Hacettepe University

BBM 203

Software Laboratory I

ASSIGNMENT 2

Fatih OKSUZOGLU

21328284

1-Problem Definition

In order to solve the problem, we are expected to have more detailed information about C language by using priority queue and stack structure. In problem solving, we need to connect the two structures to each other and realize the necessary scenario.

2-Methods and solution

I read the data in the input file with realloc and split it line by line.I have created of flights and passengers arrays as the leading number of lines.I sent lines and other parameters to the command function.

Addseat: When defining the Addseat command, I defined all the features of the flight before adding seats to any flight.I have assigned seat classes separately for each flight with stack structure. I've added seats if the flight already exists.

Enqueue: If the arriving passenger is not in the queue before, I first added the passenger to the array. I created the priority queue in this function. Then I placed all the passengers according to the priority and the ticket class he wanted to buy. I did comparing up to the front passenger one by one. I called the swap function until I saw someone with the same priority or a higher priority. Of course, this swap cycle is up to the front index.

It is also a function that pops a passenger who cannot be placed and pushes it to the end.

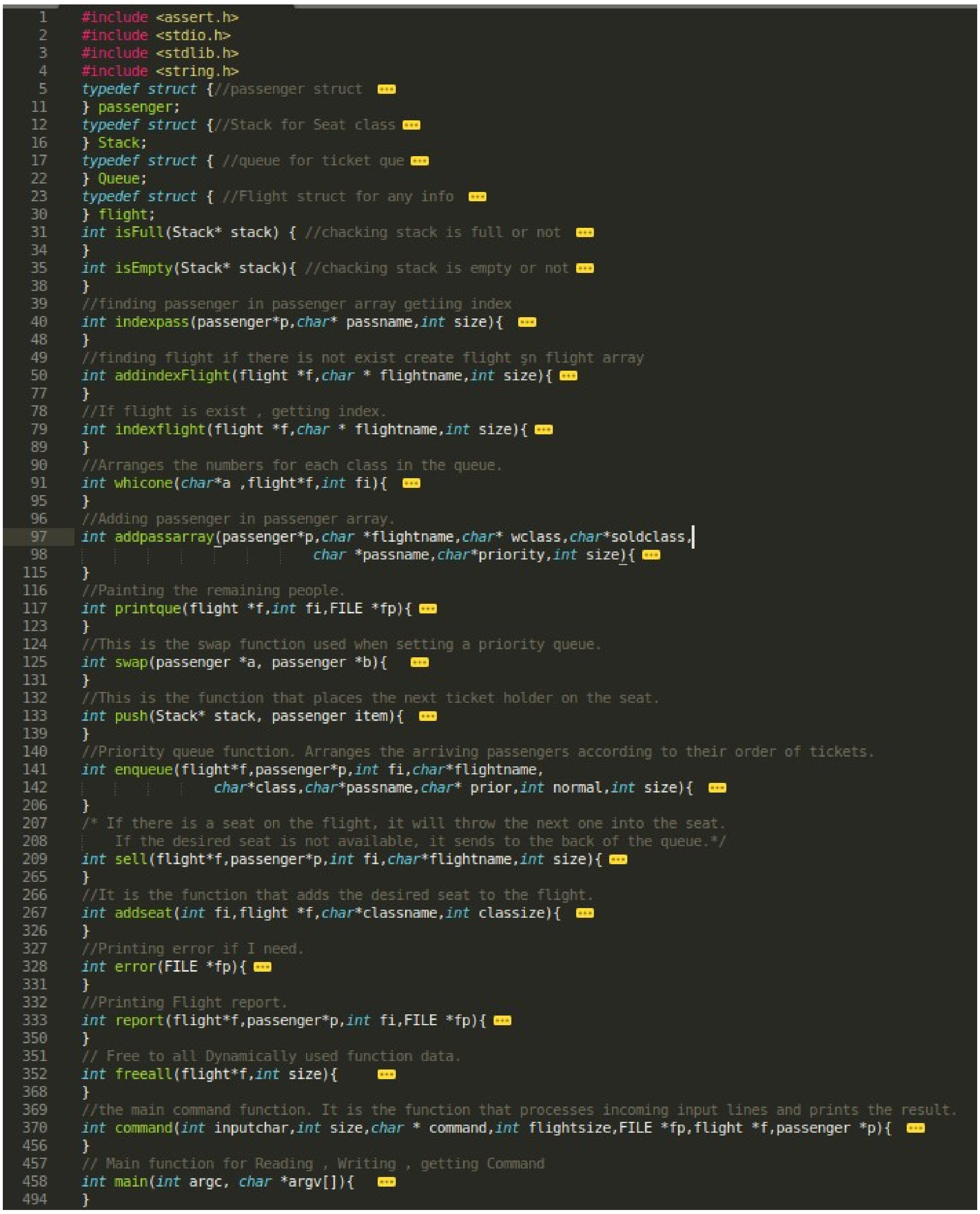
Sell: When performing this command, it is necessary to establish a connection between the queue and the stack.I put the people in the queue in the seats on the plane.I pushed until the stacks were filled with the passengers I made pop out of the queue.If the seat is full and there are still passengers who want that class in the queue before the pop I did then push.,

