**Lab3.Ogbondah**

**LAB 3**

**SECTION SE 185**

**Chimzim Ogbondah**

**SUBMISSION DATE: 9.18.2018**

**DATE: 9.18.2018**

# Problem

1. The purpose of part one of this lab was to be able to modify the ds4rd.exe code source in Cygwin to the specifications needed. Then rotate the controller to see the different output values based on the orientation of the controller.
2. The purpose of part two of this lab was to be able to manipulate the outputting values to me our decimal requirements and understand how to use more complex functions. This way the same equations don’t get used over and over.
3. The purpose of part three of the lab is to implement our own functions, so we understand truly how functions work as well as how to call them respectively so they work with our program.

# Analysis

1. The problem for part one states that we must run the **./ds4rd.exe -d 054c:05c4 -D DS4\_BT -t -g command** three different times for three different orientations of the controller. Therefore, the three different commands in Cygwin will be > flat.csv, >front.csv and costum.csv. Once the CSV files are created the only reasonable graph to show the data is to use a scatter plot for the many different points to show the trend. (was gathering data in milliseconds)
2. The problem for part two states that we must compute the time from milliseconds to seconds for section 0. Compute the right accelerations’ magnitude for x, y, z and finally created a function that work with section 2 which would print out the correct values. Therefore, the problem first consisted of converting t from milliseconds to seconds. Then adding in functions because converting all the time over and over would waste time as well as the accelerations’ magnitude. The only reasonable equation to use for the first conversion was t / 1000 because this process would waste time we defined the prototype function above the main function and below calling for int minutes (int t) and dividing t by 60000 to get the seconds, int seconds(int t) by 60000 and then modulus 1000 so it only prints out the seconds column, and int millis(int t) modulus 60000 and modulus 1000 to only get the milliseconds column.
3. The problem for part three states that we compute the total number of buttons being pressed on the PS34 controller (Square, triangle, circle, x). Therefore, the problem input consisted of the values given to the buttons above by the controllers input using scanf() statements and the output was the printf() statement taking on the sum of all the buttons being pressed and then telling the user the number of buttons being pressed. The only reasonable equation was to use a function by defining the variables that were going to be used putting the prototype function above and then the function below the main function. Inside the function setting the sum of Square, Triangle, X , and Circle = to a variable and then inside the main function set the value inside the printf() statement equal to that function call.

# Design

1. Our problem was to run the program creating 3 different CSV files named based off their controller orientation and then opening the files and using a scatter plot to show how the data looks. I used a simple decision process in which I ran the command three separate times each time renaming the csv file, and then going on to make the graphs from the csv file.

Using the basic outline above I met the criteria and correctly moved the orientation of the controller. To ensure I did this correctly I looked over the instructions form the Lab.

1. Our problem was to compute the conversion from milliseconds and turn them into second and minutes. As well as using an equation to calculate the accelerations’ magnitude which was as well put into a function
   1. Convert t in section 0 to t / 1000
   2. Define functions for minutes, seconds, mag (ex. Int minutes(int t)
   3. Sqrt(pow(ax, 2) + pow(ay, 2) + pow(az, 2)) for the mag
   4. minutes = t / 60000
   5. seconds = t % 60000 / 1000
   6. milliseconds = t % 60000 % 1000
      1. This implies that the prototype was given above the main function and each variable was define, with function given somewhere in the main function.

Using the outline above, I designed my program by using steps. I assigned all numerical values with (int) except for mag which took on a double value. I then defined the prototype for the functions above the main function and then again below with the corresponding equations to give the correct outputs inside the main function. We finally printed the results of our computation to the terminal so that way the user could see the results. To make sure that everything was correctly done I looked over the lab instructions for guidance on what confused me.

1. Our problem was to compute the sum of buttons being pressed on the PS4 controller. And then print that value out for the user to see. I used a step by step process to breakdown and solve the problem
   1. I used a function to simplify my math where I ended up calling the prototype function above and then below my main unction
   2. Inside the function I set a variable equal to the sum of all the controller buttons added together
   3. Inside my main function I called the function inside my printf() statement so it output the total number of buttons being pressed.

Using the outline above, I used steps to solve my problem. I used scanf() statements to give the values into the corresponding variables I defined. I then used a printf() statement to inform the user of how many buttons were being pressed, and this worked through the function below adding all the values of the buttons together. I looked over the instructions to ensure that I coded the program correctly.

# Testing

1. In order to verify the results of the command in Cygwin was correct I thought about the values I should be getting in my head and then asked to of the TA’s to look over my graphs to ensure I had done the right process.
2. To make sure the results of the solution were correct, I complied and ran the program for multiple seconds to read the different columns, so if it wouldn’t give me numbers in the columns vs when all of them would give numbers. (more than 1 minute less than one minute)
3. To be certain that my results were correct, I ran the command on Cygwin doing different combinations of button mashing (1, 2, 3, 4) to ensure that the total buttons being pressed lined up with how many I was pressing and that the order didn’t matter.

# Comments

In doing this Lab I learned about how important it is to read the instructions because I was struggling with part 3 of the lab because I couldn’t get my program to count how many buttons had been pressed. I spent a lot of time in the Lab trying to pass the problem until I read the lab instructions again and saw that I need the total number of buttons being pressed not counting. When I realized this I stopped and fixed my original code because it was giving me a max of four when all the buttons were being pressed. Knowing the instructions well would have saved me a lot of time

Part 1.

