```
//Andrew Ingle 04/010/2021 - Team I - Final Project
//An overall algorithmic loop, a master function manages sequencing of menu
 functions
//and the functions needed to handle the server processing of the next
 client/customer's selections.
//When new client connects to chosen server's port,
//server commits a thread to run this master function
//loop continues until until client ends session by choosing "exit the
 program"from main menu


//will need to have synchronized access to available seats struct in shared
 memory and "summary" files

#include "andrew_trainTicketMaster.h"
#include "max_trainSeating.h"
#include "caleb_server.h"
#include "andrew_serverFuncs.h"
#include "aarushi_funcs.h"


int trainTicketMaster(int socket, int server_name, availableSeats* shm_ptr,
 int shm_fd, sem_t *reader, sem_t *writer){


        while(1) {//infinite loop until customer exits program

                int customerResponse = 0;
                dates date; //struct type dates will hold today and tomorrows
                 date


                customerInfo nextCustomer; //temp struct to hold next
                 customers info
                int ticketNumber = 0;
                customerInfo customersMods; //struct that holds customers info
                 for modification or cancellation
                bool cancelConfirmation = false;
                int previousDayOfTravel = 0; //used to hold previous day
                 during modify day of travel
                int newDayOfTravel = 0; //for modify day of travel
                int numberOfTravelersRequested = 0; //for changing day of
                 travel
                int addedTravelers = 0; //if more travelers added during
                 modify reservations
                int travelersToRemove = 0; //if less travelers than before
                 during modify reseravation

                int exitReturnType = 0; //unused for now
```

```
customerResponse = mainMenu(socket); //returns the int
 response (see below)- presents main menu to customer via tcp,
 receives response and returns int response adapted from
 Caleb's readFromUser()
switch(customerResponse){
case 1: //makeReservation
        nextCustomer = reservationMenu(socket); //will ask for
         and receive via TCP customerInfo, and save to
         customerInfo struct and return struct
        if (checkIfAvailableSeats(nextCustomer.dayOfTravel,
         nextCustomer.numberOfTravelers,socket,shm_ptr) ==
         true){ //dayOfTravel 1 for today and 2 for tomorrow
                if (confirmReservationMenu(socket) == true)
                 {//menu asking to confirm reservation//if
                 returns true then proceed
                        //needs to be synchronized: //priority
                         is given to customers with most
                         travelers
                        displayAvailableSeats(nextCustomer
                         .dayOfTravel,nextCustomer
                         .numberOfTravelers,socket,shm_ptr);
                         //shows available seats customer
                         selects starting index (seat) and #of
                         travelers fills in seats
                        sem_wait(writer);
                        nextCustomer =
                         selectAvailableSeats
                         (nextCustomer,socket,nextCustomer
                         .numberOfTravelers,shm_ptr);
                         //accesses shared memory and alows
                         customer to select from available
                         seats and writes to shared memory and
                         saves bookedSeats to customer struct
                         copy
                        nextCustomer.ticketNumber =
                         assignTicketNumber
                         (nextCustomer,socket,shm_ptr);
                         //assign ticket number //can be a
                         random num or incremented value in
                         shared memory
                        writeToSummaryFile
                         (nextCustomer,server_name,socket);
                         //writes to appropriate day's summary
                         file, ticket number will be used to
                         search summary later on
                        sem_post(writer);
                        sendReceipt
                         (nextCustomer,socket,server_name);
                         //sends receipt code via tcp (which
                         tell client to get call
                         makeReceipt(), which opens a file
                         fprints received data(receipt) and
                         closes file)
                        // then sends receipt strings to
                         client//
                }
                else {//customer didn't confirm reservation //
                        trainTicketMaster
                         (socket,server_name,shm_ptr,shm_fd
                         ,reader,writer); //recursively
                }
        }
        else {//sorry not enough seats available!
