LAB REPORT

LAB 7
SECTION C

SUBMITTED BY:

SCOTT VLASIC

SUBMISSION DATE:

MARCH 24, 2017

Problem:

The problem for this lab was that I needed to modify the skeleton code for the esplora so that when I was moving the esplora, the values would change based on its position. It would create a graph of the values as I changed the position of the esplora based on the pitch and roll. I had to create code that would print a 0 if the output was 0 at column 40, a r if the output was less than 0 at column 40 and finally a l if the value was greater than 0 at column 40. I also had to write code that would stop the program if I pressed the left button on the esplora. I also had to write code that would switch between the pitch and roll based on the press of a button.

Analysis:

To begin this lab I started by opening lab5.c on Notepad++ so I could edit the code and being solving the problem. I then looked that the functions that the skeleton code had provided me and the parameters for which I would need to write my code for. For example, in the double roll(double x_mag) function, it's stated that -1.0<= x_mag <= 1.0 and for roll I needed to use the equation roll = asin(a_x). So, I had to write if() statements for when x_mag was =< -1.0, >= 1.0, and neither. Then I needed to return whatever that value was. This was my process for most of the lab and reading through the comments at the beginning was very useful to my development.

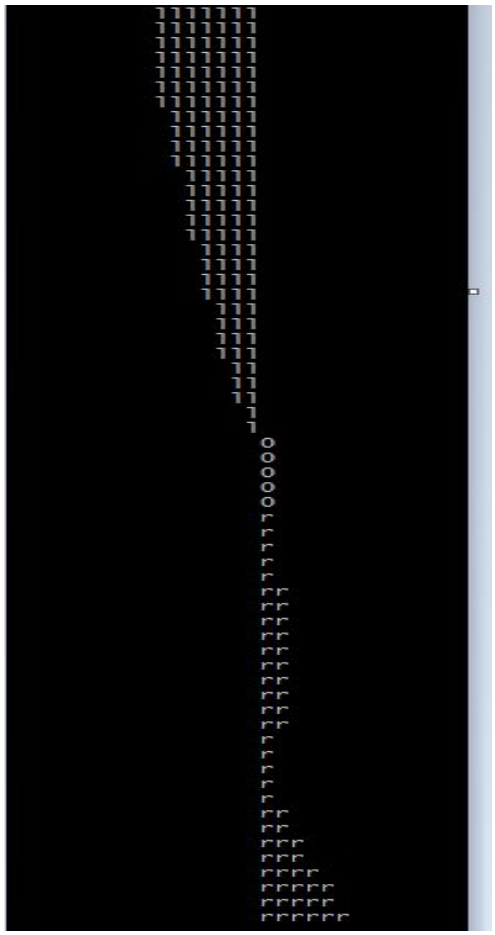
Design:

To begin solving this problem, I had to fill in the do while() loop within the main function to perform the processes asked of me in the comments of the skeleton code. First, I had to scan in the line of input using the read_acc function and analyzing the values x, y, z, b_Up, b_Down, b_Left, and b_Right. These values accounted for the magnitude of x,y,z and the button statuses. I then set roll_rad=roll(x) and pitch_rad=pitch(y) since x and y corresponded to roll and pitch respectively. Finally, within the while loop, I said that when !b_Left the function would stop. So, when anything was happening except the left button being pushed, the function would run, otherwise it would stop. Then I looked at the roll function and since it was stated that -1.0<= x_mag<= 1.0, I had to write if() statements that would find the asin(a_x) for asin(-1.0), asin(1.0), and asin(x_mag). It was very similar for the pitch function except replacing x with y. Thus, my if() statements were looking for asin(a_y), resulting in asin(1.0), asin(-1.0) and asin(y_mag). For the next function, the scaleRadsForScreen, I had to scale the roll values so that they fit on the screen. To do this I multiplied the value rad by (78.0/PI) since the screen was only being mapped to 39 characters. So I had to do 39*2 since a screen is accounting for the length and height, both of which were 39 characters. The next function, print_chars, printed the character 'use' on the screen a certain number of times. For this function I wrote a for() loop that went through and said that if the variable i < num then the function would print use, otherwise nothing would happen.

Finally, for the graph_line function I had to graph a number from -39 to 39 characters on the screen. For this I wrote more if() statements. The first said that if the number < 0 i would use print_chars (39-(number*-1), " ") and print_chars(number*-1, 'l'). This displayed that when the values were greater than 0, an l would be printed. Next, when number >0, I used print_chars (39, " ") and print_chars (number, 'r') stating that when the value was less than 0 the code would print a 'r'. Finally when number==0, print_chars(39, " ") and print_chars(1, '0'), stating that when the value was equal to 0, a zero would be printed by the function.

