Assignment 2: mysort

Date: October 31st 2017
Deadline: November 17th 2017 23:59

Objectives

You must implement a list API and a multi-tool number sorting program.

Requirements

Your sorting program must be named `mysort` and its basic operation is as follows:

- it reads input numbers from its standard input;
- it prints the same numbers in sorted order on its standard output, one number per line.

Numbers in the input are separated by white space. Your program must perform sorting by maintaining a sorted list in memory and maintaining this order while reading the input (insertion sort).

You must submit your work as a tarball [1]. Next to the source code, your archive must contain a text file named "AUTHORS" containing your name and Student ID.

Getting started

1. Unpack the provided source code archive; then run `make`.
2. Try out the generated `mysort` and familiarize yourself with its interface.
3. Read the file `list.h` and understand the interface.
4. Implement the data structure in `list.c`.
5. Implement the input and insertion sort algorithm in `main.c`.

Testing

A small set of tests is provided for both the list data structure and the algorithm. The file `check_list.c` contains a set of testcases for the data structure functions [2]. The script `check_sort.sh` comes with a very basic set of tests for your sorting algorithm. To run these tests type `make check`. This command first runs the data structure tests and if these all pass it runs the algorithm tests.

The test sets provided are not complete. Add your own tests to make sure that your implementation of the data structure and the algorithm is correct.

Grading

Your grade starts from 0, and the following tests determine your grade:

- +0,5pt if you have submitted an archive in the right format with an `AUTHORS` file.
- +0,5pt if your source code builds without errors and you have modified `list.c` or `main.c` in any way.
- +2pt if your list API processes insertions and removals properly.
- +3pt if your `mysort` processes and sorts its input properly and terminates with exit code 0.
- -0,5pt if your program misbehaves on zero-sized inputs.
- -0,5pt if your program misbehaves when the last line does not terminate with a newline character.

- -1pt if `valgrind` reports errors while running your program. Valgrind and the address sanitizer don't play well together so temporarily remove the `-fsanitize=address` flags from the Makefile when testing your code with `valgrind`.
- -1pt if `clang -W -Wall` reports warnings when compiling your code.

The following extra features will be tested to obtain higher grades, but only if you have obtained a minimum of 5 points on the list above already:

- +1pt if your `mysort` accepts option `-u` which causes it to eliminate duplicate output lines.
- +0,5pt if your `mysort` accepts option `-S` which calculates the sum of all the items and appends that to the end of the list.
- +0,5pt if your `mysort` accepts option `-s` taking a single argument, which causes it to only consider input lines multiple of that number.
- +0,5pt if your `mysort` accepts option `-x` taking a single argument, which causes it to ignore any input line that are multiple of that number.
- +0,5pt if your `mysort` accepts option `-h` taking a single positive number *N* as argument, which causes it to stop after outputting the first *N* numbers of the sorted list.
- +0,5pt if your `mysort` accepts option `-t` taking a single positive number *N* as argument, which makes the program output the last *N* numbers of the sorted list.
- +0,5pt if your `mysort` accepts option `-3` which causes it to replace all occurrences of the numbers 51, 69 and 42 by 666.

See also

- Insertion sort: https://en.wikipedia.org/wiki/Insertion_sort
- Parsing program options: https://www.gnu.org/software/libc/manual/html_node/Getopt.html

---

1 `make tarball` will create the tarball for you.

2 These tests are written with the unit test library `check`. If this library is not yet installed on your system, run `sudo apt install check` to install it.