Programming Assignment 4

Due by 10월 5일 오후 9시

1. For this programming assignment, we implement the address book with a linked list, together with our own memory management as discussed in the class. Implement `new_node()`, `add()`, `delete()` and `print_list()`, and submit "backend.c" and "memory.c". DO NOT CHANGE ANYTHING ELSE!

2. **IMPORTANT:** The records are stored in the linked list **in an alphabetical order**. This will be a little harder to implement than Assignment 3. Use `compare()` to decide the order between two records. Other than `add()`, it behaves the same way as Assignment 3. So, you can reuse `delete()` and `print_list()` of Assignment 3.

3. When a duplicate name is added, the newly added record is inserted in front of the other (old) records with the same name. (Why? Because that's easier :)

   Again, a search gives the first record found, and a deletion deletes the first record found.

4. For the dynamic memory management, as discussed in the class, a chunk of memory is allocated using an array `pool[POOL_SIZE]` and it is intialized with `init_pool()` to make it another linked list. And `new_node()` and `free_node()` do the memory management. Note that `POOL_SIZE` is defined to be 10. When `add(name)` was called, if an overflow occurs, then `add()` prints a message "`Can't add. The address book is full!`" and our program waits for a new command.