Project 9, Program Design

1. (60 points) In project 8, a t-shirt is added to end of the linked list. Each t-shirt was stored with the student organization name, size, price, and quantity in inventory. In this project, you will modify the add_to_inventory function so a t-shirt is added to an ordered linked list. Name your file *tshirt_store2.c*.

    a. Instead of adding to the end of the linked list, a t-shirt is added to an ordered linked list **by student organization name and then by size**. The list remains ordered after the t-shirt is added.

    For example, a t-shirt for Association for Computing Machinery of size XS should be before a t-shirt for Society of Competitive Programmers of size L; a t-shirt for Association for Computing Machinery of size XS should be after a t-shirt for Association for Computing Machinery of size L.

2. (40 points) Modify project 7 so that it uses quick sort to sort the states by percentage of population whose ages are equal or greater than 65 years old. Instead of using the sorting function you wrote for project 7, use the quick sort library function and implement the comparison function. Name your file *demographics2.c.*

Testing guidelines:

1. Download the files *try_tshirt_store2* and *try_demongraphics* from Canvas and upload them to the **student cluster (sc.rc.usf.edu)**. Change the file permissions of the test script with the following command:

    chmod +x try_tshirt_store2

    chmod +x try_demographics

2. Compile and test your solution:

    gcc -Wall -std=c99 tshirt_store2.c

    ./try_tshirt_store2

    gcc –Wall demographics2.c

    ./try_demographics

Submission instructions:

Download the program *tshirt_store2.c* and *demographics2.c* from student cluster and submit it on Canvas>Assignments->Project 9.

Grading

Total points: 100

1. A program that does not compile will result in a zero.
2. Runtime error and compilation warning 5%
3. Commenting and style 15%
4. Functionality 80%
    a. Function implementation meets the requirements
    b. Function processes the linked list using **`malloc`** and **`free`** functions properly

Programming Style Guidelines

The major purpose of programming style guidelines is to make programs easy to read and understand. Good programming style helps make it possible for a person knowledgeable in the application area to quickly read a program and understand how it works.

1. Your program should begin with a comment that briefly summarizes what it does. This comment should also include your **name**.
2. In most cases, a function should have a brief comment above its definition describing what it does. Other than that, comments should be written only *needed* in order for a reader to understand what is happening.
3. Variable names and function names should be sufficiently descriptive that a knowledgeable reader can easily understand what the variable means and what the function does. If this is not possible, comments should be added to make the meaning clear.
4. Use consistent indentation to emphasize block structure.
5. Full line comments inside function bodies should conform to the indentation of the code where they appear.
6. Macro definitions (#define) should be used for defining symbolic names for numeric constants. For example: **#define PI 3.141592**
7. Use names of moderate length for variables. Most names should be between 2 and 12 letters long.
8. Use underscores to make compound names easier to read: **tot_vol** or **total_volumn** is clearer than totalvolumn.