Hacettepe University - Ankara

Programming Assignment 2 BBM203 Report

Deadline : 04/12/2019
Advisors : R.A Alaettin UÇAN

Written by:

Berk Karaimer -b21827541
Departement of Computer Science

Academic Year: 2019-2020

Problem Definition :

In this assignment we are supposed to implement a passenger ticket selling program using stack and priority queue dynamically.

Solution and Functions:

-main:

Calls file_to_string function with args[1]. This functions reads all the characters in a file and simply puts them in a string. This string is then split by \r\n characters and each piece is fed to process_line function.

After all the pieces of the string is done, it is freed. The output file is closed and all the flights and passengers are freed.

-process_line:

Takes the first word of the line and saves it as order.

if order is addseat:

Reads the flight name, class and seat count from the line. Calls the add_seat function which adds the requested amount of seats to given flight. If the flight does not exist then it will be created and the classes other than the one given to us will have 0 seats.

if order is enqueue:

Reads the flight name, class, passenger name and priority from the line.
Adds the passenger to the given flight's wanted seat class' queue.

if order is sell:

Reads the flight name from the line. Calls the sell function on the wanted flight. This function will be explained further.

if order is close:

Changes the given flight's closed flag to 1.
Prints the info for passengers in the waiting queue.

if order is report:

Calls the report function which prints the passengers who bought tickets in each class of the flight.

if order is info:

Finds the passenger with get_passenger function. Prints the name of the passenger, the name of his/her flight, the class of seat he/she originally wanted to buy from and the class he/she ended up buying, if any.

if neither an error will be raised.

-sell_tickets:

Firstly current quotas of each class is calculated (originally added - sold from that class). These are kept in an int array of three where first is standard quota, second is economy quota and last is business quota.

Then all the passengers in waiting queue are popped, and pushed either back into waiting or sold queues.
When pushing them back priority is not checked.
Between popping and pushing back:
passenger's wanted class is found. Calculated as (passenger->priority + 1)/3.
If there is quota left in that class the passenger is pushed to sold queue and quota is decreased. Passenger's purchased_seat_class is set to wanted class.
If no quota is left then it is pushed back to waiting queue to see if it can buy from standard quota.

After they have all been popped and pushed remaining standard seats are sold. At this point if there is some quota left for standard all those waiting should be business or economy as all standard passengers had tried to buy from standard quota earlier.

As long as there is quota and there are passengers in the queue, passengers are added to the sold queue, this time purchased_seat_class is 0, index of standard.

push:
Firstly the base conditions are checked.
If the queue is empty, passenger (given as an argument) is set as both front and the rear of the queue.
If the first element of the queue has a lower priority than passenger and priority is being checked, passenger is set as the front of the queue.

Otherwise which element of the queue must come before the passenger is figured out.
This is done by iterating through the linked list until the next element will either be null or if we are checking priority has a lower priority than passenger.

Passenger is then added to this element's next_in_queue after the passenger's next_in_queue has been set to that of the element's.
If the element was at the rear of the queue then passenger is assigned as the new rear.

Entity Relations:

-Passenger:

name: Name of the passenger.

flight: Shows the name of the flight this passenger requested a ticket from.

priority: The priority with which passenger will have in the waiting queue. 0 for standard, 2 for ordinary economy, 3 for veteran, 5 for ordinary business 6 for diplomat. Wanted class can be extrapolated from that.

purchased_seat_class: The class of seat passenger ended up buying from. 0 for standard, 1 for economy, 2 for business.

next_in_queue: Points to the passenger that comes after this one in a queue(waiting or sold).

next_in_database: Points to the passenger that was added to the system after this one.

-Queue:

front: Pointer of the passenger at the front of the queue.

rear: Pointer to the passenger at the end of the queue.

size: How many passengers the queue holds.

-Flight:

name: Name of the flight.

standard_seats: Number of standard seats that had been added to the flight over the course of the program's lifecycle.

economy_seats: Number of economy seats that had been added to the flight over the course of the program's lifecycle.

business_seats: Number of business seats that had been added to the flight over the course of the program's lifecycle.

-Showing how much unsold tickets are left in their respective classes.

is_closed: Can be 0 or 1. 1 if the flight is closed, 0 otherwise.

QUEUE* waiting_queue: Priority queue of all the passengers waiting to buy from this flight.

sold: Non priority queue of the passengers who had bought seats from this flight.

prev: Points the next object in the stack that was added before this one.

-Heads:

first_passenger: Pointer to the first passenger that was added to the system.

last_passenger: Pointer to the last passenger that was added to the system.

last_flight: Pointer to the last flight that was added to the system.