**Lab7.Ogbondah**

**LAB 7**

**SECTION D**

**Chimzim Ogbondah**

**SUBMISSION DATE: 10.23.2018**

**DATE: 10.23.2018**

# Problem

1. The purpose of this lab was to be able to write functions to specifications, practice using output parameters and top down designs for the program as well as knowing how to use pointers.

# Analysis

1. The problem for this lab was to finish writing function given in the source code so that the desired input gave the desired output and then once all the function were done, we were then suppose to implement the code inside of our main function so that way depending on which button was pressed it would change from roll to pitch to the left joystick. Inside each one of the calls based on the orientation of the controller or the joystick, it would then print out R’s or L’s based on if the controller was tilted right or left with respective spaces, so it would only print R’s with spaces to the right and L’s with spaces to the left.

# Design

1. Our problem was to create a program using the functions given in the source code to take the inputs given and return the desired outputs and then from there use those function inside of the main function to dictate the 3 different modes for the program, Roll, Pitch, and Joystick, printing out spaces and then R’s to the right if it was tilted right, and the L’s to the left followed by spaces to the right and if it was in the middle it was to print out a 0. It works the same way for roll being up and down and the joystick being pushed to the left of the right. To work through this task, I first started by getting all my functions to work properly and then from there I worked to get the game to work properly.
   1. I first I worked on the read\_input function which used pointers, so I called the function using the address of a regular variable, then inside of the function with pointers I used the pointers to update if the left button was being pressed by adding a star in front of it.
   2. Next, I worked on the scaleForScreenMag function. For this one I used and if else statement to see if the values being inputted were greater than 1 and if they were I set them equal to 1, and if it was less than -1 I set it equal to -1. Next, I worked on the desired returning value which was 39. For this after I took the input value and multiplied it by 39 and then returned that value. I was able to do this by looking at my equation for arcsin. (equation is stated in the questions below)
   3. For the print\_chars function I set the variable i to an int value and then I made a for loop starting with the number taken in and looping through it until it was greater than 0. Each time it looped it would print out the character given as well in the function call. This allowed me to print the R’s or the L’s or the spaces based on how many I needed to create my graph\_line function.
   4. Then I worked on the scaleForScreenJoystick function which took in the input of 127 and wanted it to be scaled to the screen and needed an output of 39. For this I took 39 and divided it by 127 and then I multiplied it by the number that was inputted inside of the function and then returned that value.
   5. Finally, the last function to work on was the graph line function which I used an if else statement to determine if the orientation of the controller. If it was left or down it would give a number less than zero and so then I called the print chars function inside of the statement printing to the specification to print to the left. If it was equal to 0 I would print 39 spaces and then a 0 and then a new line, and finally I would print 39 spaces and then R’s if it was to the right for the numbers that were greater than zero.
   6. Once all the functions were created I called the given functions with the respective input from the controller (scaleforscreenmag(x)) and then set it equal to the given variables that were for this reason (scaled\_roll, scaled\_pitch, scaled\_joystick) then I made a variable called previous and set it equal to 0. This would allow me to loop through all 3 modes using one button. I then made if statements checking the previous and was number it had if the triangle button was pressed. I then updated the pervious variable and then called the graph\_line function with the value I wanted to print inside (scaled\_roll, scaled\_pitch, scaled\_joystick).
   7. Then I made the last branch on the statement that if the X button was pressed it was to break the loop which would then result to a return 0 which would end the program.

Using the basic outline above I met the criteria from the lab instructions and the functions instructions which in the main function would print R’s and L’s based on the orientation of the controller or joystick (left one).To ensure that I did this right I compiled my code and ran it several times looking out for anomalies.

# Testing

1. To make sure the results of the solution were correct, I compiled to make sure I didn’t get any errors, and then I went back and ensured that I added comment to every line to make the code work, and what each line meant and that it made sense also looking at what the function was supposed to do. I then I ran the program to make sure it worked effectively again.

