CS 5/7343 Spring 2021

Programming Homework 2

Due Date: 4/3 (Sat) 11:59 pm (No extensions). 5% bonus if you hand in by 3/31 (Wed) 11:59pm

The goal of the program is to provide experience on using synchronization tools (mutex) and gain some experience on server/threadpool programming.

Application – Auctions

This program will simulate auctions. The main process will be an auctioneer which will have different type of items (denoted by integers 1 to k) up for auction. There may be more than one item available for auction for each type.

Customers and tables

There are a set of tables in the auction house. (Each table should have an id, similar for what you do with the threads in program 1). Each table will sit one customer. Customers arriving will be put into a wait queue (there is no limit on the length of the queue). For any table without a customer, it will scan the wait queue to see if there is a customer. If there is, the table will get the customer to sit there. Customer are assigned to tables on a first-come-first-serve basis.

Each customer has three attributes:

- An ID (as an integer)
- The total amount of money the customer has (as a positive integer)
- A list of items that he/she is interested in, represented by a list of integers. (notice that the customer may be interested in more than one item for a certain type, so the list of integers can have duplicates). You can assume the list will only be of integers between 1 and k.

A customer can leave any time a round of auction has finished. However, when he/she has 0 dollars left, he/she must leave. Also, if he/she has obtained all the items he/she is interested in, he/she must leave immediately.

Each table is represented by a thread that is created in the beginning of the program. Once each thread is created, it is NOT joined when a customer leaves. Rather, the thread will see if there is any customer available on the wait queue and sit one of them if available. The thread only quits when the main process tells it to.

Each round of auction

A round of auction starts when the auctioneer (main process) announce the next item to be auctioned. Before the auctioneer do that, he/she check which table is occupied, and will only allow those tables that are occupied at that time to take part in that round of auction.

Each customer will then check whether he/she is interested in the item. Then he/she will bid a random amount (not more than the amount of money he/she has). A customer will bid 0 if he/she has no interest in the item. Any bidding amount must be an integer.

If no one bid, the auction ends and the item is discarded.

If only one customer bids, the auction ends immediately and the customer that bid will win the item.

However, if more than one customer bids, then any customer that bids will be allowed to bid one more time. He/she must either maintain the same amount or bid a higher amount. Notice that the customer does NOT know the amount being bid by any customers. For this program, you can assume in this case the customer will have a 50% chance of maintaining the same bid amount, and 50% chance of increasing the bidding amount randomly. After that, the auctioneer will determine who get the highest bid, and declare the winner. Notice that if there is a tie, no one win the auction and the item is discarded.

Once the round finishes, the auctioneer need to inform every table that the round is done, and inform everyone who has won the auction (or nobody wins).

Input

Your program should read an input file. The input file have the following content:

- The first line consists of six integers
    - The first number is the number of tables available. It should be at least 2.
    - The second number is the number of types of items (k in the description)
    - The third number is the number of customers in the input file. It should be at least 2.
    - The fourth number is the maximum numbers of items that are auctioned. If all the items are auctioned, the main process should tell all customers that it is quitting (and ask everyone to leave). On the other hand, if there is no customer left, the customer should quit).
    - The fifth number denote whether the items to be auctioned is to be generated randomly. If it is 1, then the items are generated in a cyclic sequence of 1, 2, 3, …, k, 1, 2, 3 ….. Any other number denotes that the items is to be generated randomly.
    - The sixth number is 1 if you want to run the extra credit, otherwise it is ignored.
- Each subsequent line represents a customer. Each line has the following format:
    - The first number is the ID of the customer (integer)
    - The second number is the amount of money the customer has in the beginning (positive integer)
    - The third number denote the number of items that the customer is interested
    - The rest of the line denote the type of each item that the customer is interested in (can have duplicates)

You can assume the input file to follow the format. If the file input format is incorrect and your program crash, you are not responsible.

Program Structure

Your program should have the following structure:

Main Process

Read the input file
Pre-process and initialization
Create the tables (threads)
Sit the initial customers

Repeat
    Determine who can participate in the next round of auction
    Select an item to auction
    Auction the item
    Send message to threads to denote who win/lose

Until all auction ends or all customers left
Print final information
Tell all threads to quit
Join all threads
Exit the program

Foe each thread

Initialization
Repeat
    If there is no customer
      Find a customer to sit join this table
    Repeat
      Going through each round of auction
    Until customer leaves
Until told by the main program to quit
Print final information
Quit

Output Requirements

Your program should print the following statements. **The output is considered a major part of this project, so please follow direction closely**.

- When a customer is added to the customer queue, you should print the following statement "Customer <ID of the customer> added to the queue"
- When a customer is assigned to a table, you should print "Customer <ID of the customer> assigned to table <table ID>".
- When a customer leaves, you should print "Customer <ID of the customer> leaves, winning <the number of auction won> auctions, amount of money spent = <amount of money spent>, amount of money left = <amount of money left>
- For each round of auction:
    - After determining who is eligible for this round, the program should print "the following customers are eligible for this round of auction : <list of ID of customers eligible>"
    - After the item is selected, the program should print "Item of type <item type> up for auction"
    - When anyone makes a bid (at any round), the program should print "Customer <ID of the customer> at table <table ID> bids <amount being bid>".

- When a second round of bidding is required, the program should output “Second round of bidding needed”
- At the end of each auction, the program should print “Customer <ID> won the auction, with amount <bid amount>”. IF no one win the auction, it should print “No one won the auction”
- At the end of the program, you should print: “total number of item auctioned : <# of auctioned> item, total number of item successfully auctioned: <# of item that have actually been won by someone>, total amount: <sum of the total amount paid for the items>”

Bonus (20 points, Required for 7343 students, optional for 5343 students)

Your program should create an extra thread that add new (randomly generated) customer. Your extra thread structure should be like that

```
Repeat
    Int Delay = 100000;
    Int Dtry = 1;
    Int x;
    For (int I = 0; I < Dtry; i++)
        For (int j = 0; j < Delay; j++)
            X = 1;
     Generate a random number between 0 and 4
     If (number generated == 4)
        Generate a random customer and add it to the queue
        Dtry = Dtry + 1
      Else
          If (Dtry >: 2) then Dtry = Dtry / 2; else Dtry = 1
until received a message from the main process to quit
quit
```

Comment bonus

You will get a maximum of 5% bonus for comments in your program. To get that 5% you need to:

- For each function, put comment to describe each parameter, and what is the expected output
- For each mutex/semaphore used, denote what is it used for
- For each critical section, denote when is the start and end of that critical section (you can name your critical section in various ways to distinguish among them (if necessary))
- For loops and conditional statements that do non-trivial work (you decide what is non-trivial) put comment before it describing what it does.

What to hand in

You should put all you source code into a zip file and upload the zip file to Canvas