CS2303 Systems Programming Concepts A08

Program 5 64 Points
Due: October 15, 2008 at 11:59 p.m.
A Simulation of Shoppers in a Mall

{Note – Since this a large assignment that may not be completed by the due date, this program will be graded with applying some partial credit for programs that do not work completely. Receiving partial credit is strongly dependent on explaining the state of your program in the README file.}

This assignment provides the students with the opportunity to implement a fairly large and complex C++ program that includes data structures while reusing components from previous programming assignments.

Your task is to simulate concurrent shoppers moving and shopping in a four-floor shopping mall. The floor plans for the First, Second and Fourth Floors are the same as in Program 2. The floor plan for the Third Floor is given at the end of this assignment. The rules for shoppers entering stores are the same as the rules for robots accessing bins in Program 2. Stores on the Third Floor can be accessed from **any** square adjacent to the store that a shopper can walk.

Shopper motion on all four floors is quite restrictive. To reduce **(but not eliminate)** multiple shoppers 'crowding' onto the same mall square, the floor plans indicate one-way and two-way paths that a shopper can travel on each floor. Note, for this simulation it is possible to have multiple shoppers occupy the same square or be shopping in the same store at the same time. To explain the rules of shopper motion, we define crowd (s1) to indicate the number of shoppers in a square s1 when it is shopper i's turn to move.

This program provides a simulation of the time and money used by all the shoppers to shop at their list of stores. The time it takes shopper i to move from square s1 to an adjacent square s2 is crowd (s1) **time units**. It also takes crowd (s1) **time units** to move from a square to a stairs square, from a stairs square to a square, or to move from a square into a store. It takes **2 x** crowd (s1) **time units** to use the stairs between any two floors, i.e, to move from a stairs square to another stairs square.

To keep all the simulations similar, your program should move the shoppers in increasing shopper-id order for the same simulated time. Thus, to simulate shopper 6 and shopper 18 moving at the same time, your simulation should move shopper 6 first.

Command Line Arguments

The mallsim simulator takes two command line arguments in the form

```
./mallsim  shoppers  number-sales
```

where

shoppers	indicates the number of concurrent shoppers in the simulation.
number-sales	indicates the number of stores having sales today.

Assignment Input

To initialize the simulation requires three distinct pieces of input:

1. (sale information)

The program reads number-sales store-ids and records the fact that these are the stores having sales today. For example, if number-sales is 5 a sample input line is:

```
1207 2301 2402 3311 4305
```

You can assume the store-id's are input here in increasing store-id order.

2. (initial shopper information)

The program reads in shoppers lines of input of the form:

```
shopper-id  arrival-time  xpos  ypos  zpos  sales  non-sales
```

where for each shopper

shopper-id	is an integer that identifies this shopper in the simulation.
arrival-time	indicates the simulated time when this shopper enters the simulation.
xpos ypos zpos	indicate the start location for this shopper. zpos indicates the floor and xpos ypos indicate the location on that floor. {Assume the upper left corner position on every floor is (0, 0).}
sales	indicates the number of potential stores with sales that this shopper hopes to visit.
non-sales	indicates the number of other stores this shopper will visit regardless of whether or not that store is having a sale.

For example, an input of:

```
6 20 1 7 3 2 4
```

indicates that shopper 6 will enter the simulation at simulated time 20 at an initial location on the Third Floor in location (1, 7) which is just to the left of store 3201 (see the Third Floor plan). This shopper will possibly visit 2 stores that are having sales and will definitely stop at 4 other stores.

All the initial shopper information will be read in before the detailed shopper information.

3. (detailed shopper information)

For each shopper, the program will read in sales + non-sales lines of input of the form:

```
shopper-id  store-id  service-time  money-to-spend
```

where the first sales input lines refer to possible sales stores and the next non-sales input lines are information about the other stores the shopper will visit. In both cases:

service-time	indicates the simulated time that shopper-id will spend in store-id.
money-to-spend	indicates the amount of money (in dollars and cents) that shopper shopper-id will spend in store-id.

