CPRE 1850 – INTRODUCTION TO FUNCTIONS

LAB 3
SECTION CN

SUBMITTED BY:

MAXWELL MILLER

9/26/2025

Problem

This lab requires the use of organization, print formatting, and abstraction through functions, to build the most effective and readable programs. For this lab we use the Dual Shock controller in as an input device and print a variety of formatted outputs in each section. Each of these parts are designed to test our knowledge on metamathematical operators in C, along with our implementation of a variety of functions.

Analysis

This lab requires use of several problem-solving techniques, particularly in integer division and functional abstraction. The lab was clearly designed to practice in these two key areas.

Design

My implementation of the criteria set forth by this lab was relatively straight forward. I did decide to have functional prototypes, even though it was technically not necessary, it makes more cleaner, readable, code.

Testing

Each file was run multiple times with a variety of inputs (if applicable). These test admirably arbitrary tests indicated that the programs ran as they should, but without a proper rigorous unit test it is unknown what limits these files may have. Despite this, all files ran correctly given the sample inputs from the lab 3 outline.

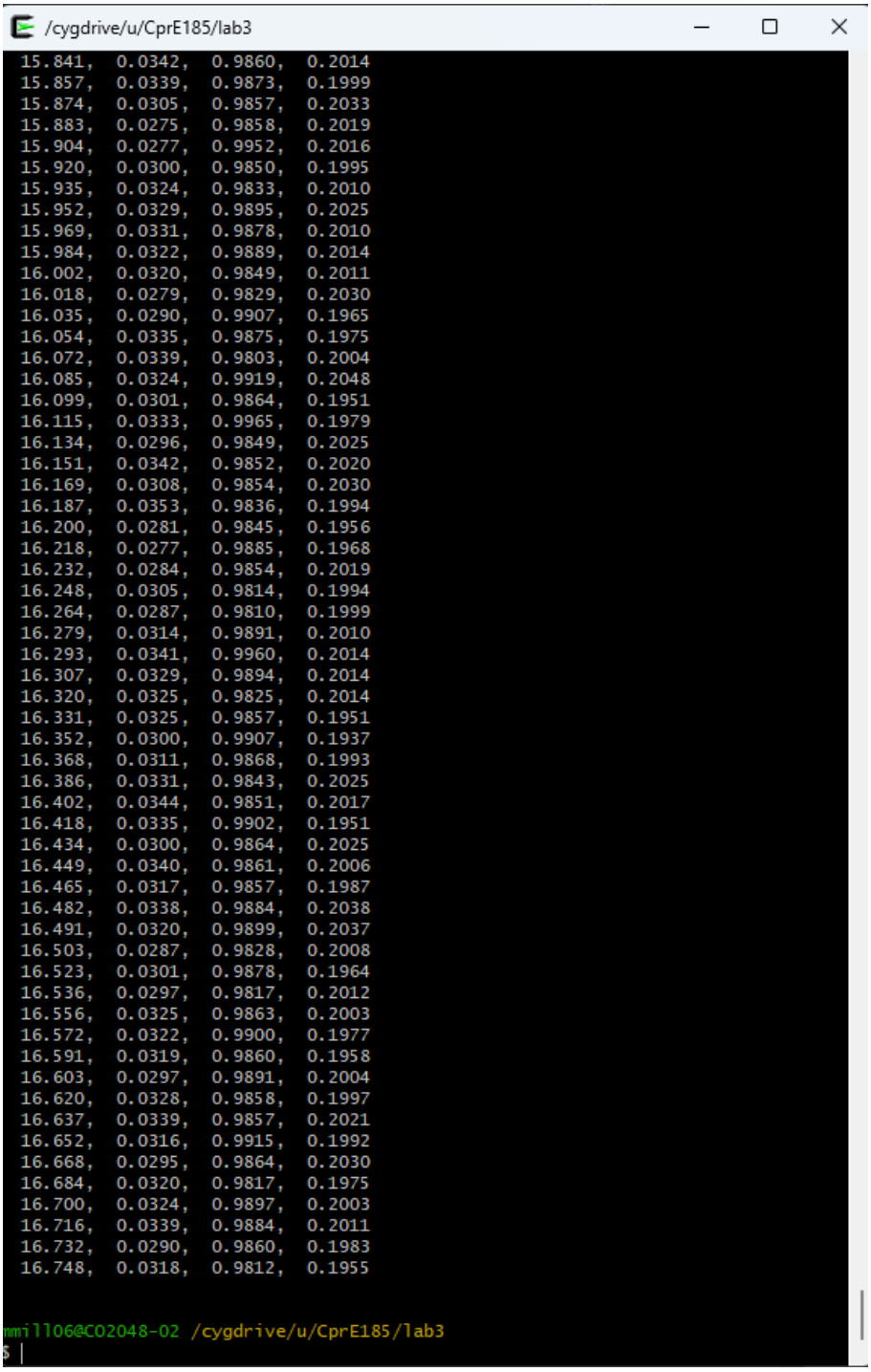
Comments

This lab was not particularly challenging given my vast prior programming experience, but it did provide me an opportunity to remind myself of the unique attributes of the C programing language. It also allowed me to refresh my knowledge on integer division and functional prototype syntax.

Part 1 – Code Section 0

```
/* 185 Lab 3 */

#include <stdio.h>
#include <math.h>


const int MILLIS_IN_MIN = 60000;
const int MILLIS_IN_SEC = 1000;

int main(void) {
	int t;
	double  ax, ay, az;

	while (1) {
		scanf("%d,%lf,%lf,%lf", &t, &ax, &ay, &az);

		printf("Echoing output: %d, %lf, %lf, %lf\n", t, ax, ay, az);
		printf("%8.3lf, %7.4lf, %7.4lf, %7.4lf\n", t / MILLIS_IN_SEC, ax,
ay, az);
	}

return 0;
}
```

