**CSC2302**

**Data Structures using C**

**Spring 2021**

**Project2**

**To Read CAREFULLY:**

---

- This is your second project which is meant to test your understanding of:

**Queues implemented using linked list!**

- This is a work that you need to do in a group of two: **every** member need to get involved in every detail
    - **So far I have an idea about the programming skills of every student. Any suspect about any code submitted will implies calling the student for a meeting to present his/her work**
- You assume the responsibility of the code that you will submit as I will assign an immediate WF grade for any form of plagiarism
- You need to submit your work by email **Friday April 16th** before midnight
    - The work has to be sent by **one member** with a CC to project partner
    - You need to submit 1 file which is your .C file
    - .C file should be named as follow: Student1Name_Student2Name.c
- Any work received Saturday April 17th starting 00:01 will not be accepted
    - You will get a zero as project grade
    - Don't wait till the last minute to submit your work

---

**Let's get started**

During particular weeks of the semester, I am receiving many students during office hours: I receive students that are showing in my office to discuss issues related to a course I am teaching, but I also receive advisees that need to finalize their schedule for the coming semester. Students showing in my office are asked to stand in one of the following two queues: **advising or course_discussion** and for fairness reasons, I try to alternate between serving a student who need

assistance in one my courses, if any, then serve an advisee seeking help to choose courses, if any! Obviously the two queues are managed in a FIFO way.

Let's work with the following menu:

- Assist to Mrs.Talei office hours
- Mrs Talei saying: "Coming In"
- Office Hours are Over
- Change Queue
- Quit

As usual, every menu choice including the menu and excluding the last choice should be implemented using a user defined function. Your code must have at least the following standard function:

1. Enqueue(): this function will be used to add a student to a particular queue: **advising or course_discussion**
    **a.** Make sure that you define this function according to the discussion we had in class and you may adapt it to the requirement of this project
2. Dequeue() this is a function that you will use to release a student from a particular queue:
    **a.** Make sure that you define this function according to the discussion we had in class and you may adapt it to the requirement of this project
3. Create_fill_node(): a function that you will use to create and fill a node.
    **a.** Every student will be represented with a name and ID

**Important Points to consider (Part1):**

1. For the first menu choice, a new student shows up to use office hours either for advising or course material discussion. Once a student node is created you need to enqueue the student to the appropriate queue
    a. Please note that your main function should display a message informing the student about which queue he/she is currently belonging to
2. For the second choice, I am calling for serving a student! To remember, for fairness reasons, I will start serving first a student willing to discuss course

material (if any) then an advisee (if any), then a student willing to discuss course material(if any)….

   a. Please make your main function print a message about which student go served, if queue is empty….

3. For the third choice, office hours are over! That means that the two queues should be destroyed- in case they contain any student!
   a. Your function should add the name and ID of any left student in a file called to_serve_later.txt
4. For the fourth choice, sometimes, I can have a student in a course I am teaching and at the same time the student is also an advisee. Assume that a student planned for a visit to discuss course material then he/she decided to switch to discussing courses for next semester, or vice versa
   a. Remember queues should be managed in a FIFO way: when a student get dropped from a queue to join another one, he/she need to join the other queue in a FIFO way!
   b. Quitting a queue implies searching for the student who want to change the queue, if he/she exist, detach his/her node from the former queue to join the new one!

**Important points to consider (Part2):**

a. There is no need to create two different functions doing the same job
b. File should be opened and closed properly and correctly
c. You are not allowed to use global variables in this project
d. You loose points if you use any input/output function (such as scanf(), printf()….) inside a function while it should be an input/output argument
e. Remove from the main() function the maximum details you can
f. Make sure that you use DMA and free the memory that you no more need

**Best of Luck,**