# LAB REPORT

## LAB 8
## SECTION C

SUBMITTED BY:

SCOTT VLASIC

SUBMISSION DATE:

APRIL 7, 2016

**Problem:**

The problem in this lab was that I wanted to write code that would be able to apply a moving average data. The task was to develop a program that reads from explore.exe the x, y, and z accelerations and computes moving averages on the data in real time. The moving average of length n would also have to compute the average of the last n inputs. In short, I would have to read through the explore.exe file, and output the maximum and minimum to compute the averages. Finally, when I pressed the left button, the program would end.

**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 avg(double buffer[], int num_items) function it stated that I had to compute the average of the first num_items of buffer. So I had to write a for loop that ran whenever some int 'i' was less than or equal to the num_items so that I could take the average of the first num_items. The process for the other two functions was similar as I read through the comments provided and coded based on them. In the main function I made sure to call the other functions correctly. As in lab 7, reading through the comments provided to me in the skeleton code was very important to the development of my code.

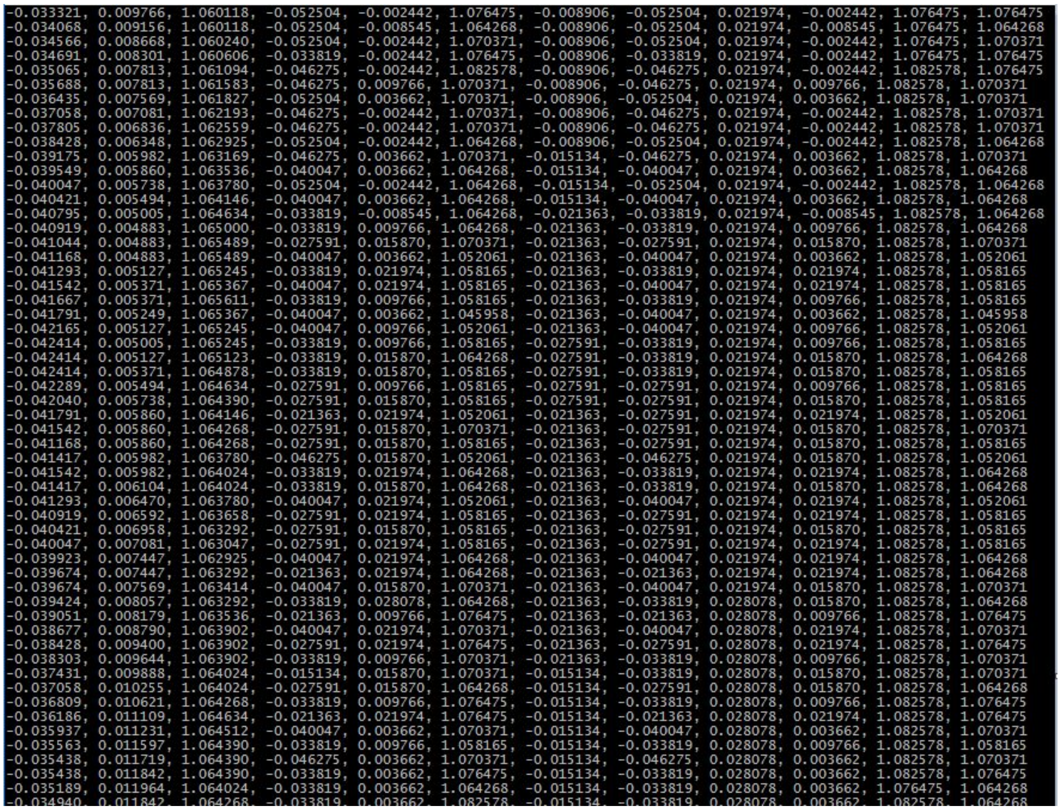
**Design:**

To begin solving this problem, I completed the functions to be used in the main function. First was double avg(double buffer[], int num_items) and my task was to compute the average of the first num_items of buffer. To do this I initialized a variable int i=0 to go through the for loop and a variable for sum and average both equated to 0. My for loop said that when 'i' was less than or equal to num_items it would iterate through and the sum would be modified so that sum = sum + buffer[i]. Once the loop ended, the average = sum/num_items and the return was the average. Next was the void maxmin(double array[], int num_items, double* max, double* min) which wanted me to update the maximum and minimum of the first num_items of the array. To do this I initialized int i=0 for my for loop and set *max = array[0] as well as *min = array[0]. Since both the min and max needed to be updated, pointers needed to be used which is why these were initialized the way they were. In my for loop, when 'i' was less than or equal to num_items, if(array[i] > *max), then the max was updated so that *max = array[i]. Otherwise, *min = array[i] and