input needed for stage 1. As robots make deliveries the shopper database is updated. Once the last robot leaves the RoboMall, the simulator begins stage 2 of the simulation.

Stage 2: Shoppers Input Data

Stage 2 of the simulation begins by reading in the next line of input script from the same file used for stage1, which contains the number of shoppers, s, in the simulation. This is followed by s lines of input (one line per shopper) of the form:

arr-time j-items $item_1$ $count_1$ $item_2$ $count_2$ … $item_j$ $count_j$

where

arr-time	is the relative arrival time of the ith shopper to the simulation**
j-items	is the number of unique items on the ith shopper's shopping list.
$item_j$	is the two digit ASCII code for the jth item on the shopping list.
$count_j$	is the quantity of $item_j$ that the shopper wishes to buy.

Note – the input shopper arrival times are given relative to **shop time 0 defined previously.

The final sentinel in the input data file will be a single line with

99999

If you process all the input data correctly, you should **never** read this final input line. If you reach this input line, there is a mistake in your reading in of all the script data for the RoboMall simulation.

Stage 2 of Program 5 simulates the traversal and shopping of each of the **s** shoppers **concurrently**. Each shopper enters the RoboMall (and thereby the simulation) at its defined arrival time. Upon arrival, each shopper uses the smart phone app in conjunction with its shopping list to create a traveling itinerary (this controls the movement of that shopper through the RoboMall) and to create a data structure to keep track of all the items on the **i**th shopper's shopping list **AFTER** the app has been run.

Shoppers proceed concurrently through the RoboMall going to the stores in order based on their travel itinerary. Once in a store, a shopper is processed by a clerk using the appropriate queuing algorithm. When a shopper finishes at a store, it exits the store and proceeds to the next store on its itinerary.

Assignment Outputs

For each robot in the simulation, the program should print out to a log file **annotating** the robot's significant events that includes at a minimum the following type of information (but not necessarily in this format) :

robot i enters the RoboMall at time t.
robot i arrives at store s (row, col) on the floor+1 Floor at time t.
robot i delivered the following items to store s (row, col):
{list items and quantities delivered here}
robot i leaves store s (row, col) on the floor+1 Floor at time t.
robot i left the RoboMall at time t.

For each shopper in the simulation, the program should print out to the log file the shopper's significant events that include at a minimum the following type of information (but not necessarily in this specific format):

shopper i enters the RoboMall at time t.
shopper i arrives at store s (row, col) on the floor+1 Floor at time t.
shopper i bought the following items at store s (row, col):
{list items and quantities delivered here}
shopper i leaves store s (row,col) on the floor+1 Floor at time t.
shopper i left the RoboMall at time t.

Note – Robot and shopper events will be interleave due to the concurrency of robot movement and shopper movement. Shopper event times should be listed as simulated times and NOT times relative to shop time 0. Additionally, be sure to annotate in your shopper log when a shopper runs into a problem with inadequate quantity of items either when the app is run or when the shopper reaches a store.

When the simulation ends, indicate the simulation ending time appropriately in the log file. Additionally, compute and print out the time each shopper spent in the RoboMall and the average shopper time in the mall.

Possible Extra Credit Points

If your team's Program 5 is running completely, your team can earn 5 extra credit points by running the shopper app EACH time a shopper is about to leave a store. Running the app at this time might change the shopper's itinerary due to a new shortage of an item on the shopper's shopping list due to an earlier shopper exhausting the supply at that store.

Your team can earn up to additional 5 extra credit points by updating the store map printout from Program 2 such that your program periodically (e..g, every 50 simulated time units) prints out a map of the RoboMall that indicates the current location of all robots and shoppers. Note, this is a tricky task because multiple robots and shoppers can be at the same location or in the same store.

What to turn in for Program 5 (prog5)

An official test script file will be made available a few days before the due date. Turn in your assignment as a tarred file to the Linux ***turnin*** program on the CCC machines. that includes all your source code, a make file, a README file and two distinct log files. While teammates may work together on one specific subroutine or function, the comments in the source code must list ONLY A SINGLE AUTHOR per piece of code. This provides a rough estimate of the amount of effort each team member contributed to Program 5. The README must provide any information to help the TA or SA test and grade your program. You need to run your program **twice** with different RR slices such that each submitted log file corresponds to one time slice.

If you turn in a program that does not compile or a program that only works partially, but you believe that your program does parts of the assignment correctly, then it is critical that your README file state clearly what parts of the program you believe are working and those parts that are not working. Basically, the grader will only give partial credit if you provide assistance in determining how much partial credit you should receive. The README file needs to state honestly the state of the program when turned in. Failure to provide a clear README file can seriously lower any partial credit you might receive for this programming assignment.