Homework 4, CSE 1320

Task1 (100 pts)

Files to submit: `wait_in_line.c` (source code) and `wstudent.txt` (input file created by student, for testing)
Objective: implement linked-list queues and use them. Create your own test file.
Write an entire program that simulates students waiting in line to see one of two advisors. The program will ask the user for a file name and process all the requests in the file.
The first line of the file has the column labels and should be skipped.
The next N lines consist each of:
`ID advisor time Name`

- `ID`s will be consecutive numbers from 1 to N
- `advisor` is a number 1 or 2 indicating that the student will want to see advisor 1 or advisor 2.
- `time` is a number indicating how much time that student will spend with the advisor
- `Name` is the student's first name. It will have at most 12 letters in it.
- A line in the file will have at most 50 symbols (excluding the new line symbol).

You should process the data as follows:

- For each line in the file, except the first one, create a node and add it to the queue for the corresponding advisor: q1 for advisor 1, q2 for advisor 2.
- Print the lines side by side
- Remove the students in the order in which they FINISH. To compute the finish time, add the time of the students at the head of the queue to the total time of all the students seen so far by that advisor.
- Repeatedly remove from the 2 queues until they are empty.
- Print the number of students that each advisor had seen.

Requirements:

- Implement the queues using singly linked lists (not arrays).
- A node in the linked list must contain as information ***at least*** the `ID, time`, and `Name`. You can add more members.
- Put all the code in a single file.
-
- Match the sample output. Print the same text, use the same spaces and especially the same alignment. For your reference I included a line with numbers to help you count.

```
12345678901234567890123456789012345678901234567890
 Line at Desk 1         | Line at Desk 2
 Time   ID Name         | Time   ID Name
    3    1 Jane         |    5    2 Luis
  ……
Processing queues:
12345678901234567890123456789012345678901234567890
leave   ID Name
    3    1 Jane
```

Hints and tips:

1. Be mindful that one queue may be empty and the other not. This can happen when:
    a. Printing the queues side-by-side
    b. Deciding which node to remove (which student finishes first: from `q1` or from `q2`).
2. You can choose the members of the linked list nodes and the queue struct. For example you can add a member for the finish time either in the list node or in the struct for the queue.

To calculate the finish time (when a student leaves the advisor's desk) use the sum of the time when the student before it finished, and the current student's time.
For example if we look at :

```
Line at Desk 1          | Line at Desk 2
 Time   ID Name         | Time   ID Name
    3    1 Jane         |    5    2 Luis
    7    3 Sam          |    4    5 Bob
    3    4 Alice        |    2    7 Mary
    4    6 Will         |
```

For desk 1 we will have:

- Jane will leave at time 3 (she is first in line)
- Sam will leave at time 10 (= 3+7 = time when Jane left + time Sam needs)
- Alice will leave at time 13 (= 10+3 = time when Sam left + time Alice needs)
- Will will leave at time 17 (= 13 + 4)

For desk 1:

- Louis will leave at time 5
- Bob will leave at time 9 (5+4)
- Mary will leave at time 11 (= 9+2)

And they will leave in order of their finish time so we get:

```
Processing queues:
leave   ID Name
    3    1 Jane
    5    2 Luis
    9    5 Bob
   10    3 Sam
   11    7 Mary
   13    4 Alice
   17    6 Will
```

Grading:

- 10 - Coding style: indentation, name, program description, variable names, comments
- 15 - Students are correctly added to the queue for the corresponding advisor. If queues are not printed at all, this cannot be checked and no credit is given.
- 15 - Print queues side-by-side (if queues are printed one below the other, 10 points are lost)
- 15 - Students leave in correct order of their finish time and the leave time is printed and is correct for each student.
- 8 - Print the data in each queue aligned as shown in the sample run.
- 10 - All the required information is printed in the given format both when printing the queues and when printing the students leaving (e.g. student IDs are printed as well, not just the name as multiple students may have the same name).
- 4 - Prints correct the total number of students processed by each advisor (at each desk).

- 8 - other correctness
- 15 - No Valgrind errors (no memory leaks or any other errors)

Penalty:

- 50% of the score is lost if the queues are not implemented using linked lists or if arrays are used.

Submission and Penalties

*Place the **`wait_in_line.c`** and **`wstudent.txt`** files in a folder called **hw4_Lastname**. Zip this folder and submit it to Canvas.*

Penalties

1. ****** Code using elements we had not covered at the time the homework was due, receives no credit.**
2. Code must compile and run on a Unix system. Code that does not compile is not graded and earns a score 0.
3. Code must compile WITHOUT any warnings.
4. File(s) for each task must be named as shown.
5. Up to **10** points will be lost for non-compliance with the submission requirements: folder name, all files in a folder, zipped folder, the compressed file is a zip, the program files have extension
6. Each program must have:
    - description and your name at the top
    - proper indentation
    - meaningful variable names
    - comments

Sample runs:

See more sample runs in Canvas: w2.txt, w20.txt, w100.txt and their sample runs: w2_out.txt, w20_out.txt, w100_out.txt

Sample run

```
Enter filename: w1.txt

 Line at Desk 1          | Line at Desk 2
 Time   ID Name          | Time   ID Name
    3    1 Jane          |    5    2 Luis
    7    3 Sam           |    4    5 Bob
    3    4 Alice         |    2    7 Mary
    4    6 Will          |

Processing queues:
leave   ID Name
    3    1 Jane
    5    2 Luis
    9    5 Bob
   10    3 Sam
   11    7 Mary
   13    4 Alice
   17    6 Will


Total requests processed at Desk 1: 4
Total requests processed at Desk 2: 3
```