CS 6103D Software Systems Laboratory

PROBLEM 1B

The objective is to learn the following:

- implementation of list using pointers (as a singly linked list)
- implementation of queue using pointers (as a doubly linked list)
- maintaining multiple header and source files
- use of make utility.

Submission date: on or before 03-09-2022 Saturday 11.59 PM
Submission: as a single *.tar* file named as per the format
$P1B\_ < FIRSTNAME > \_ < ROLLNO > .tar(eg : P1B\_ARUN\_M180xxxCS.tar)$

Modify the program developed for problem 1A as follows:

1. For each course, provide the option for registering students in the course. For this, it is required to maintain a list of names of students who have registered for each course with provisions for adding and deleting names. Implement this list as a *singly linked list* and keep a pointer to the head of the list as an additional field, named *regList* in the *course struct*. Define functions *insert()* to insert a name to the front of the list and *delete()* to delete a given name from the list.

2. Maintain the *regList* in sorted order. For this modify the *insert()* function such that it inserts every new name in its correct sorted position in the list. Keep the function definitions for list operations in a separte file named *list.c*, with the related type definitions and prototypes in a header file named *list.h*.

3. For each course, the course faculty decides how many students can register for the course. For this, add a field *maxLimit* to *struct course* indicating the maximum number of students that can register for a course. Once the number of students registered reaches *maxLimit*, any further request for registration is kept in a waiting list. Add another field *waitList* to *struct course* which points to the head of the waiting list. Whenever a student drops a course, the name of that student is removed from *regList*, a student from *waitList* is removed in FIFO (First In First Out) order and added to the *regList*. This requires implementing the *waitList* as a queue of names with operations *enQueue()*, *deQueue()*, and *isEmptyQueue()*. Use *doubly linked list* for queue implementation. Name the header file *queue.h* and the implementation file *queue.c*.

4. Now your project consists of five separate files. Create a *makefile* for building the executable.