                trainTicketMaster
                 (socket,server_name,shm_ptr,shm_fd,reader
                 ,writer); //recursivley
        }
        break;

case 2: //ticketInquiry //syncrhonization, just reading so
 just have to make sure no other writers at time of reading
        ticketNumber = ticketInquiryMenu(socket); //will ask
         for ticket
        displayTicketInfo(ticketNumber,socket); //will search
         summary files for ticketNumber
        break;

case 3: //modifyReservation //needs to be synchronized so no
 other concurrent writers or readers
        ticketNumber = ticketInquiryMenu(socket); //will ask
         for ticket
        customersMods = retrieveCustomersInfo(ticketNumber);
         //will retrieve customer info from summary file
        displayTicketInfo(ticketNumber,socket); //display
         ticket info to customer
        customerResponse = modifyReservationMenu(socket);
         //returns int for response
        switch (customerResponse){
                case 1: //change customers seats
                        customersMods =
                         freeCustomersSeatsInSharedMem
                         (customersMods,socket,0,shm_ptr);
                         //uses customer struct properties
                         dayOfTravel and bookedSeats[] to find
                         and free seats in shared memory,
                         updates customers .bookedSeats[] to
                         be empty
```

```
                                   displayAvailableSeats(customersMods
                                    .dayOfTravel,customersMods
                                    .numberOfTravelers,socket,shm_ptr);
                                   sem_wait(writer);
                                   customersMods =
                                    selectAvailableSeats
                                    (customersMods,socket,nextCustomer
                                    .numberOfTravelers,shm_ptr);
                                    //customer selects new seats, updates
                                    shared mem, can use
                                    .numberOfTravelers to cap how many
                                    they can select
                                   sem_post(writer);
                                   //send seats changed message
                                   break;
                           case 2: //change day of travel
                                   previousDayOfTravel =
                                    customersMods.dayOfTravel;
                                   newDayOfTravel = requestInt("\nWhen
                                    would you prefer to
                                    travel:\n1.Today\n2
                                    .Tomorrow\n",socket);//caleb wrote
                                    request int and string
                                   if
                                    (checkIfAvailableSeats
                                    (newDayOfTravel,
                                    nextCustomer
                                    .numberOfTravelers,socket,shm_ptr) ==
                                    true){
                                           customersMods =
                                            freeCustomersSeatsInSharedMem
                                            (customersMods,socket,0
                                            ,shm_ptr); //using customers
                                            old dayOfTravel and booked
                                            seats, frees customers
                                            seats,updates their
                                            bookedSeats[]
                                           customersMods.dayOfTravel =
                                            newDayOfTravel;
                                           displayAvailableSeats
                                            (customersMods
                                            .dayOfTravel,customersMods
                                            .numberOfTravelers,socket
                                            ,shm_ptr);
                                           sem_wait(writer);
                                           customersMods =
                                            selectAvailableSeats
                                            (customersMods,socket
                                            ,nextCustomer
                                            .numberOfTravelers,shm_ptr);
                                           sem_post(writer);
```

```
                                //send dayOfTravelChanged
                        }else{
                                //send sorry not enought seats
                                 available on this day
                        }

                case 3: //change number of travelers
                       //
                        displayTicketInfo
                        (ticketNumber,socket); //to show them
                        current number of travelers chosen
                        numberOfTravelersRequested =
                         requestInt("\nHow many total
                         travelers are you
                         requesting\n",socket);//caleb wrote
                         request int and string
                        if (numberOfTravelersRequested >
                         customersMods.numberOfTravelers){
                                addedTravelers=
                                 numberOfTravelersRequested -
                                 customersMods
                                 .numberOfTravelers;
                                if
                                 (checkIfAvailableSeats
                                 (customersMods
                                 .dayOfTravel,addedTravelers
                                 ,socket,shm_ptr)== true){
                                        displayAvailableSeats
                                     (customersMods
                                     .dayOfTravel
                                     ,addedTravelers,socket
                                     ,shm_ptr);
                                        sem_wait(writer);
                                        customersMods =
                                     selectAvailableSeats
                                     (customersMods,socket
                                     ,addedTravelers,shm_ptr);
                                     //optionally can use
                                     cutomerMods
                                     .numberOftravel, which
                                     would still be to let you
                                     know which bookedSeats
                                     index to start write
                                     writing to
                                        sem_post(writer);
                                        customersMods
                                     .numberOfTravelers =
                                     numberOfTravelersRequested
                                     ;
                                }
                        } else if (numberOfTravelersRequested
                         < customersMods.numberOfTravelers){
                                travelersToRemove =
                                 customersMods
                                 .numberOfTravelers -
                                 numberOfTravelersRequested;
                                sem_wait(writer);
                                customersMods =
                                 freeCustomersSeatsInSharedMem
                                 (customersMods,socket
                                 ,travelersToRemove,shm_ptr);
                                 //this also updates the
                                 customersMods struct with
                                 removed seats and returns
                                 this struct
                                sem_post(writer);
                                customersMods
                                 .numberOfTravelers =
                                 numberOfTravelersRequested;
                        }
                        break;
                }
                sem_wait(writer);
                modifyReservation(customersMods,server_name,socket);
                 //will use customerMods.ticketNumber to search,
                 commits modification to summary files, adds note at
                 end saying which server made modificaitons
                sem_post(writer);
                sendReceipt(customersMods,socket,server_name);


        case 4: //cancelReservation  //writing to summary file needs
         to be synchronized
                if (confirmCancellationMenu(socket) == true){
                 //confirm cancellation menu
                        ticketNumber = ticketInquiryMenu(socket);
                         //will ask for ticket
                        customersMods =
                         retrieveCustomersInfo(ticketNumber); //will
                         retrieve customer info from summary file
                        //displayTicketInfo(ticketNumber,socket);
                         //display ticket info to customer
                        sem_wait(writer);
                        freeCustomersSeatsInSharedMem
                         (customersMods,socket,0,shm_ptr); //uses
                         customer struct properties dayOfTravel and
                         bookedSeats[] to find and free seats in
                         shared memory
                        cancelReservation(customersMods,socket);
                         //using customers info .dayOfTravel and
                         .bookedSeats[], cancel reservation by
                         deleting from summary files
                        sem_post(writer);
                        //message customer know reservation cancelled
                }
                break;

            case 5: //exits progrom, closes socket
                //send "exit" code via tcp, for client to read
                exitReturnType = exitProgram(socket,shm_ptr,shm_fd);
                //function returns and thread is returned to server's
                 threadpool
                return exitReturnType;

            default: //this is probably redundant
                //send not a valid input message
                trainTicketMaster
                 (socket,server_name,shm_ptr,shm_fd,reader,writer);
                 //recursvie call

            }
        }


    }
```