To continue the example using the input for shopper 6 above, there would be 6 input lines (2 sales lines and 4 non-sales input lines) such as:

```
6  4102   10     45.00
6  1406   12     72.50
6  3213    5    20.00
6  2202   15   120.00
6  1107     8    35.35
6  4307   20  208.00
```

Shopper 6 will shop in stores 4102 and 1406 if they are having sales. Shopper 6 will also shop in stores 3213, 2202, 1107 and 4307. Note, the store-ids are **NOT** sorted at this point.

Shopper Cell Phone Use, Motion Planning, Store Visits and Spending Accounting

Each shopper enters the simulation at the specified arrival-time and begins shopping from the specified start location. The first action of each shopper is to make cell phone inquiries (in zero time) from the start location to determine whether each store on that shopper's sales list is having a sale today. Namely, the shopper checks to see which of the stores on the shopper's sales list is on today's sales-list. The shopper's expense per cell phone inquiry is given by the formula:

$$\text{phonecallcost} = \$0.10 + \text{ceil}\left(\log_3(\text{Euclidean distance(shopper, store)})\right) \times \$0.10$$

Stores (and associated information) on a shopper's sales list but **not** on today's sales-list are eliminated. The stores having sales and the non-sales stores for a given shopper must then be sorted in increasing numerical order before the shopper begins walking and shopping in the mall. Identical to the robot motion in Program 2, shoppers will access stores in increasing numerical order by obeying the travel rules specified on the floor plans. Your program should use the shortest-path to a store regardless of the shopper's initial position. Remember several shoppers can occupy the same square in the mall, but this affects the crowd factor discussed previously.

Note- shoppers cannot travel 'through' stores. To go to a different floor, the shopper must go to and use the stairs.

Shoppers receive service-time (store-id) units of service in the stores they visit. Since multiple shoppers can be in any store at the same time, the simulation will model store service time using FCFS and Round Robin (RR) queuing where the store is only able to service one customer at a time. Store service on Floors 1 and 2 will be modeled using FCFS queues and store service on Floors 3 and 4 will be modeled using RR queues where the shopper time slice equals the floor number. For example, for store 3213 the RR time slice is 3 time units. Once a shopper has completed its service time in a store, it is located at the appropriate adjacent square in the mall with no additional motion time.

Your program keeps track of the total money spent by each shopper including cell phone costs. Thus, a shopper's life begins at his/her starting location by making cell phone calls to determine the final sorted list of stores to visit. The shopper then moves and enters all the stores on its final store list using the stairs when switching floors. When a shopper has finished shopping at the last store on the list, it leaves the mall (and is eliminated from the simulation) via the stairs on the First Floor. Namely, a shopper's recorded completion time is when the shopper **arrives** at the First Floor stairs after shopping at all the stores on the shopper's final store list..

Assignment Output

Your output routines need to be designed to produce line oriented output to be sent to the screen or to an output file for grading analysis.

Every 50 time units, you need to provide the current position of every shopper in the Mall. This can be done by a simple list or by indicating all the shopper locations via a 'creative' floor plan map for all four floors of the mall. For the simple list option, the output should look something like:

```
At time = 50
Shopper  Floor   Square/Store

   3       4       (3,5)
   6       1       1302
   8       2       stairs
   9        3          ( 7,7)
```

Students successfully providing a creative floor plan map can earn up to **4 extra credit points for this programming assignment.** Note this floor plan is labeled creative because you need to design a scheme that somehow shows multiple shoppers on a square and multiple shoppers in a store.

After the simulation completes print out a summary of each shopper's simulated activity that includes the following sample outputs:

