Kyle Chen

# Results of Timing Experiment

The output of the timing experiment from my computer:

```
Starting stopwatch for SelectionSort()...
[alleged … zip]
Program took 24 cycles and 0.000024 ms to complete.

Starting stopwatch for InsertionSort()...
[alleged … zip]
Program took 15 cycles and 0.000015 ms to complete.
```

The insertion sort was faster, because the average time complexity of insertion sort is better. Selection Sort has a time complexity of O(n^2) for all linked lists because it has to parse through each element, even if the list is already sorted. Insertion sort has a worst case of O(n^2), but a best case of O(n), meaning that for the vast majority of linked lists, which are partly sorted, insertion sort will perform faster than selection sort. This result was what I expected.

The output of the timing experiment from the board:

```
Starting stopwatch for SelectionSort()...
[alleged … zip]
Program took 243 cycles and 0.243000 ms to complete.

Starting stopwatch for InsertionSort()...
[alleged … zip]
Program took 170 cycles and 0.170000 ms to complete.
```

The board is much slower than the computer, and is also less efficient with its cycles.

# Summary of Lab

We learned a lot of pointer manipulation, and how linked lists work, which are both fundamental topics in computer science. We also learned about a few sorting algorithms, namely insertion and selection sort, and also the time complexity of sorting. We used this knowledge to implement a linked list program with a testbench, and wrote sort functions to operate on linked lists.

## Approach to Lab

I wrote out the linked list functions fully because I am familiar with linked list implementation. Then, I wrote out the testbench enough to make sure the linked list works. After that I wrote the sort functions, then tested it with the main lab function. After that was done, I cleaned up the testbench output and added a few more tests to make it comprehensive, while discovering a couple of bugs in NULL parameter handling. Then I ran it on the nucleo and fixed some bug with printf and NULL parameters.

The lab went very well, there was little confusion and any bugs were quickly found and patched.

## Implementing the Lab

I think I ended up with a fairly robust program. I spent about 3 hours on it, most of that working on the test functions and trying to make that comprehensive and readable. The tests were probably the hardest part, as I am still not sure what the best way to test the linked list program, from an organizational and methodical standpoint. I feel like my current testbench is a bit cumbersome to read.

The lab was fine, grading was fine, very worthwhile and people learn a lot. Lab manual and examples during class covered the material in enough detail, although the visuals for selection sort in the manual are clunky. I found that online resources were much clearer. Only note is that the usleep() function used in Lab06_main.c is obsolete and should probably be replaced with nanosleep().