Testing:

To test this code I went through many trials and errors. I first made sure the 0's would print first since that seemed to be the easiest to code. I followed this by testing for l's and r's when the values were less than or greater than 0. I also made sure they correlated properly to the pitch and roll. I ran into some very frustrating trouble with trying to make the left button stop the code as I had the while() loop reading as while(b_left) instead of while(!b_Left). This was a small error that would cause my code to only print out one value then stop running. Some of my output for the corrected code can be seen below.



Comments:

I found this lab to be much more challenging than the last, which seems to be a common theme with the labs as a whole. One of the problems I ran into was stated above with the while() loop issue but another was trying to output my graph line. I asked one of the TA's for assistance and they helped me come up with graph_line(scaled_value) which correctly performed the output. To scale my values I used the equation rad*(78.0/PI) since the screen was 39 characters on each side. Near the limits of the values, the graph becomes very jagged and barely reaches the end of the screen. This may be in part because of the cables connection and their poor connection but it is very difficult for the 'l' and 'r' values to go to 39 and -39 respectively. Each value , 'l' or 'r', could be converted to degrees from radians by multiplying the radian value by 180/PI.

Implementation:

```
// lab7.c
//
// This is 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.

#include <stdio.h>
#include <math.h>
#define PI 3.141592653589

//NO GLOBAL VARIABLES ALLOWED


//PRE: Arguments must point to double variables or int variables as appropriate
//This function scans a line of explore 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_acc(double* a_x, double* a_y, double* a_z, int* time, int* Button_UP, int*
Button_DOWN, int* Button_LEFT, int* Button_RIGHT);

// PRE: -1.0 <= x_mag <= 1.0
// This function computes the roll of the esplora in radians
// if x_mag outside of -1 to 1, treat it as if it were 1 or -1
// POST: -PI/2 <= return value <= PI/2
double roll(double x_mag);

// PRE: -1.0 <= y_mag <= 1.0
// This function computes the pitch of the esplora in radians
// if y_mag outside of -1 to 1, treat it as if it were 1 or -1
// POST: -PI/2 <= return value <= PI/2
```

```
double pitch(double y_mag);


// PRE: -PI/2 <= rad <= PI/2
// This function scales the roll value to fit on the screen
// POST: -39 <= return value <= 39
int scaleRadsForScreen(double rad);

// 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);

//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);

int main()
{
	double x, y, z;			// magnitude values of x, y, and z
	int b_Up, b_Down, b_Left, b_Right; // variables to hold the button statuses
	double roll_rad, pitch_rad;		// value of the roll measured in radians
	int scaled_value;	// value of the roll adjusted to fit screen display
	int position = 1, t=0;
	//insert any beginning needed code here

	do
	{
		read_acc(&x, &y, &z, &t, &b_Up, &b_Down, &b_Left, &b_Right);// Get line of
input

		roll_rad = roll(x);
		pitch_rad = pitch(y);// Calculate roll and pitch.

		if(b_Up == 1){
			if(position = 1){
				position=2;
```

```
                        }
                        else{
                                position=1;
                        }
                }
        if (position == 1){
                scaled_value = scaleRadsForScreen(-roll_rad);
        }
        else if (position == 2){
                scaled_value = scaleRadsForScreen(-pitch_rad); // Scale your output value
        }
        graph_line(scaled_value); // Output your graph line

                fflush(stdout);
        }
        while (!b_Left); // Modify to stop when left button is pressed
        return 0;
}

int read_acc(double* a_x, double* a_y, double* a_z, int* time, int* Button_UP, int*
Button_DOWN, int* Button_LEFT, int* Button_RIGHT){
        int k, j;
        scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d", time, a_x, a_y, a_z, Button_UP,
Button_DOWN, Button_LEFT, Button_RIGHT, &k, &j);
        if(Button_LEFT != 0){
                        return 1;
        }
return 0;
}

double roll(double x_mag){
        double getValue;
        if(x_mag >= 1.0){
                getValue =  asin(1.0);
        }
        else if(x_mag <= -1.0){
                getValue =  asin(-1.0);
        }
        else{
```

```
		getValue =  asin(x_mag);
	}

	return getValue;
}

double pitch(double y_mag){
	double getValue2;
	if(y_mag >= 1.0){
		getValue2 = asin(1.0);
	}
	else if(y_mag <= -1.0){
		getValue2 = asin(-1.0);
	}
	else{
		getValue2 = asin(y_mag);
	}

	return getValue2;
}

int scaleRadsForScreen(double rad){
	double getValue3;
	getValue3 = rad * (78.0/PI);
	return getValue3;
}

void print_chars(int num, char use){
	int i;
	for(i=0; i < num; i++){
		printf("%c", use);
		}
}

void graph_line(int number){
	if(number < 0 ){
		print_chars(39-(number * -1), ' ');
		print_chars(number * -1, 'l');
```

```
        }
        else if(number > 0){

                print_chars(39, ' ');
                print_chars(number, 'r');
        }
        else if(number == 0){
                print_chars(39, ' ');
                print_chars(1, '0');
        }
        print_chars(1, '\n');
}
```