Part 2./\*-----------------------------------------------------------------------------

- SE 185 Lab 03

- Developed for 185-Rursch by T.Tran and K.Wang

- Name: Chimzim Ogbondah

- Section:SE 185

- NetID: ogbondah

- Date: September 17, 2018

-----------------------------------------------------------------------------\*/

/\*-----------------------------------------------------------------------------

- Includes

-----------------------------------------------------------------------------\*/

#include <stdio.h>

#include <math.h>

/\*-----------------------------------------------------------------------------

- Defines

-----------------------------------------------------------------------------\*/

#define TRUE 1

/\*-----------------------------------------------------------------------------

- Prototypes

-----------------------------------------------------------------------------\*/

/\* Put your function prototypes here \*/

double mag**(**double x**,** double y**,** double z**);**

int minutes**(**int t**);** /\* Defining minutes function to an integer value \*/

int seconds**(**int t**);** /\* Defining seconds function to an integer value \*/

int millis**(**int t**);** /\* Defining milliseconds function to an integer value \*/

/\*-----------------------------------------------------------------------------

- Implementation

-----------------------------------------------------------------------------\*/

int main**(**void**)** **{**

/\* DO NOT MODIFY THESE VARIABLE DECLARATIONS \*/

int t**;**

double ax**,** ay**,** az**;**

**while** **(**TRUE**)** **{**

scanf**(**"%d,%lf,%lf,%lf"**,** **&**t**,** **&**ax**,** **&**ay**,** **&**az**);**

/\* CODE SECTION 0 \*/

printf**(**"Echoing output: 8.3%d, 7.4%lf, 7.4%lf, 7.4%lf\n"**,** t**/**1000**,** ax**,** ay**,** az**);** /\* divided t by 1000 to turn milliseconds into seconds \*/

/\* CODE SECTION 1 \*/

printf**(**"At %d ms, the acceleration's magnitude was: %lf\n"**,** t**,** mag**(**ax**,** ay**,** az**));**

/\* CODE SECTION 2 \*/

printf**(**"At %d minutes, %d seconds, and %d milliseconds it was: %lf\n"**,**

minutes**(**t**),** seconds**(**t**),** millis**(**t**),** mag**(**ax**,**ay**,**az**));** /\* calls to the functions below (minutes, seconds, millseconds, and mag. then prints out their values based on the function output \*/

**}**

**return** 0**;**

**}**

/\*Fuction the calculate the acceleration's magnitude\*/

double mag**(**double x**,** double y**,** double z**)** **{**

//Step 8, uncomment and modify the next line

**return** sqrt**(**pow**(**ax**,** 2**)** **+** pow**(**ay**,** 2**)** **+** pow**(**az**,** 2**));** /\* eqaution to measure the accelerations magnitude \*/

**}**

/\* function to calculate minutes column \*/

int minutes**(**int t**)** **{**

**return** t **/** 60000**;** /\* Equation for minutes column \*/

**}**

/\* Function to calculate seconds column \*/

int seconds**(**int t**)** **{**

**return** **(**t **%** 60000**)** **/** 1000**;** /\* Equation for the seconds column \*/

**}**

/\* Function to calculate the milliseconds column \*/

int millis**(**int t**)** **{**

**return** **(**t **%** 60000**)** **%** 1000**;** /\* Equation for the milliseconds column \*/

**}**

Part 3. /\*-----------------------------------------------------------------------------

- SE 185 Lab 03

- Developed for 185-Rursch by T.Tran and K.Wang

- Name: Chimzim Ogbondah

- Section: SE 185

- NetID: ogbondah

- Date: Septermber 17, 2018

-----------------------------------------------------------------------------\*/

/\*-----------------------------------------------------------------------------

- Includes

-----------------------------------------------------------------------------\*/

#include <stdio.h>

#include <math.h>

/\*-----------------------------------------------------------------------------

------------------

-----------------------------------------------------------\*/

#define TRUE 1

/\*-----------------------------------------------------------------------------

- Prototypes

-----------------------------------------------------------------------------\*/

/\*-----------------------------------------------------------------------------

- Implementation

-----------------------------------------------------------------------------\*/

int main**(**void**)** **{**

int TRIANGLE**;** /\* Defining Triangle to int value \*/

int SQUARE**;** /\* Defining SQUARE to int value \*/

int CIRCLE**;** /\* Defining CIRCLE to int value \*/

int X**;** /\* Defining X to int value \*/

**while** **(**TRUE**)** **{**

scanf**(**"%d, %d, %d, %d"**,** **&**TRIANGLE**,** **&**SQUARE**,** **&**CIRCLE**,** **&**X**);** /\* Scanning in vaules for the TRIANGLE, CIRCLE, X, SQUARE through the PS4 controller \*/

printf**(**"Buttons being pressed: %d\r\n"**,** buttonsPressed**(**totalButtonsPressed**));** /\* statement will print the number of buttons being pressed by calling the function below \*/

fflush**(**stdout**);** /\* TA said put this in \*/

**}**

**return** 0**;**

**}**

/\*Function for summing up the number of buttons being pressed on the PS4 controller \*/

int buttonsPressed**(**int TRIANGLE**,** int SQUARE**,** int CIRCLE**,** int X**)** **{**

totalButtonsPressed **=** TRIANGLE **+** SQUARE **+** CIRCLE **+** X**;** /\* Equation to calculate the sum of buttons being pressed \*/

**}**