BBM 203 PROGRAMMING ASSIGNMENT 2 REPORT

İBRAHİM KOZ 21786303

1. Problem Definition

In this assignment, I was expected to implement a flight ticket sales automation program in C language by using stack and priority queue data structures.

2. Solution Approach and Functions

Firstly I created structs for flight, passenger and priority queue.

- Each flight has a flight name; business, economy and standard seats are actually passenger arrays; business, economy and standard priority queues.
- Each priority queue has a passenger array holds waiting passengers inside it.
- Each passenger has passenger name, passenger class, wanted seat class and place.
- I take advantage of Binary Heaps to implement priority queue.

If the command is add seat, firstly I checked if the flight has been created. If it does not exist, I created a new flight in ‘init_new_flight' function. Then I invoked add_seat function to reallocate the appropriate seat array of the flight by given value if the flight is not closed.

If the command is enqueue, firstly I created a new passenger by using the information that has been given. Then if the flight is not closed, I added the passenger binary heap’s end then a operation is called floating is processed. So whenever the function invokes, the appropriate binary heap reorder.

If the command is sell, firstly I checked how many seats can be sold in each class via `how_many_seat_can_be_sold()` . Next, I’ve sold subsequently business tickets, economy tickets and lastly standard tickets according to procedure. I’ve sold a ticket the passenger that obtained processing dequeue operation is supposed to take away head of the appropriate binary heap. Step by step: I replaced the last element with head, then sinking operation is invoked to reorder heap via `transform_to_min_priority_que()` .

If the command is close, I changed the flight’s ‘is_closed’ boolean attribute as ‘true’, so the flight became only readable. Then I printed the passengers which are still in the queue.

If the command is report, I printed the flight’s name, number of passengers of each stack and the passengers’ names for each class.

If the command is info, firstly I traversed all seats and queues inside flights to find the passenger. Subsequently I printed the passengers informations contains name, flight name, wanted class and which class did he/she get.

Lastly I read the input file and parsed each line by ‘strtok’ function. I kept every token and the number of tokens in parsed_line struct and called my functions according to them. Finally I free flights struct array includes all flights, all the seat stacks and queues and their attributes which every flight has.

UML DIAGRAM IS BELOW: