ENCE360 Lab 1: C Review

Objectives

The overall goal of this lab is to revise your experience with simple C programs. The subsequent labs and assignments build upon this knowledge and assume you have a strong familiarity with the topics here. It is an opportunity to assess your knowledge in C and put some time to brush up on the basics. Note that quiz questions for this week are based on the topics covered in the lab.

The aims of this lab are for you to revisit:

1. Using pointers
2. `structs` and `typedefs`
3. Memory allocation
4. Strings
5. Data structures
6. Function pointers

Preparation

As well as this handout, `lab1Revision.zip` should contain the files `vector.c`, `buffer.c`, and `linkedList.c`.

For this lab, it will suffice to compile with the command

```
gcc <program>.c -o <program> -g -Wall -Werror
```

where `<program>` is the filename of what you want to compile.

Run the executable produced with

```
./<program>
```

Once you've completed the programs below, make sure you also complete the lab quiz. These may need you to have finished code ready, so do the programming first! The quizzes are what give you the lab marks, but working through the PDFs is what will make you learn the content and be ready for the test/exam. Be sure to do both.

Program `vector.c`

This is an exercise in memory allocation and pointers. Your goal here is to implement two functions: `vector_new()` and `vector_add()`. `vector_new()` should allocate space for a vector. `vector_add()`should use `new_vector()` to create space for a sum vector, and populate it with the correct sum value.

Program `buffer.c`

This is an exercise in memory management and handling strings. String handling can be awkward in C because all memory needs to be manually allocated and

free'd. Handling binary data is even more tricky, and just using string functions to handle it won't work, since these all expect a null terminator `\0`.

The goal here is to implement `copy_buffer()` which copies a buffer of binary data.

Program `linkedList.c`

An exercise in data structures and function pointers and memory management. Your task is to write a function `map_list()` which takes a linked list and a function pointer and returns a new linked list in the same order, with the values in the new list transformed by the function passed in.

The second task is to write a function to clean up, to free a linked list. Memory management can be a pain in C, but all languages (and operating systems) rely on this type of manual allocation. Write a function to free a linked list and any of the memory allocated in it's creation by implementing `free_list()`.

Test your modified program for memory leaks using Valgrind. As a reminder, you can run a program under Valgrind with the command

```
valgrind --leak-check=yes ./<program>
```

Make sure you compiled with the `-g` flag to get full benefit!

Program `numSort.c`

This file should be empty when you unzip it. As a final exercise, write a program from scratch that uses `qsort()` (`man qsort` for details) to sort an array of `int32_t`. As practice, try doing this only by using the man page; you'll need to get used to reading them!