the min was updated for num_items. Next, I had to work on the void updatebuffer(double buffer[], int length, double new_item) function which asked that I shift length-1 elements of the buffer to the left and put the new_item on the right. To do this I initialized int i =0 for my for loop and iterated through when i<length. The loop would then modify buffer[i] = buffer[i+1] thus moving the elements to the left. Finally, outside the for loop, buffer[length-1] = new_item which put new_item on

the right.  Once all of these functions were completed, I began to work on the main function. Some of the code was already given to us so I had to basically implement my functions.  I started by initializing variables for the acceleration values, the buttons\slider\joystick, the maximum\minimum values, and the averages of the accelerations.  I began by creating a for loop that iterates through when i is less than lengthofavg and reads in data from the esplora.  It also calls the updatebuffer function for the acceleration values for x,y, and z.  Next I created a do while loop so that the loop ran as long as the left button was not being pressed.  In the do part, I scanned in the data again and called the updatebuffer function again.  Next, I set the averages of the acceleration variables equal to the avg(x,lengthofavg) for each one.  This calculated the averages for each variable.  I then printed the averages using printf("%lf, " , avgx).  I also printed the acceleration values themselves by writing printf("%lf, ", accX).  I then had to call the maxmin function for each variable(x,y,z).  To do this I wrote, maxmin(x,lengthofavg, &max, &min).  Since we are using pointers and calling them in the main function, the & sign was needed.  I also printed out these values for each variable using a printf("%lf, %lf", max, min).  To finish the main function I wrote while(!b4) so that the function would end when the left button was pressed.

**Testing:**

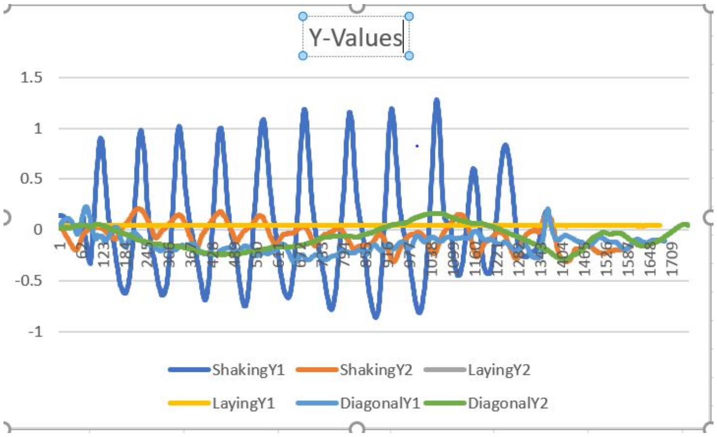
To test this code I went through many trials and errors.  For example, I would forget to put the window length in the execute line of cygwin and would keep wondering as to why my code wouldn't work.  To test these values and the code itself I first just wrote ./explore.exe -p COM? -a -b in to see if the correct things were being displayed.  I then added the ./lab8.exe part and made sure everything was where it was supposed to be.  Some of my output for my code can be seen below.