Close: The close command holds an int value in flight, which is not closed if it is 0, and the addseat and sell commands are executed. But if it's 1, they won't work. This function makes this value 1.

Report: The report command gives information about the flight as to who is sitting in the seats. In fact, in a way, it is the content and dimensions of the stacks that I keep in the flight structure.

Info: I searched the given passenger in the passenger array and suppressed information.

3-Functions implementation



```c
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct {//passenger struct
} passenger;
typedef struct {//Stack for Seat class
} Stack;
typedef struct { //queue for ticket que
} Queue;
typedef struct { //Flight struct for any info
} flight;
int isFull(Stack* stack) { //chacking stack is full or not
}
int isEmpty(Stack* stack){ //chacking stack is empty or not
}
//finding passenger in passenger array getiing index
int indexpass(passenger*p,char* passname,int size){
}
//finding flight if there is not exist create flight şn flight array
int addindexFlight(flight *f,char * flightname,int size){
}
//If flight is exist , getting index.
int indexflight(flight *f,char * flightname,int size){
}
//Arranges the numbers for each class in the queue.
int whicone(char*a ,flight*f,int fi){
}
//Adding passenger in passenger array.
int addpassarray(passenger*p,char *flightname,char* wclass,char*soldclass,
                            char *passname,char*priority,int size){
}
//Painting the remaining people.
int printque(flight *f,int fi,FILE *fp){
}
//This is the swap function used when setting a priority queue.
int swap(passenger *a, passenger *b){
}
//This is the function that places the next ticket holder on the seat.
int push(Stack* stack, passenger item){
}
//Priority queue function. Arranges the arriving passengers according to their order of tickets.
int enqueue(flight*f,passenger*p,int fi,char*flightname,
                char*class,char*passname,char* prior,int normal,int size){
}
/* If there is a seat on the flight, it will throw the next one into the seat.
    If the desired seat is not available, it sends to the back of the queue.*/
int sell(flight*f,passenger*p,int fi,char*flightname,int size){
}
//It is the function that adds the desired seat to the flight.
int addseat(int fi,flight *f,char*classname,int classize){
}
//Printing error if I need.
int error(FILE *fp){
}
//Printing Flight report.
int report(flight*f,passenger*p,int fi,FILE *fp){
}
// Free to all Dynamically used function data.
int freeall(flight*f,int size){
}
//the main command function. It is the function that processes incoming input lines and prints the result.
int command(int inputchar,int size,char * command,int flightsize,FILE *fp,flight *f,passenger *p){
}
// Main function for Reading , Writing , getting Command
int main(int argc, char *argv[]){
}
```

REFERENCES

[1] “Dynamic Memory Allocation in C”.

https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/ [Accessed: Nov 22, 2019].

[2] “Dinamik Bellek Yönetimi”.

http://www.cemdemir.net/c-programlama-dili/dinamik-bellek-yonetimi-321.html

[Accessed: Nov 24, 2019].

[3] “Dynamically allocate a 2D array in C”.

https://www.geeksforgeeks.org/dynamically-allocate-2d-array-c/ [Accessed: Nov 24, 2019].

[4] “Priority Queue implementation”.

https://www.programiz.com/dsa/priority-queue [Accessed: Nov 09, 2019].

[5] “Stack implementation in C”.

https://www.geeksforgeeks.org/stack-data-structure-introduction-program/ [Accessed: Nov 25, 2019].