COP290
ASSIGNMENT 6: Event Driven Simulation Solution

DIPEN KUMAR (2018CS50098)

SUMMARY OF MY PROGRAM:

I have a struct Event which has five members (fields) –

1) Float time- This stores the time of the event when its action is to be performed.
2) Int id- This helps me to distinguish between event associated with customers and events associated with tellers. It also help me to distinguish between "newly arrived customer" and "customer whose service at the bank has been completed". Specifically speaking-
    - "-2" represent "newly arrived customer"
    - "-1" represent "customer whose service at the bank has been completed"
    - i where i={0,1,2,3,…} represent "teller with index i"
3) Float arrTime- This stores the arrival time of the customer for events associated with customers where as "-1.0" for events associated with tellers.
4) Float startExe- This stores start time of service for events associated with customers else "-1.0" for events associated with tellers.
5) Float stopExe- This stores the time when the customer is served for events associated with customers where as "-1.0" for events associated with tellers.

I have another struct Node which I created in order to further create linked list. It has two members (fields)-

1) Event *event- A pointer to refer an object of an Event. This is the value of the node.
2) Node *node- A pointer to refer an object of the next node.

Next I have struct Queue which is a linkedlist of nodes which further have their values as event. This has two fields-

1) Node *head- This points to the head node of the linkedlist.
2) Node *tail- This points to the tail node of the linkedlist

Below are the methods I have implemented as asked in problem statement-

1) To add an Event to the event queue-
    - pushBack- This given a pointer of queue and a pointer of event pushes the node of value given event at the back of the queue (linkedlist of node of value event). It first creates a node of value given event by allotting memory using malloc.
    - pushTime- This given a pointer of queue and a pointer of event pushes the node of value given event just before the first node it encounter with value event whose time is greater than the given event time. It first creates a node of value given event by allotting memory using malloc.
2) pop- To remove an Event from the event queue. This given a pointer of queue removes the head node from the queue (linkedlist of node of value event) and returns the pointer of its value event.
3) To invoke some action when an Event is removed from the event queue-
    - action- with type of queuing: one per teller.

when event is “newly arrived customer” pushes it back of the shortest queue among the array of the queue for tellers, one queue per teller. I have a static variable array of int whose size is same as that of array of queue for tellers i.e. number of tellers. Array of int at index i has size of queue at index i of array of queue for tellers. With this array of int I iterate and find out what are those queues with shortest size. If they are multiple I chose a random from them.
When event is “customer whose service at the bank has been completed” then after collecting the statistics, I delete the event object.
When event is that of teller then- If there is no customer waiting in any line, put the teller event back into the event queue with a random idle time of 1-150 seconds else If there is a customer waiting in its line, remove the first customer from its line, generate a random service time according to the input parameters of the program, and add two events to the event queue, sorted by time. One is a customer event and represents the completion of that service. The other event is a teller event representing completion of a service and to look for the next customer (or to idle). If its line is empty then check randomly for other lines.

- action2- with type of queuing: common.
  When event is “newly arrived customer” pushes it back of the common queue.
  When event is “customer whose service at the bank has been completed” then after collecting the statistics, I delete the event object.
  When event is that of teller then- If there is no customer waiting in common queue, put the teller event back into the event queue with a random idle time of 1-150 seconds else If there is a customer waiting in common queue, remove the first customer from the line, generate a random service time according to the input parameters of the program, and add two events to the event queue, sorted by time. One is a customer event and represents the completion of that service. The other event is a teller event representing completion of a service and to look for the next customer (or to idle).

In my main method I have created a single eventQueue and a array of queue tellerQueue with one queue per teller in case of one per teller queuing model whereas I created a single common queue in case of common queuing model. I have created event for every customer and teller and put into eventQueue. For customer I have chosen random service time and id=”-2” for “newly arrived customer”. For teller with index i id=i and time is random idle time of 1-600 seconds. I started removing events one by one from eventQueue and invoked its action method i.e. action in case of one per teller and action2 in case of common until simulation time is over. At last I printed out the summary with the information as asked in problem statement. I have shown command line.
GNUplot is used to plot a graph between average amounts of time a customer spent in bank versus number of tellers (assuming number of queue = 1)

HOW TO RUN:

There is a make file in Top directory-

1) make qSim- This will make an object file qSim.o in Top/obj and also make an executable qSim in Top/bin by using source code qSim.c in Top/src.

2) make run- This will run the qSim with an input testcase from Top/TestCases. It show command line and the output in the console. It also make a data.txt with “average amounts of time a customer spent in bank versus number of tellers“ and its gnuplot-plot.png in Top/output
3) make clean- This will delete Top/output/plot.png Top/output/data.txt Top/bin/qSim Top/obj/qSim.o. It will get rid of extraneous files.

DETAILS OF ANY PROBLEM:

There is no problem with my implementation to my knowledge. Simulator is working fine and this assignment is completed from my side with 3 testcases.

ANALYSIS OF MY RESULTS:

A single queue is **better** than a queue per teller because-

1) single queue has lower maximum time a customer spent in the bank.
2) single queue has lower maximum wait time.
3) single queue has lower standard deviation of the amount of time a customer spent in the bank.
4) single queue has lower standard deviation of the amount of time a customer wait in the line.