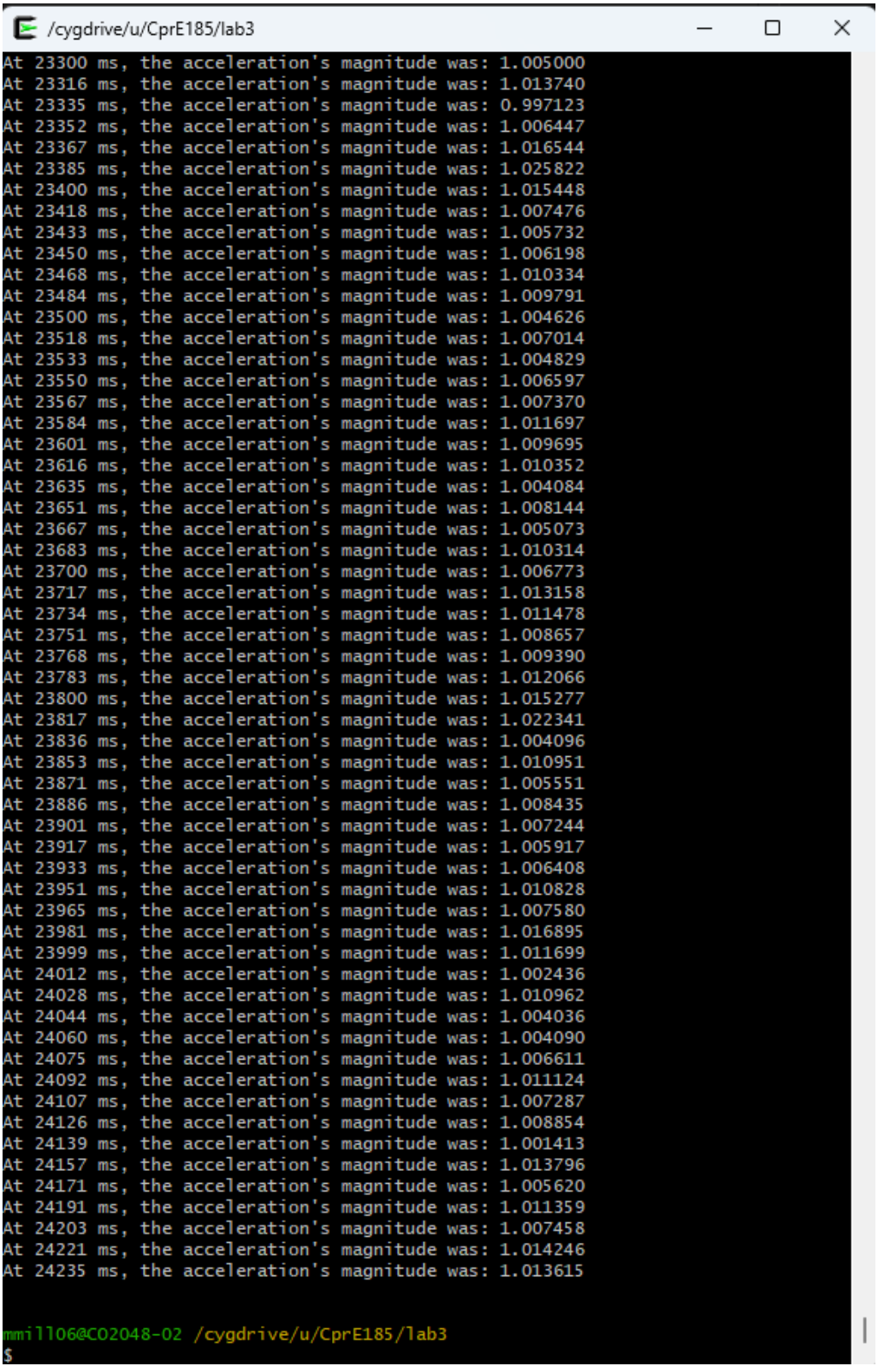
Part 1 – Output



Part 2 – Code Section 1

```c
/* 185 Lab 3 */

#include <stdio.h>
#include <math.h>


const int MILLIS_IN_MIN = 60000;
const int MILLIS_IN_SEC = 1000;

double mag(double ax, double ay, double az);

int main(void) {
	int t;
	double  ax, ay, az;

	while (1) {
		scanf("%d,%lf,%lf,%lf", &t, &ax, &ay, &az);
		/* CODE SECTION 1 */
		// printf("At %d ms, the acceleration's magnitude was: %lf\n", t,
mag(ax, ay, az));
	}

return 0;
}

double mag(double ax, double ay, double az) {

	// returns the vector acceleration magnitude
	// ie. sqrt(a^2 + b^2 + c^2)
	return sqrt( ( ax * ax ) + ( ay * ay ) + ( az * az ) );
}
```

Part 2 - Output



Part 3 – Code Section 2

```
/* 185 Lab 3 */

#include <stdio.h>
#include <math.h>


const int MILLIS_IN_MIN = 60000;
const int MILLIS_IN_SEC = 1000;

int minutes(int t);
int seconds(int t);
int millis(int t);


int main(void) {  int t;
      double  ax, ay, az;

      while (1) {
            scanf("%d,%lf,%lf,%lf", &t, &ax, &ay, &az);

            printf("At %d minutes, %d seconds, and %d milliseconds it was:
%lf\n", minutes(t), seconds(t), millis(t), mag(ax,ay,az));

      }

return 0;
}

int minutes(int t) {
      return t / MILLIS_IN_MIN;
