Programming Assignment 3

Due by 10월 11일 오후 9시

1. For this programming assignment, we implement the address book with a linked list. Implement `add()`, `delete()` and `print_list()`, and submit "backend.c". DO NOT CHANGE ANYTHING ELSE!

2. Make sure to read all the source files provided, especially "backend.h" and "memory.h" and of course "backend.c", and understand the structure of the program.

3. A newly `add()`ed record are stored **at the front** of the linked list. As always, `data` is the pointer to the linked list that stores the data for the address book.

4. Duplicate names are allowed; a search gives the first record found, and a deletion deletes the first record found.

5. The functions `new_node()` and `free_node()` are implemented in "memory.c," but we provide it only in compiled form as "memory.o". **Since it is linux-binary, it works only in linux.**

   The memory management needs an initialization, hence we have `init()`, which calls `init_pool()`. We will discuss the memory management in the next class and the next programming assignment. Note that `POOL_SIZE` is defined to be 10 in "memory.h".

   Recall that `new_node()` returns `NULL` if it cannot allocate a node. When `add(name)` is called, if an overflow occurs (`new_node()` returns `NULL`), then `add()` prints a message "`Can't add. The address book is full!`" and our program waits for a new command.

6. Note that we don't have `search_index()` this time. Why??? Functions `search()`, `delete()` and `print_list()` are supposed to examine the nodes one by one from the first node until the task is accomplished.

7. Starting from this assignment, we directly deal with pointers. Be very careful.

8. Submit with the following command:

   5분반: `submit dsta hw3a`,
   6분반: `submit dsta hw3b`.

   To submit kth assignment, you are supposed to use either "`submit dsta hw`k`a`" or "`submit dsta hw`k`b`".