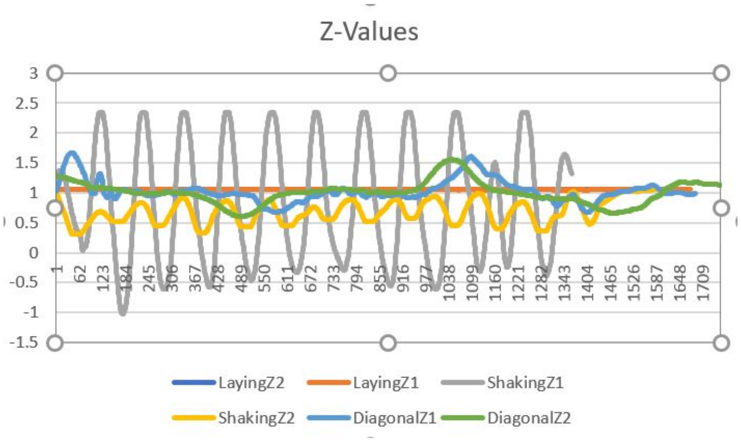


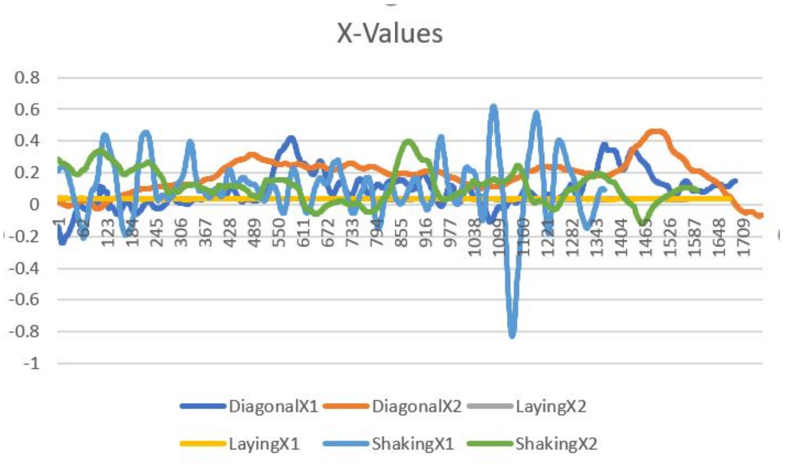
```
-0.033321, 0.009766, 1.060118, -0.052504, -0.002442, 1.076475, -0.008906, -0.052504, 0.021974, -0.002442, 1.076475, 1.076475
-0.034068, 0.009156, 1.060118, -0.052504, -0.008545, 1.064268, -0.008906, -0.052504, 0.021974, -0.008545, 1.076475, 1.064268
-0.034566, 0.008668, 1.060240, -0.052504, -0.002442, 1.070371, -0.008906, -0.052504, 0.021974, -0.002442, 1.076475, 1.070371
-0.034691, 0.008301, 1.060606, -0.033819, -0.002442, 1.076475, -0.008906, -0.033819, 0.021974, -0.002442, 1.076475, 1.076475
-0.035065, 0.007813, 1.061094, -0.046275, -0.002442, 1.082578, -0.008906, -0.046275, 0.021974, -0.002442, 1.082578, 1.076475
-0.035688, 0.007813, 1.061583, -0.046275, 0.009766, 1.070371, -0.008906, -0.046275, 0.021974, 0.009766, 1.082578, 1.070371
-0.036435, 0.007569, 1.061827, -0.052504, 0.003662, 1.070371, -0.008906, -0.052504, 0.021974, 0.003662, 1.082578, 1.070371
-0.037058, 0.007081, 1.062193, -0.046275, -0.002442, 1.070371, -0.008906, -0.046275, 0.021974, -0.002442, 1.082578, 1.070371
-0.037805, 0.006836, 1.062559, -0.046275, -0.002442, 1.070371, -0.008906, -0.046275, 0.021974, -0.002442, 1.082578, 1.070371
-0.038428, 0.006348, 1.062925, -0.052504, -0.002442, 1.064268, -0.008906, -0.052504, 0.021974, -0.002442, 1.082578, 1.064268
-0.039175, 0.005982, 1.063169, -0.046275, 0.003662, 1.070371, -0.015134, -0.046275, 0.021974, 0.003662, 1.082578, 1.070371
-0.039549, 0.005860, 1.063536, -0.040047, 0.003662, 1.064268, -0.015134, -0.040047, 0.021974, 0.003662, 1.082578, 1.064268
-0.040047, 0.005738, 1.063780, -0.052504, -0.002442, 1.064268, -0.015134, -0.052504, 0.021974, -0.002442, 1.082578, 1.064268
-0.040421, 0.005494, 1.064146, -0.040047, 0.003662, 1.064268, -0.015134, -0.040047, 0.021974, 0.003662, 1.082578, 1.064268
-0.040795, 0.005005, 1.064634, -0.033819, -0.008545, 1.064268, -0.021363, -0.033819, 0.021974, -0.008545, 1.082578, 1.064268
-0.040919, 0.004883, 1.065000, -0.033819, 0.009766, 1.064268, -0.021363, -0.033819, 0.021974, 0.009766, 1.082578, 1.064268
-0.041044, 0.004883, 1.065489, -0.027591, 0.015870, 1.070371, -0.021363, -0.027591, 0.021974, 0.015870, 1.082578, 1.070371
-0.041168, 0.004883, 1.065489, -0.040047, 0.003662, 1.052061, -0.021363, -0.040047, 0.021974, 0.003662, 1.082578, 1.052061