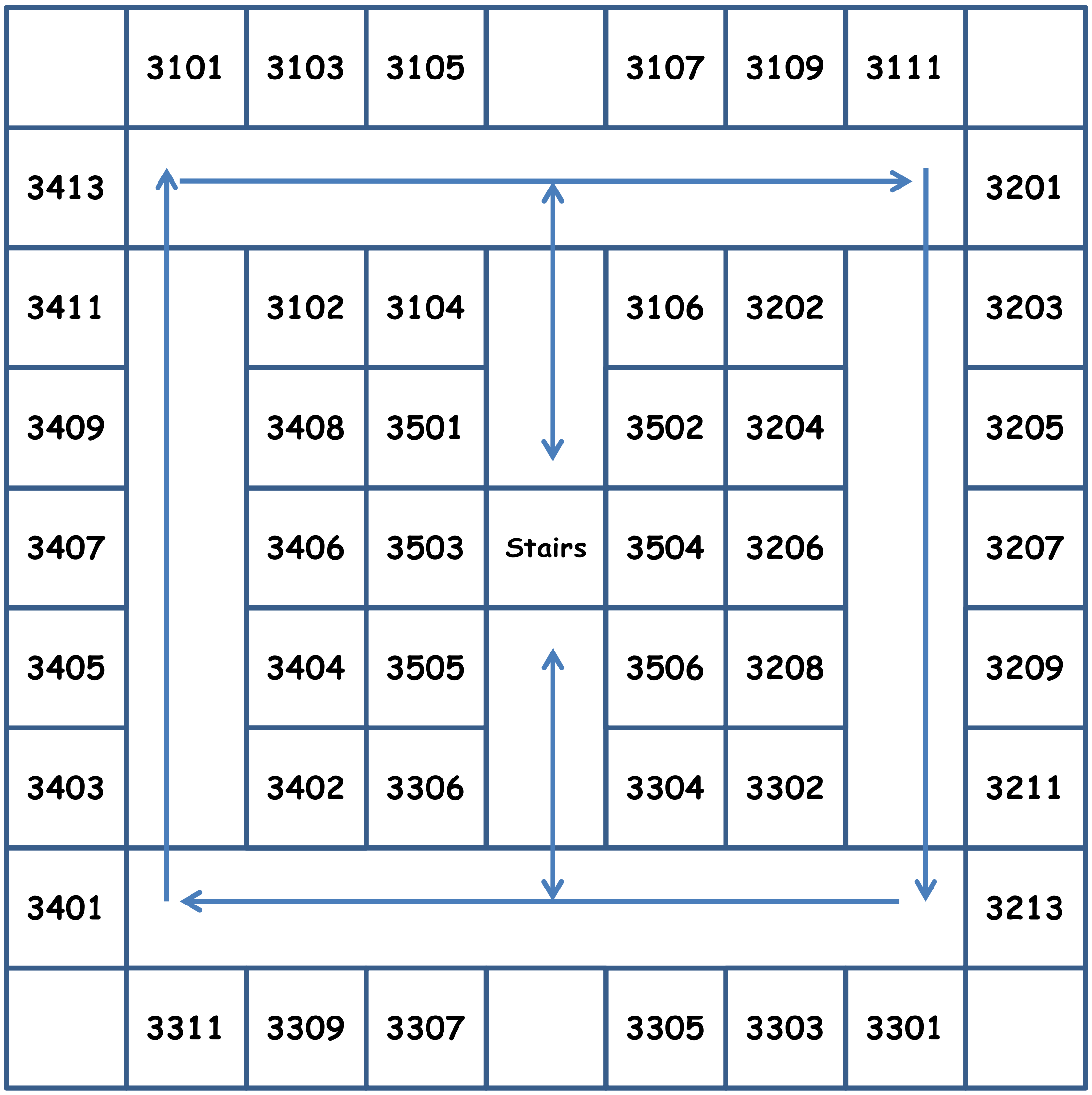
```
Shopper    Start Time   Finish Time   In-Store Time    Total Spent

    1           0            89            36           $234.25
    2           3           212           124           $310.50
...
```

What to turn in for Program 5

An official test file will be made available a few days before the due date. Turn in your assignment using the ***turnin*** program on the CCC machines. You should turn in a tarred file that includes your source code, a make file and a README file. The README file should provide any information to help the TA or SA test your program for grading purpose.

If you turn in a program that does not compile or a program that only works partially, but you believe that your program does some 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 will receive. The README file needs to state honestly the current state of the program that you turn in.

Third Floor


3101
3103
3105
3107
3109
3111
3413
3201
3411
3102
3104
3106
3202
3203
3409
3408
3501
3502
3204
3205
3407
3406
3503
Stairs
3504
3206
3207
3405
3404
3505
3506
3208
3209
3403
3402
3306
3304
3302
3211
3401
3213
3311
3309
3307
3305
3303
3301