In-lab assignment for week 12

Objectives:

Queue, and Big-O concept

Data structure:

We'll assume the following three structs in this lab:

```
typedef struct studstruct{
    int age;
    struct studstruct *nextPtr;
}Student; // this is the basic "node" struct in your queue

typedef struct linkedlist{
    Student *head, *tail;
    int numOfItems; // this member indicates the number of "Student" structs in the queue
    int numOfOdd; // indicates how many "Student" there are whose age is an odd number
    int numOfEven; // indicates how many "Student" there are whose age is even number
}StudentList;// this is the "second-level" struct as your prelab

typedef struct queuelist{
    StudentList* list;
}Queue; // this is the "upper-level" struct as your prelab
```

Requirements:

In this lab, seven functions need to be implemented using the following function prototypes:

1. Queue initQueue();

This function initializes an empty "Queue". Note: all members in your empty queue should be properly initialized in this function, including "list", "head", "tail", "numOfItems", "numOfOdd", and "numOfEven".

2. int enQueue(int, Queue);

This function receives the current “Queue” and an integer value. It creates a “Student” struct with the member “age” equals to the input integer value, and enqueues this struct into the current “Queue”. It also updates the “numOfItems”, “numOfOdd“ and “numOfEven” members. This function returns 1 if the insertion succeeds, or -1 if the insertion fails. Note: this function should have O(1) computational complexity.

3. Student deQueue(Queue);*

This function receives the current linked-list, and dequeues the earliest “Student” struct out of the current “Queue” and returns the pointer to the dequeued struct. It also updates the “numOfItems”, “numOfOdd“ and “numOfEven” members. This function returns NULL if no struct can be dequeued. Note: this function should have O(1) computational complexity.

4. int getQsize(Queue);

This function returns the total number of “Student” structs in the current “Queue”. Note: this function should have O(1) computational complexity.

5. int getNumOfEven(Queue);

This function returns the total number of structs whose “age” values are even numbers in the current “Queue”. Note: this function should have O(1) computational complexity.

6. void printQueue(Queue);

This function prints out all the “age” values from the head of the list to the tail of the list.

7. void emptyQueue(Queue);

This function frees all the previously allocated memories and sets the current “Queue” to empty, meaning the “list” pointer in the “Queue” should be freed as well.

Note: The provided .h header file should NOT be modified or submitted. If you have any custom functions, you need to write them in your .c header file. Your .c header file is the ONLY file you need to submit through Canvas. An example of header files can be found as lab 7 solution on Canvas.

Grading Criteria:

initQueue function: 5 points

enQueue function: 8 points

deQueue function: 8 points

getQsize function: 2 points

getNumOfEven function: 2 points

printQueue function: 2 points

emptyQueue function: 3 points

General notes:

1. Command to compile your code in cmd/powershell window: gcc main.c labx.c -Wall -Werror
2. If your code couldn't compile with "-Wall -Werror" flags, you will receive an automatic 0 grade.
3. Changing the given .h header file will lead to an automatic 0 grade.
4. Using any global variables will lead to an automatic 0 grade.
5. Function implementation should include comments describing what it is intended to do and how this function should be called. Examples can be found in lab solutions. Failing to do so will result in a point deduction.