-0.041293, 0.005127, 1.065245, -0.033819, 0.021974, 1.058165, -0.021363, -0.033819, 0.021974, 0.021974, 1.082578, 1.058165
-0.041542, 0.005371, 1.065367, -0.040047, 0.021974, 1.058165, -0.021363, -0.040047, 0.021974, 0.021974, 1.082578, 1.058165
-0.041667, 0.005371, 1.065611, -0.033819, 0.009766, 1.058165, -0.021363, -0.033819, 0.021974, 0.009766, 1.082578, 1.058165
-0.041791, 0.005249, 1.065367, -0.040047, 0.003662, 1.045958, -0.021363, -0.040047, 0.021974, 0.003662, 1.082578, 1.045958
-0.042165, 0.005127, 1.065245, -0.040047, 0.009766, 1.052061, -0.021363, -0.040047, 0.021974, 0.009766, 1.082578, 1.052061
-0.042414, 0.005005, 1.065245, -0.033819, 0.009766, 1.058165, -0.027591, -0.033819, 0.021974, 0.009766, 1.082578, 1.058165
-0.042414, 0.005127, 1.065123, -0.033819, 0.015870, 1.064268, -0.027591, -0.033819, 0.021974, 0.015870, 1.082578, 1.064268
-0.042414, 0.005371, 1.064878, -0.033819, 0.015870, 1.058165, -0.027591, -0.033819, 0.021974, 0.015870, 1.082578, 1.058165
-0.042289, 0.005494, 1.064634, -0.027591, 0.009766, 1.058165, -0.027591, -0.027591, 0.021974, 0.009766, 1.082578, 1.058165
-0.042040, 0.005738, 1.064390, -0.027591, 0.015870, 1.058165, -0.027591, -0.027591, 0.021974, 0.015870, 1.082578, 1.058165
-0.041791, 0.005860, 1.064146, -0.021363, 0.021974, 1.052061, -0.021363, -0.027591, 0.021974, 0.021974, 1.082578, 1.052061
-0.041542, 0.005860, 1.064268, -0.027591, 0.015870, 1.070371, -0.021363, -0.027591, 0.021974, 0.015870, 1.082578, 1.070371
-0.041168, 0.005860, 1.064268, -0.027591, 0.015870, 1.058165, -0.021363, -0.027591, 0.021974, 0.015870, 1.082578, 1.058165
-0.041417, 0.005982, 1.063780, -0.046275, 0.015870, 1.052061, -0.021363, -0.046275, 0.021974, 0.015870, 1.082578, 1.052061
-0.041542, 0.005982, 1.064024, -0.033819, 0.021974, 1.064268, -0.021363, -0.033819, 0.021974, 0.021974, 1.082578, 1.064268
-0.041417, 0.006104, 1.064024, -0.033819, 0.015870, 1.064268, -0.021363, -0.033819, 0.021974, 0.015870, 1.082578, 1.064268
-0.041293, 0.006470, 1.063780, -0.040047, 0.021974, 1.052061, -0.021363, -0.040047, 0.021974, 0.021974, 1.082578, 1.052061
-0.040919, 0.006592, 1.063658, -0.027591, 0.021974, 1.058165, -0.021363, -0.027591, 0.021974, 0.021974, 1.082578, 1.058165
-0.040421, 0.006958, 1.063292, -0.027591, 0.015870, 1.058165, -0.021363, -0.027591, 0.021974, 0.015870, 1.082578, 1.058165
