Programming Assignment 1: Linked Lists

Due: ~~Monday, Sept 18 by 11:59PM~~
Tuesday, Sept 19 by 11:59PM

Key Concepts:

- Linked-Lists
- Memory allocation and deallocation in C
- Compilation with Multiple Files
- Runtime (a little)
- Recursion

Background Reading:

- Section 1.7 of the Zybook (runtime)
- Chapter 2 of the Zybook (lists)
- Chapter 3 of Aho/Ullman (runtime)
- Chapter 6 of Aho/Ullman (runtime)
- Stanford C pointer tutorial
- Stanford linked-list tutorial

Summary: you have been given a C implementation of a list ADT in which some functions are already written and some are not. Your job is to complete the unwritten functions. (In one case you will make modifications to improve the runtime of one already implemented function).

Examine the files `list.h` and `llist.c` in the `src` directory where you found this handout.

You will discover that it is a partially implemented linked list "module" or ADT.

The lists store numeric values (the type of which can be changed by altering the `typedef` for `ElemType` in `list.h`). The idea here is to allow some degree of flexibility in what a list stores -- however, for the purposes of this assignment, you don't need to worry about this. Just continue to use `ElemType` as `int`.

The header file `list.h` file gives the interface for an ADT while the actual implementation is given in the `llist.c` file. The members of `list_struct` are also "hidden" in the `.c` file. The ADT defines many natural operations on lists -- some of these have already been implemented and will be used as motivating examples during lecture; others have not been implemented: It is your job to do the implementation! Look for `TODO` labels throughout the files.

A subtle detail: why did I decide to name the header file `list.h` (one `'l'`), but the implementation file `llist.c` (two `'l's`)???

Your Job: completion of all of the `TODO` items specified.

Rules:

You cannot modify `list.h`

Exceptions: if you want to fiddle with different ElemType (e.g., double) OR you want to add new sanity checkers to call externally.

All of your "real" work is in `llist.c` (except testing code).

Discussion: The given linked list structure has two "levels":

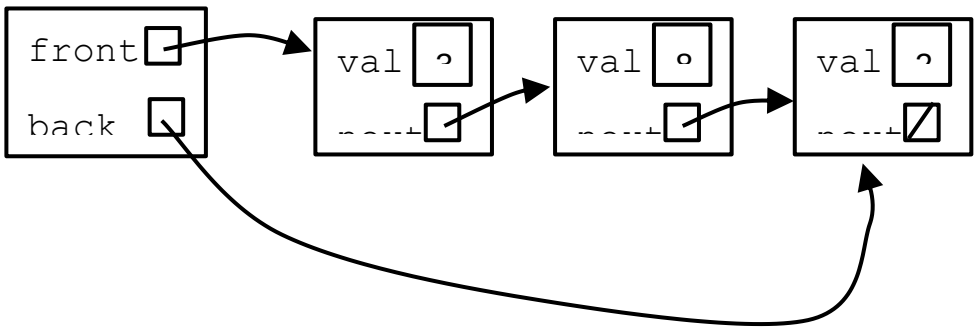
At the "lowest" level are the linked-list nodes themselves specified as:

```
typedef struct node {
    ElemType val;
    struct node *next;
} NODE;
```

However, the type `NODE` isn't even visible to a client program. Only the type `LIST` is visible to a client (just the type -- not the struct members). Through the header file, `LIST` is equivalent to a `struct list_struct` which is specified as follows:

```
struct list_struct {
    NODE *front;
    NODE *back;
};
```

Here is a diagram of a list with three entries: `<3, 8, 2>`. The `struct` at the left (a `LIST`) gives access to the actual nodes.



There are 15 functions tagged with the word TODO in both list.h and llist.c. The points for each function is given in the table below.

Detailed descriptions of the requirements of each function are given in banner comments above the functions themselves.

(Note that the unwritten functions already have placeholder "stubs" for them (i.e., the functions technically already exist, but their bodies are empty).

Function	Points
lst_are_equal	10
lst_count	10
lst_length	15
lst_pop_back	15
lst_print_rev	15
lst_insert_sorted	15
lst_concat	15
lst_clone	15
lst_from_array	15
lst_to_array	15
lst_reverse	20
lst_prefix	20
lst_filter_leq	20
lst_merge_sorted	20
lst_remove_all_fast	20

The grand total is 240 possible points.

Miscellaneous Stuff

Makefile: you have been given a basic makefile along with the source files. There is also another handout which reviews the following:

Manual compilation of ADTs like used in this assignment using gcc

Crash course on makefiles and the make program in the context of this assignment.

Testing: A major portion of your work will be devoted to testing your implementations. You have been given a bare-bones "driver" program in `ll_tst.c` which gives you a model of the sorts of things you should be thinking about.

Submission Details:

Your submission will be through blackboard and will include the following:

```
list.h (which actually should not have changed)
llist.c (this is the biggie!)
tester programs
makefile (you may extend the given makefile if you want),
readme (but only if there is something unusual about your submission
that you want to point out to the TAs)
```

NOTES:

As suggested, you are expected to submit your tester programs. Your tester programs will not contribute to your score, but we want to be able to give you feedback on it.

Approximately 4 days before the assignment is due, we will release a suite of test cases. This will take the form of a driver program and they will be a ***subset*** of the test cases we will use for grading your submission.

The idea is to encourage you to think carefully about testing -- ideally, the test cases given will be redundant for you because you've already done similar tests; but if not, it is a bit of a wakeup call.

A somewhat trivial driver program has been given in `ll_tst.c`; we've also included a baseline makefile which you can modify.