Procedure-Oriented Programming, Fall 2016

Homework Assignment #2

Due midnight Wednesday, November 9, 2016

Instructions

1. If any question is unclear, please ask for a clarification.
2. You are required to do all the homework assignments on Linux. To ensure that your C program is also a C++ program, you are required to use both gcc and g++ version 4 or later to compile your program.
3. You are encouraged to make sure that your program can be compiled by MFC, but it is not required.
4. You are required to give your TA a demo of your program. Make sure that your program can compile and run on the server machine, which will be used for the demo.
5. For the program that you write, you are required to include a Makefile. Otherwise, the grade for your program will be zero.
6. Unless stated otherwise, you are required to work on the homework assignment individually.
7. No late homework will be accepted.

Programming Project

The purpose of this homework assignment is again to get you acquainted with the GNU make and the modular design of a *large* program in a procedure-oriented programming language, C.

The requirement of this assignment is as follows:

- First, you are required to implement a whole bunch of modules each of which consists of two files: one for the interface (i.e., the so-called `.h` file) and one for the implementation (i.e., the so-called `.c` file). The details are as follows:
    1. First, you are required to implement a doubly linked list of `char`, `short`, `int`, `long`, `float`, and `double` and a doubly linked list of pointer to these types.[1]

[1] Although not required, you are encourgaed to implement a doubly linked list of "whatever structure" that you can imagine and a doubly linked list of "pointer to whatever structure."

2. Next, you are required to build a "stack" module on each doubly linked list that you implement in step 1.
3. Then, you are required to build a "queue" module on each doubly linked list that you implement in step 1.
4. Finally, you are required to *use*[2] the "mm" module, which you implemented for the first homework assignment, for memory management. Repeated, the "mm" module acts as the memory manager by wrapping up the functions `malloc`, `calloc`, `realloc`, and `free` as defined in the standard library. One way to wrap up these functions is to add a prefix to the name of these functions so that `malloc` is named `mymalloc`, `calloc` is named `mycalloc`, and so on. Also, if you have not done so yet in the first homework assignment, this time you may want to use `mymalloc` instead of `malloc`, `mycalloc` instead of `calloc`, and so forth.

Taking into account the interface and implementation, you are supposed to have at least the following files:

- `list_<type>.h`,
- `list_<type>.c`,
- `stack_<type>.h`,
- `stack_<type>.c`,
- `queue_<type>.h`,
- `queue_<type>.c`,
- `list_<ptr_to_type>.h`,
- `list_<ptr_to_type>.c`,
- `stack_<ptr_to_type>.h`,
- `stack_<ptr_to_type>.c`,
- `queue_<ptr_to_type>.h`,
- `queue_<ptr_to_type>.c`,
- `mm.h`, and
- `mm.c`,

where `<type>` denotes `char`, `short`, `int`, `long`, `float`, and `double`; `<ptr_to_type>` denotes pointer to `char`, `short`, `int`, `long`, `float`, and `double`. Moreover, it is up to you to define the interface of each module and to hide as much as possible the implementation of each module.

- Then, you are required to write drivers to test all the modules that you design and implement in the previous step. You may name them
  - `main_stack_<type>.c`,
  - `main_queue_<type>.c`,
  - `main_stack_<ptr_to_type>.c`, and
  - `main_queue_<ptr_to_type>.c`,

  and assume that the input is a list of `<type>` each of which is on a line by itself, and so is the output.
- Finally, you are required to write a Makefile—which contains at least three targets: **all**, **dep**, and **clean**—to manage the project.

Of course, you are encouraged to reuse all the modules and drivers that you wrote for the previous homework assigmments. You are also encouraged to prepare for the changes that may require you to implement all the stack and queue modules again but in a different way.

[2]Or if you like, you can call it *reuse*.

Grading Policy

The grading policy for this assignment is as follows:

- This assignment accounts for 10 points to your final grade.
- Make sure that a **Makefile**, which contains at least three targets—**all**, **dep**, and **clean**—is provided. Otherwise, the grade for your program will be zero.
- 8 points if your program compiles and runs without errors and warnings.
- 2 points if the program is properly modularized and well structured.

Gentle Reminder

1. If you have never had experience on using Linux, start earlier. It may take you quite a while to get used to it.
2. If you have never had Linux installed on your system, it is time to get it installed.