Assignment 01 - Refresher on C programming

Objectives

1. Get a hang of the programming and compilation tools that will be used throughout the semester.
2. Acquire good habits for assignment submission.

Foreword

This first assignment was deliberately designed to be very easy to solve. If you finish it fast, all the better. If you have trouble answering some of the questions, then it is imperative that you brush up your C development skills.

Submission Format

When submitting your work, please stick to the conventions described below.
You will prepare a working directory that will contain the following:
- `bin` directory: empty directory for all the executable files created by editing the links between your object files
- `include` directory: directory containing all of your header files (* .h headers)
- `lib`: directory containing all of your pre-compiled libraries (* .a)
- `obj` directory: empty directory for all the object files (* .o) created by compiling your source files
- `src`: directory containing all of your source code files (*.c)
- `Makefile`: file containing all of your compiler directives
- `README.txt` file: plain text file that reports on your work submission

Of all the elements in the working directory, the `README` file is the most important. It shall contain:
- Your first name, your last name, and your NYU ID
- The detailed list of files you submitted in the directories `include`, `lib`, and `src` (for each file, list the questions it provides a solution for)
- The explanation of the compilation rules provided in your `Makefile`
- Comments about the work you submitted, if any (what does not run, what is encoded but produces errors at compile / execution, what feature is incomplete, ...)
- Textual answers to some questions in the set (eg. Q7 in this set)

Before submitting your working directory, please rename it like so:
`[lastname-firstname-nyuid].os.[assignment#]` (*eg. marin-olivier-ogm2.os.01*)
archive it in TAR format and compress it in GZIP format. The submission should be a single file:
`[lastname-firstname-nyuid].os.[assignment#].tgz` (*eg. marin-olivier-ogm2.os.01.tgz*)
This is the file you attach to your submission ***before*** the deadline.

Question Set

Question 1

Download the coding canvas for this assignment, and compile/run the program by calling the make command in the terminal.

Question 2

Using an array is not satisfactory; we would prefer to use a doubly linked list.
Complete the `list_impl.c` file code to:
- Extract an element from the head (the associated cell is removed from the list and the memory it occupied is deallocated)
- Extract an element from the tail (the associated cell is removed from the list and the memory it occupied is deallocated)
- Compute the number of items in the list

Question 3

Write a file called `stack_list.c` that uses the primitives from `list.h` to build a dynamic stack that implements `stack.h`
Add a compiler directive in the makefile to build a new library `libstack.a` from `stack_list.c` and `list_impl.c`
Recompile an executable from the stack test program (`stack_test.c`) and use your library to verify that it works correctly.

Question 4

Write a file called `fifo_list.c` that uses the primitives from `list.h` to build a dynamic queue that implements `fifo.h`
Add a compiler directive in the makefile to build a new library `libfifo.a` from `fifo_list.c` and `list_impl.c`
Recompile an executable file from the test program (`fifo_test.c`) and use your new library to verify that it works correctly.

Question 5

Traversing the entire list to determine the number of items is too expensive. What changes should you make, and in which file(s), to determine the size of the list in O(1)? *Write your answer in your `README.txt` file*.

Notes

1. Both Q3 and Q4 require you to build a library with the same name: libstack.a
   The difference is that the library must implement the stack with an array when compiling for Q3, with a linked list when compiling for Q4.
2. A few pointers to help you handle your coding projects in C
   - How to Compress and Extract Files Using the tar Command on Linux (how to extract the starting code and compress your solution)
   - How to Use Linux's ar Command to Create Static Libraries
   - Makefile Tutorial