# Comments

1. (PI/2 \* return value = 39) and or (PI \* return value = 39) so I divided by Pi/2 which is the same thing as \* by 2 over Pi and then I returned that value. Which the joystick I divided 39 by 127 and then took that number and \* it by the input value and returned that number.
2. The only problem I saw was that the L’s and R’s would make like a jump back words almost like they didn’t want to print the last character

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

- SE 185 Lab 07

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

- Name: Chimzim Ogbondah

- Section: D

- NetID: ogbondah

- Date: 9.23.2018

-

- This file provides the outline for your program

- Please implement the functions given by the prototypes below and

- complete the main function to make the program complete.

- You must implement the functions which are prototyped below exactly

- as they are requested.

-----------------------------------------------------------------------------\*/ // -39 < x < 39

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

- Includes

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

#include <stdio.h>

#include <math.h>

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

- Defines

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

#define PI 3.141592653589

/\* NO GLOBAL VARIABLES ALLOWED \*/

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

- Prototypes

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

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

PRE: Arguments must point to double variables or int variables as appropriate

This function scans a line of DS4 data, and returns

True when left button is pressed

False Otherwise

POST: it modifies its arguments to return values read from the input line.

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

int read\_input**(** int**\*** time**,**

double**\*** g\_x**,** double**\*** g\_y**,** double**\*** g\_z**,**

int**\*** button\_T**,** int**\*** button\_C**,** int**\*** button\_X**,** int**\*** button\_S**,**

int**\*** l\_joy\_x**,** int**\*** l\_joy\_y**,** int**\*** r\_joy\_x**,** int**\*** r\_joy\_y **);**

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

PRE: ~(-1.0) <= mag <= ~(1.0)

This function scales the roll/pitch value to fit on the screen.

Input should be capped at either -1.0 or 1.0 before the rest of your

conversion.

POST: -39 <= return value <= 39

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

int scaleMagForScreen**(**double rad**);**

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

PRE: -128 <= mag <= 127

This function scales the joystick value to fit on the screen.

POST: -39 <= return value <= 39

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

int scaleJoyForScreen**(**int rad**);**

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

PRE: -39 <= number <= 39

Uses print\_chars to graph a number from -39 to 39 on the screen.

You may assume that the screen is 80 characters wide.

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

void graph\_line**(**int number**);**

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

PRE: num >= 0

This function prints the character "use" to the screen "num" times

This function is the ONLY place printf is allowed to be used

POST: nothing is returned, but "use" has been printed "num" times

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

void print\_chars**(**int num**,** char use**);**

/\*----------------------------------------------------------------------------- //for loop 39 spaces and then however many rs for right

- Implementation

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

int main**()**

**{**

double x**,** y**,** z**;** /\* Values of x, y, and z axis\*/

int t**;** /\* Variable to hold the time value \*/

int b\_Up**,** b\_Down**,** b\_Left**,** b\_Right**;** /\* Variables to hold the button statuses \*/

int j\_LX**,** j\_LY**,** j\_RX**,** j\_RY**;** /\* Variables to hold the joystick statuses \*/

int scaled\_pitch**,** scaled\_roll**;** /\* Value of the roll/pitch adjusted to fit screen display \*/

int scaled\_joy**;** /\* Value of joystick adjusted to fit screen display \*/

/\* Put pre-loop preparation code here \*/

**do**

**{**

/\* Scan a line of input \*/

read\_input**(&**t**,** **&**x**,** **&**y**,** **&**z**,** **&**b\_Up**,** **&**b\_Down**,** **&**b\_Left**,** **&**b\_Right**,** **&**j\_LX**,** **&**j\_LY**,** **&**j\_RX**,** **&**j\_RY**);**

/\* Calculate and scale for pitch AND roll AND joystick \*/

scaled\_roll **=** scaleMagForScreen**(**x**);** //calculates number for the roll based off of function calls with the scanned in values from the controller

scaled\_pitch **=** scaleMagForScreen**(**z**);** //calculates number for the pitch based off of function calls with the scanned in values from the controller

scaled\_joy **=** scaleJoyForScreen**(**j\_LX**);** //calculates number for the joystick based off of function calls with the scanned in values from the controller