-0.040047, 0.007081, 1.063047, -0.027591, 0.021974, 1.058165, -0.021363, -0.027591, 0.021974, 0.021974, 1.082578, 1.058165
-0.039923, 0.007447, 1.062925, -0.040047, 0.021974, 1.064268, -0.021363, -0.040047, 0.021974, 0.021974, 1.082578, 1.064268
-0.039674, 0.007447, 1.063292, -0.021363, 0.021974, 1.064268, -0.021363, -0.021363, 0.021974, 0.021974, 1.082578, 1.064268
-0.039674, 0.007569, 1.063414, -0.040047, 0.015870, 1.070371, -0.021363, -0.040047, 0.021974, 0.015870, 1.082578, 1.070371
-0.039424, 0.008057, 1.063292, -0.033819, 0.028078, 1.064268, -0.021363, -0.033819, 0.028078, 0.015870, 1.082578, 1.064268
-0.039051, 0.008179, 1.063536, -0.021363, 0.009766, 1.076475, -0.021363, -0.021363, 0.028078, 0.009766, 1.082578, 1.076475
-0.038677, 0.008790, 1.063902, -0.040047, 0.021974, 1.070371, -0.021363, -0.040047, 0.028078, 0.021974, 1.082578, 1.070371
-0.038428, 0.009400, 1.063902, -0.027591, 0.021974, 1.076475, -0.021363, -0.027591, 0.028078, 0.021974, 1.082578, 1.076475
-0.038303, 0.009644, 1.063902, -0.033819, 0.009766, 1.070371, -0.021363, -0.033819, 0.028078, 0.009766, 1.082578, 1.070371
-0.037431, 0.009888, 1.064024, -0.015134, 0.015870, 1.070371, -0.015134, -0.033819, 0.028078, 0.015870, 1.082578, 1.070371
-0.037058, 0.010255, 1.064024, -0.027591, 0.015870, 1.064268, -0.015134, -0.027591, 0.028078, 0.015870, 1.082578, 1.064268
-0.036809, 0.010621, 1.064268, -0.033819, 0.009766, 1.076475, -0.015134, -0.033819, 0.028078, 0.009766, 1.082578, 1.076475
-0.036186, 0.011109, 1.064634, -0.021363, 0.021974, 1.076475, -0.015134, -0.021363, 0.028078, 0.021974, 1.082578, 1.076475
-0.035937, 0.011231, 1.064512, -0.040047, 0.003662, 1.070371, -0.015134, -0.040047, 0.028078, 0.003662, 1.082578, 1.070371
-0.035563, 0.011597, 1.064390, -0.033819, 0.009766, 1.058165, -0.015134, -0.033819, 0.028078, 0.009766, 1.082578, 1.058165
-0.035438, 0.011719, 1.064390, -0.046275, 0.003662, 1.070371, -0.015134, -0.046275, 0.028078, 0.003662, 1.082578, 1.070371
-0.035438, 0.011842, 1.064390, -0.033819, 0.003662, 1.076475, -0.015134, -0.033819, 0.028078, 0.003662, 1.082578, 1.076475
-0.035189, 0.011964, 1.064024, -0.033819, 0.003662, 1.064268, -0.015134, -0.033819, 0.028078, 0.003662, 1.076475, 1.064268
-0.034940, 0.011842, 1.064268, -0.033819, 0.003662, 1.082578, -0.015134, -0.033819, 0.028078, 0.003662, 1.082578, 1.064268
```