}


int seconds(int t) {
      return t % MILLIS_IN_MIN / MILLIS_IN_SEC;
}


int millis(int t) {
      return t % MILLIS_IN_MIN % MILLIS_IN_SEC;
}
```

Part 4 - lab3_2.c

```c
/* 185 Lab 3-2 */

#include <stdio.h>
#include <math.h>


/* Put your function prototypes here */
int numButtons(int a, int b, int c, int d);


int main(void) {
	/* DO NOT MODIFY THESE VARIABLE DECLARATIONS */
	int t;
	int a,b,c,d;


	/* This while loop makes your code repeat. Don't get rid of it. */
	while (1) {
		scanf("%d,%d,%d,%d", &a, &b, &c, &d);
		printf("Buttons pressed: %d\n", numButtons(a,b,c,d));

		fflush(stdout);
	}

	return 0;
}

/* Put your functions here */
int numButtons(int a, int b, int c, int d) {
	return (a + b + c + d);
}
```

Output:

```
Buttons pressed: 0
Buttons pressed: 0
Buttons pressed: 0
Buttons pressed: 0
Buttons pressed: 0
Buttons pressed: 0
Buttons pressed: 0
Buttons pressed: 0
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 2
Buttons pressed: 2
Buttons pressed: 2
Buttons pressed: 2
Buttons pressed: 2
Buttons pressed: 2
Buttons pressed: 2
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 2
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 1
Buttons pressed: 0
Buttons pressed: 0
```