/\* Switch between roll, pitch, and joystick with the up, down, and right button, respectively \*/

**if** **(**b\_Left **==** 1**)** **{** // switches to pitch when the square button is pressed

graph\_line**(**scaled\_pitch**);**

**}**

**else** **if(**b\_Right **==** 1**)** **{** //switches to pitch when the circle button is pressed

graph\_line**(**scaled\_roll**);**

**}**

**else** **if(**b\_Up **==** 1**)** **{** //switches to joystick when the triangle button is pressed

graph\_line**(**scaled\_joy**);**

**}**

**if(**b\_Down **==** 1**)** **{** //end the program when the X button is pressed

**break;**

**}**

// Output your graph line

fflush**(**stdout**);**

**}** **while** **(**1**);** /\* Modify to stop when left button is pressed \*/

**return** 0**;**

**}**

int read\_input**(** int**\*** time**,** double**\*** g\_x**,** double**\*** g\_y**,** double**\*** g\_z**,** int**\*** button\_T**,** int**\*** button\_C**,** int**\*** button\_X**,** int**\*** button\_S**,** int**\*** l\_joy\_x**,** int**\*** l\_joy\_y**,** int**\*** r\_joy\_x**,** int**\*** r\_joy\_y **)** **{**

scanf**(**"%d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d, %d, %d"**,** time**,** g\_x**,** g\_y**,** g\_z**,** button\_T**,** button\_C**,** button\_X**,** button\_S**,** l\_joy\_x**,** l\_joy\_y**,** r\_joy\_x**,** r\_joy\_y**);** //returns the memory address that points tha values initialized inside the main function

**if(\***button\_S **==** 1**)** **{**

**return** 1**;** //returns true if the square button is pressed

**}**

**else** **{**

**return** 0**;** //returns false if any other button is pressed

**}**

**}**

int scaleMagForScreen**(**double rad**)** **{**

double radx **=** 0**;**

**if(**rad **>** 1.0**)** **{** // sets the input to 1 if it is greater than 1

rad **=** 1**;**

**return** 39**;**

**}**

**else** **if(**rad **<** **-**1.0**)** **{** //sets the input to negative one if it is less than -1

rad **=** **-**1**;**

**return** **-**39**;** //set

**}**

**else** **if** **(**rad **>** 0.0 **&&** rad **<** 1.0**)** **{** //returns rad times 39 for positive numbers

**return** rad **\*** 39**;**

**}**

**else** **if(**rad **<** 0.0 **&&** rad **>** **-**1.0**)** **{** //returns rad for negative numbers

**return** rad **\*** 39**;**

**}**

**}**

void graph\_line**(**int number**)** **{**

**if** **(**number **<** 0**)** **{** //checks to see if the number inputed into the graph line is less than 0

print\_chars**(**39 **+** number**,** ' '**);**

print\_chars**(-**number**,** 'l'**);** // then it prints l for the left orientation

print\_chars**(**1**,** '\n'**);**

**}**

**else** **if(**number **>** 0**)** **{** //if the number is greater than zero it loops 40 times printing spaces

print\_chars**(**39**,** ' '**);**

print\_chars**(**number**,** 'r'**);** //then prints rs

print\_chars**(**1**,** '\n'**);**

**}**

**else** **if** **(**number **==** 0**)** **{** //if the number is equal to 0 then it loops 40 times printing spaces and then prints a zero

print\_chars**(**39**,** ' '**);**

print\_chars**(**1**,** '0'**);** //prints the zero after for loop

print\_chars**(**1**,** '\n'**);**

**}**

fflush**(**stdout**);**

**}**

void print\_chars**(**int num**,** char use**)** **{**

int i**;**

**for** **(**i **=** 0**;** i **<** fabs**(**num**);** **++**i**)** **{** //loops to print the character based on the abs value of the number

printf**(**"%c"**,** use**);**

fflush**(**stdout**);**

**}**

**}**

int scaleJoyForScreen**(**int rad**)** **{**

**return** **(**rad **\*** **(**39.0 **/** 127.0**)** **);** //scaled the joystick to the screen

**}**