**Comments:**

I didn't find this lab to be as challenging as the previous two labs but I was still faced with some challenges. One of my problems was with cygwin as stated above but I also had to brush up on my pointer skills. For example, in the main function I had forgotten to include the & when calling the maxmin function and thus my program was messing up. I looked back in the Zyante textbook to fix this issue. For my third motion type I chose to move the esplora slowly and diagonally across my chest. Looking at the graphs of the 20 window lengths vs 100 window lengths, it seems as though the longer windows stay closer to the x-axis of the graphs. It also seems as though generally, the amplitudes of the 100 window lengths is much lower than the 20 length ones. Motion 2's data graph is much more wavy than Motion 3's therefore I would probably want to use the 100 window length so that I received a more accurate representation of the data.


Y-Values
1.5
1
0.5
0
-0.5
-1
1
62
123
184
245
306
367
428
489
550
611
672
733
794
855
916
977
1038
1099
1160
1221
1282
1343
1404
1465
1526
1587
1648
1709
ShakingY1
ShakingY2
LayingY2
LayingY1
DiagonalY1
DiagonalY2


Z-Values
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
1
62
123
184
245
306
367
428
489
550
611
672
733
794
855
916
977
1038
1099
1160
1221
1282
1343
1404
1465
1526
1587
1648
1709
LayingZ2
LayingZ1
ShakingZ1
ShakingZ2
DiagonalZ1
DiagonalZ2


X-Values
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
1
62
123
184
245
306
367
428
489
550
611
672
733
794
855
916
977
1038
1099
1160
1221
1282
1343
1404
1465
1526
1587
1648
1709
DiagonalX1
DiagonalX2
LayingX2
LayingX1
ShakingX1
ShakingX2

**Implementation:**

```
#include <stdio.h>

#define MAXPOINTS 10000

// compute the average of the first num_items of buffer
double avg(double buffer[], int num_items);

//update the max and min of the first num_items of array
void maxmin(double array[], int num_items, double* max, double* min);

//shift length-1 elements of the buffer to the left and put the
//new_item on the right.
void updatebuffer(double buffer[], int length, double new_item);



int main(int argc, char* argv[]) {

        double x[MAXPOINTS], y[MAXPOINTS], z[MAXPOINTS];
        int lengthofavg = 0;
        if (argc>1) {
                sscanf(argv[1], "%d", &lengthofavg );
                printf("You entered a buffer length of %d\n", lengthofavg);
        }
        else {
                printf("Enter a length on the command line\n");
                return -1;
        }
        if (lengthofavg <1 || lengthofavg >MAXPOINTS) {
                printf("Invalid length\n");
                return -1;
        }

        /* Put your code here */
        double accX, accY, accZ; // Variables for the acceleration values
        int b1, b2, b3, b4, b5, b6; // Variables for the buttons, slider, and joystick
        double max, min; // Variables for the maximum and minimum values
```

```
	double avgX, avgY, avgZ; // Averages of the acceleration values
	int i = 0;


	for(i = 0; i < lengthofavg; i++){
		scanf("%lf, %lf, %lf, %d, %d, %d, %d, %d, %d", &accX, &accY, &accZ, &b1, &b2, &b3, &b4, &b5, &b6);
		updatebuffer(x, lengthofavg, accX);
		updatebuffer(y, lengthofavg, accY);
		updatebuffer(z, lengthofavg, accZ);
	}

	do{
		scanf("%lf, %lf, %lf, %d, %d, %d, %d, %d, %d", &accX, &accY, &accZ, &b1, &b2, &b3, &b4, &b5, &b6);
		updatebuffer(x, lengthofavg, accX);
		updatebuffer(y, lengthofavg, accY);
		updatebuffer(z, lengthofavg, accZ);



		avgX = avg(x, lengthofavg);
		avgY = avg(y, lengthofavg);
		avgZ = avg(z, lengthofavg);

		// Prints the averages of avgX, avgY, and avgZ
		printf("%lf, ", avgX);
		printf("%lf, ", avgY);
		printf("%lf, ", avgZ);

		// Prints the accelerations of x, y, and z
		printf("%lf, ", accX);
		printf("%lf, ", accY);
		printf("%lf, ", accZ);

		maxmin(x, lengthofavg, &max, &min);
		printf("%lf, %lf, ", max, min);

		maxmin(y, lengthofavg, &max, &min);
		printf("%lf, %lf, ", max, min);
```

```
        maxmin(z, lengthofavg, &max, &min);
        printf("%lf, %lf", max, min);

        printf("\n");
        fflush(stdout);


    }
    while(!b4); // Ends the program when the left button is pressed





}


double avg(double buffer[], int num_items){
    int i = 0;
    double average = 0;
    double sum = 0;
    for(i=0; i<= num_items; i++){
        sum = sum + buffer[i]; // Only looks for the first num_items
    }
    average = sum/num_items;
    return average;
}

void maxmin(double array[], int num_items, double* max, double* min){
    int i = 0;
    *max = array[0];
    *min = array[0];
    for(i=0; i<num_items; i++){
        if(array[i] > *max){
            *max = array[i]; // Updates the max of the first num_items
        }
        else{
            *min = array[i]; // Updates the min of the first num_items
        }

    }
}

void updatebuffer(double buffer[], int length, double new_item){
    int i = 0;
    for(i=0; i<length; i++){
        buffer[i] = buffer[i+1]; // Shifts the elements to the left
    }
    buffer[length-1] = new_item; // Puts new_item on the right
}
```